ADVANCES IN TRANSFER LEARNING METHODS BASED ON COMPUTATIONAL INTELLIGENCE

A Dissertation presented to

the Faculty of the Graduate School

at the University of Missouri

In Partial Fulfillment

of the Requirements for the Degree

Doctor of Philosophy

---

by

RAYAN S. GARGEES

Dr. Mihail Popescu and Dr. James M Keller, Dissertation Supervisors

DECEMBER 2020

The undersigned, appointed by the Dean of the Graduate School, have examined the dissertation entitled:

ADVANCES IN TRANSFER LEARNING METHODS BASED ON COMPUTATIONAL INTELLIGENCE

presented by Rayan Gargees,

a candidate for the degree of doctor of philosophy and hereby certify that, in their opinion, it is worthy of acceptance.

Dr. Mihail Popescu

Dr. James M. Keller

Dr. Marjorie Skubic

Dr. Marilyn Rantz

This dissertation is dedicated with love to my parents...

ACKNOWLEDGMENTS

First and foremost, I sincerely thank my savior, Jesus Christ, to provide me with the strength, capacity, and dedication to complete this research. His innumerable blessing has made me thrive in my scholarly pursuits. My deep gratitude goes to my advisors, Dr. Mihail Popescu and Dr. James M. Keller, for their experienced guidance and enthusiasm. Their insightful knowledge led me through this research and motivated me to carry my research to a greater extent. It was a great privilege to research under their supervision. Their support has formed my advancement as a scholar. I am thankful to my dissertation committee members, Dr. Marjorie Skubic and Dr. Marilyn Rantz, for serving on the dissertation committee and for their thoughtful feedback and suggestions, which have greatly improved my dissertation. A huge thank you goes to my friends, who always provide needed support and cheerfulness. Your encouragement has meant to me more than you could think. I am incredibly grateful to my family, especially my parents, for their unconditional love, prayers, and always being there for me. They inspired me to strive in pursuit of my goals and supported me to achieve beyond what I believed I could accomplish. I'm forever indebted to you.

TABLE OF CONTENTS

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

List of Algorithms

ABSTRACT

Traditional machine learning and data mining have made tremendous progress in many knowledge-based areas, such as clustering, classification, and regression. However, the primary assumption in all of these areas is that the training and testing data should be in the same domain and have the same distribution. This assumption is difficult to achieve in real-world applications due to the limited availability of labeled data. Associated data in different domains can be used to expand the availability of prior knowledge about future target data. In recent years, transfer learning has been used to address such cross-domain learning problems by using information from data in a related domain and transferring that data to the target task.

The transfer learning methodology is utilized in this work with unsupervised and supervised learning methods. For unsupervised learning, a novel transfer-learning possibilistic c-means (TLPCM) algorithm is proposed to handle the PCM clustering problem in a domain that has insufficient data. Moreover, TLPCM overcomes the problem of differing numbers of clusters between the source and target domains. The proposed algorithm employs the historical cluster centers of the source data as a reference to guide the clustering of the target data. The experimental studies presented here were thoroughly evaluated, and they demonstrate the advantages of TLPCM in both synthetic and real-world transfer datasets.

For supervised learning, a transfer learning (TL) technique is used to pre-train a CNN model on posture data and then fine-tune it on the sleep stage data. We used a ballistocardiography (BCG) bed sensor to collect both posture and sleep stage data to provide a non-invasive, in-home monitoring system that tracks changes in the subjects' health over time. The quality of sleep has a significant impact on health and life. This study adopts a hierarchical and none-hierarchical classification structure to develop an automatic sleep stage classification system using ballistocardiogram

(BCG) signals. A leave-one-subject-out cross-validation (LOSO-CV) procedure is used for testing classification performance in most of the experiments. Convolutional Neural Networks (CNNs), Long Short-Term Memory (LSTM), and Deep Neural Networks DNNs are complementary in their modeling capabilities, while CNNs have the advantage of reducing frequency variations, LSTMs are good at temporal modeling. Polysomnography (PSG) data from a sleep lab was used as the ground truth for sleep stages, with the emphasis on three sleep stages, specifically, awake, rapid eye movement (REM), and non-REM sleep (NREM).

Moreover, a transfer learning approach is employed with supervised learning to address the cross-resident training problem to predict early illness. We validate our method by conducting a retrospective study on three residents from TigerPlace, a retirement community in Columbia, MO, where apartments are fitted with wireless networks of motion and bed sensors. Predicting the early signs of illness in older adults by using a continuous, unobtrusive nursing home monitoring system has been shown to increase the quality of life and decrease care costs. Illness prediction is based on sensor data and uses algorithms such as support vector machine (SVM) and k-nearest neighbors (kNN). One of the most significant challenges related to the development of prediction algorithms for sensor networks is the use of knowledge from previous residents to predict new ones' behaviors. Each day, the presence or absence of illness was manually evaluated using nursing visit reports from a homegrown electronic medical record (EMR) system. In this work, the transfer learning SVM approach outperformed three other methods, *i.e.*, regular SVM, one-class SVM, and one-class kNN.

Chapter 1

Introduction

1.1 Motivation

There is a plethora of data in many areas such as health, images, and industry; in fact, there is an overabundance of data in these fields. However, the most significant issues in the field of machine learning are the limitation of the labeled data sets and heterogeneous data sources. In traditional machine learning, high accuracy and reliability can be obtained if particular assumptions are satisfied. First, the training and testing data should be independent and identically distributed. Second, enough labels data are available to learn a good classification model. Nevertheless, these two assumptions may not always hold in real-world applications due to two facts: First, the new test data coming from fast-evolving information sources usually generate a distribution gap, which causes the unavailability of existing labeled data. Second, in most cases, it is tough and expensive to obtain the labeled data.

To tackle these two problems, transfer learning has become an important and challenging research topic in recent years. Transfer learning (TL) is a new machine learning method that applies the knowledge from related but different domains to

target domains. It relaxes the two underlying assumptions which mentioned earlier in traditional machine learning and aims to overcome the problems when there are few or even not any labeled data in target domains. Related data in different domains can be used to expand the availability of prior knowledge about future target data. In the past few years, transfer learning has been utilized to address such cross-domain learning difficulties by using information from data in the related area and transfer that data to the target task. This can be achieved by enhancing the learning in a new task through transfer the knowledge from a similar but not the same task that has already been learned. Thus, the performance of a transfer learning method can be appropriately examined when represented as a function of several target examples employed in the learning process.

1.2 Transfer Learning

Transferring learning is a humanistic characteristic that has been well studied across education, philosophy, and psychology [3]. In education, Transfer Learning (TL) or the transfer of learning is described as "prior-learned knowledge or skills that affect the way in which new knowledge or skills are learned and performed. The transfer is deemed to be positive if acquisition and performance are facilitated, and negative if they are impeded" [4], [5], [6].

The aim of transfer learning is similar, when utilized to a Computational Intelligence (CI) domain. Thus, transfer learning aims to improve learning in a target domain by acquiring information from a different but related source domain. Transfer learning offers the capability to use earlier acquired knowledge to enhance learning in a related area. Transfer learning can be utilized in varying domains. For instance, a web documentation task has been undertaken to manually label web site documents into defined categories. As a new website is created, the data features and data distribu-

tions are different from those contained within the old site. There is a lack of training data to categorize the new pages. Transfer learning can transfer the classification knowledge to the new domain.

Information availability creates different interpretations of problems. Hence, knowledge is comprised of sourced information and the understanding that is consequently determined [7]. However, a lack of information diminishes the ability to experience a problem. In addition, variations in data and understanding about a problem domain can be interpreted as being included within a knowledge gap. Transfer learning extends the ability to apply earlier acquired knowledge to areas where not much or no information is available, improving the learning. Transfer learning has been applied to varying domains: activity recognition [8], eye tracking [9], image classification [10], and gaming [11]. For example, transfer learning is used in a range of high-performing models that have been developed for image classification and demonstrated on the annual ImageNet Large Scale Visual Recognition Challenge (ILSVRC). A major motivation behind the supervised and unsupervised transfer learning framework comes from environments that lack any prior knowledge in the form of labeled target data.

1.2.1 Definition and Foundation

Transfer learning comprises two principal elements, a Domain, and a Task. The domain can be identified as consisting of two components [12]: a feature space x and marginal probability distribution $P(x)$ where $X = \{x_1, \ldots, x_n\} \in X$. A task consists of a label space Y and a predictive function $f(.)$. The predictive function can be learned from training data, which is constructed as data pairs x_i, y_i where $x_i \in X$ and $y_i \in Y_s$. The source domain can be identified as $Ds = \{(X_{s1}, Y_{s1}), \ldots, (X_{sn}, Y_{sn})\}$ where $x_s \in X_s$ is the data point and $y_s \in Y_s$ is the corresponding label. According to these definitions, transfer learning can be defined as: Given source domain D_s and learning task T_s, a target domain D_t and learning task T_t, transfer learning aims to

improve the learning of a new task T_t through the transfer of the knowledge from related task T_s [13] by the learning of the predictive function in the target domain D_t, where $D_s \neq D_t$ or $T_s \neq T_t$ [12] [7].

There are three main types of transfer learning, a brief definition of each type is given below:

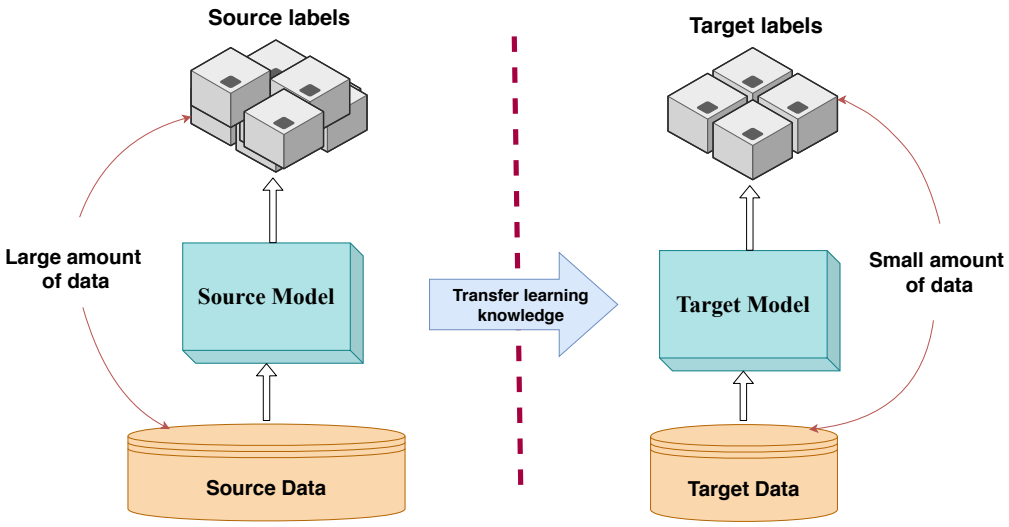


Figure 1.1: Overview of the transfer learning Methodology.

1. **Inductive Transfer Learning:** The source task T_s is different from the target task T_t as $Ts \neq T_t$ in this kind of transfer learning. In the inductive transfer learning, the tasks Domain D and Task T differ either in terms of predictive function ($f_s \neq f_t$), or label spaces ($Y_s \neq Y_t$).

2. **Transductive Transfer Learning:** The source and the target tasks, in transductive transfer learning which is also known as Domain Adaptation(DA), remain the same $T_s = T_t$. However, the source domain and the target domains are different $D_s \neq D_t$. This specific transfer learning case can further be classified into two settings, one where the feature space between the source and the target tasks are different $X_s \neq X_t$ is known as Translated Learning, and the other is where the data distribution is different $P_s(X) \neq P_t(X)$, known as Cross-domain Transfer.

3. **Unsupervised Transfer Learning:** Unsupervised Transfer learning is similar to the other forms of transfer learning, which looks to improve the target domain's predictive function by extracting information from the source to assist the target. Taking a similar stance to standard unsupervised learning, the data within neither the target or source domain contains labels. for a given source domain D_s and source task T_s, a target domain D_t and a target learning task T_t, the idea is to assist in improving the learning of the target predictive function $f_t(.)$ in the target domain D_t employing the knowledge in source domain D_s and source task T_s, where $T_s \neq T_t$ and $D_s \neq D_t$. The unsupervised transfer learning case arises where clustering the cross-domain data instances, which can be used to cluster a small collection of unlabeled data in the target domain with the guidance of a large amount of unlabeled data in the source domain.

1.3 Major Contribution

The contributions of this dissertation can be summarized as follows:

- A novel clustering framework is Proposed to learn target tasks from limited unlabeled target data and related; differing source labeled data. The publication of this contribution can be seen in [14].
  - Works in applications where data are limited and insufficient for useful clustering or are polluted by unknown noise or outliers such as the COVID-19 dataset where it is tough to obtain sizeable data and experiments cannot be performed.
- A novel deep learning architecture is Proposed to classify sleep stages and sleep posture based on CNN and LSTM to benefit transfer learning. The publication of this contribution can be seen in [15].

  - It helps people to have a repetitive and regular sleep cycle for good sleep which results in more persistent morphological templates in longitudinal studies.
  - It helps nurses in reducing changes in pressure ulcers in bed-bound patients.

- Employ a transfer learning SVM approach to address the cross-resident training problem and compare it with the other three methods, regular SVM, one-class SVM, and one class kNN. The publication of this contribution can be seen in [16].
  - It helps predict the early signs of illness in older adults by using a continuous, unobtrusive nursing home monitoring system to increase the quality of life and decrease the cost of care.

1.4 Dissertation Organization

The subsequent chapters of the dissertation are organized as follows:
The background and related work are introduced in Chapter 2. Its purpose is to provide a general context for the manuscripts' work, which comprise the next chapters. Each manuscript includes a more thorough introduction and review of the state-of-the-art. Chapter 3 details the implementation of the proposed Transfer Learning Possibilistic C-Means (TLPCM) framework. This includes the experimental studies which were thoroughly evaluated. Moreover, the advantages of TLPCM in both synthetic and real-world transfer datasets are demonstrated. This chapter's main content is from the following publication: "TLPCM: Transfer Learning Possibilistic C-Means" [14]. Chapter 4 reports a new deep learning-based hierarchical classification method for automatic sleep stage classification based on CNN and LSTM with the

help of transfer learning. The main content of this chapter is from the following publication: "Non-invasive Classification of Sleep Stages with a Hydraulic Bed Sensor Using Deep Learning" [15]. Chapter 5, explores the practicability of using transfer learning SVM for early illness recognition based on sensor data by using training data from another resident. The main content of this chapter is from the following publication: "Early Illness Recognition in Older Adults Using Transfer Learning" [16]. Finally, Chapter 6 provides some summary remarks regarding the work described in this dissertation and a discussion of possible future work.

Chapter 2

Background

This chapter is divided into two different sections. Section 2.1 will cover the machine learning and Neural network background.

2.1 Machine learning and Neural network

Machine learning (ML) is a sub-field of computer engineering that develops automatically through experience. It is also considered a subset of artificial intelligence. The machine learning algorithm can be interpreted as a method to estimate a given probability distribution. A discriminative model learns the conditional probability distribution $p(y|x)$, *i.e.*, given an input, what is the probability that the label is y. A generative model learns the joint probability distribution $p(x, y)$ where x is the input data, and y is the labels for x. Based on the available data, we can divide the machine learning algorithms into precisely three types of learning models. As their name indicates, supervised learning algorithms require labeled training data, and unsupervised learning algorithms do not require labeled training data, while semi-supervised algorithms require a combination of both labeled and unlabeled training data.

- **Supervised machine learning:** Supervised learning describes a class of systems and algorithms that determine a predictive model using data samples with known outcomes. The model is learned by training across an appropriate learning algorithm such as neural networks or random forests that naturally work through some optimization routine to minimize a loss or error function [17].In the supervised machine learning algorithm, if target data stated in some class, then the problem is called the classification problem; however, if the target data are continuous, then the problem is called the regression problem.

- **Unsupervised machine learning:** Unsupervised learning is a machine learning algorithm used to obtain inferences from datasets comprising input data without pre-existing labels and with a minimum of human supervision. The most typical unsupervised learning method is cluster analysis, which is used for experimental data analysis to discover hidden patterns or categories in data [18]. Clustering is the method to divide a collection of data samples into individual groups, called clusters. The clustering idea is to utilize the clustering algorithm on the set of data samples without building a model for the algorithm using training data. The algorithm attempted to group the data samples into clusters on pre-existing associations in the data instances itself.

- **Semi-supervised machine learning:** Semi-supervised learning is a type of machine learning algorithms that combines a small amount of labeled data with many unlabeled data during training. Hence, it is a setup in which among the training data samples available, some of the samples are not labeled, and the learning model uses additional unlabeled data samples to describe the shape of the underlying data distribution for the new samples in an effective way [19]. The semi-supervised learning algorithm's goal is to understand how labeled and unlabeled samples can enhance learning behavior.

2.1.1 Deep Learning

Deep learning is a sub-field of machine learning which has evolved from the traditional approaches to artificial neural networks. Artificial neural networks are computational systems initially inspired by the human brain. They comprise many computational units, called neurons, which perform an essential operation and pass the information to further neurons. Besides, recent deep learning improvements have surpassed expectations of what many people thought was potentially in machine learning and pattern recognition [20, 21].

Fields such as speech recognition, computer vision, and natural language processing have seen a significant progress forward toward solving problems once thought difficult. For example, in the computer vision field, a computer can perform better than a human on the object recognition, and in the field of speech recognition and natural language processing Siri and Cortana are synthetic systems that sound reasonably lifelike.

Deep Learning is a set of methods that are a natural progression of traditional neural network techniques. These include:

- *New types of layers:* The most significant difference between traditional neural networks and deep learning is adopting new layers in the network. Traditional neural network research focused on fully connected layers, in which every neuron in one layer is connected to every neuron in the next. While many of these layer types existed in the past, they usually could not significantly affect various issues in training. Convolutional neural networks learn filter banks that are convolved with the original data. The filters can also be represented as a fully connected layer where the edges of the edges are tied together to replicate the convolution operation. This weight sharing structure allows for fewer parameters than having each weight be unique and directly accounts for structure in the data. This procedure creates a network with only forward connections,

letting backpropagation work as expected at the cost of limiting the impact of the recurrent connections. Because of this, only a few architectures have seen widespread use and success in classification tasks. Recurrent neural networks (RNN) are a type of neural network robust for modeling sequence data such as time series or natural language. Diagrammatically, an RNN layer uses a for loop to iterate over the time steps of a sequence while maintaining an internal state that encodes information about the time steps it has seen so far.

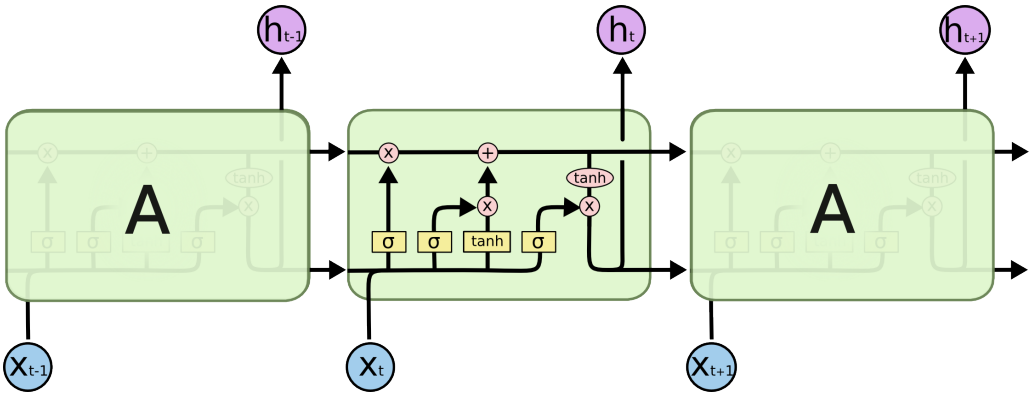


Figure 2.1: Architecture of an LSTM network [2].

Hochreiter and Schmidhuber [22] introduced Long Short-Term Memory networks in 1997, usually just called "LSTMs," which is a unique kind of RNN, capable of learning long-term dependencies. The authors were refined and popularized by many people in the following work. They work exceptionally well on a sizeable range of problems and are now widely used. LSTMs are explicitly designed to prevent the long-term dependency problem. Remembering information for long periods is practically their default behavior [21]. All recurrent neural networks have the form of a chain of repeating modules of the neural network. In standard RNNs, this repeating module will have a straightforward structure, such as a single tanh layer. The architecture of an LSTM network is shown in Fig. 2.1. LSTMs are made to function specifically with time series or sequential data. Each time its own LSTM unit processes a point in the data,

the results are passed to the next layer and the next time point within the same layer. Several gates for each LSTM unit control the flow of information. The gate is a combination of a sigmoidal activation unit and pointwise multiplication. These gates control the amount of information that flows from a one-time point to the following one. The amount of information is the output of each unit, and other LSTM unit functions.

- *Stochastic Gradient Descent (SGD) and related algorithms*: Gradient descent [23] is a first-order iterative method for optimizing an objective function with suitable smoothness properties. In neural networks, it is used in combining with backpropagation to update the weights in the network. It is formally identified with the update rule:

$$\overrightarrow{x}_k = \overrightarrow{x}_{k-1} - \eta \nabla f(\overrightarrow{x}_{k-1}) \tag{2.1}$$

where:

  - $\overrightarrow{x}_k$ is the current point in the space,
  - $\overrightarrow{x}_{k-1}$ is the previous point in the space,
  - η is the learning rate,
  - $\nabla f(\overrightarrow{x}_{k-1})$ is the gradient of the value of the function being optimized at the previous point.

SGD is a derivative of traditional gradient descent, differing in that the error function is calculated using only a few samples selected randomly instead of the whole data set for each iteration [21]. This is both easier to use and more efficient for training datasets that do not fit in memory. Moreover, adding randomness to the optimization can avoiding local minima. In Gradient Descent,

a term called "batch" denotes the total number of samples from a dataset used for calculating the gradient for each iteration. In standard Gradient Descent optimization, like Batch Gradient Descent, the batch is taken to be the whole dataset. Although using the whole dataset is useful for getting to the minima less noisy and randomly, but the problem arises when our datasets get big. The addition of momentum terms, which biases the gradient in the direction of recently calculated gradients, significantly improved the ability to train deep models by further increasing convergence speed [24]. Newer, SGD derived algorithms, such as Adaptive Moment Estimation (ADAM), calculate per parameter adaptive learning rates, allowing even more efficient training at memory cost [25].

- *New Activation Functions:* One of the most extensive and most persistent challenges in developing neural networks is the vanishing gradient problem. The error is essentially multiplied with the values between 0 and 1 repeatedly, as the gradient is propagated back along with the network. This causes the error, and hence, the update trends toward 0 exponentially, resulting in little to no ability to update the first layers in a multilayer network. Sigmoid and Tanh activation functions, which were historically the most prominently activation function, are particularly susceptible to this problem due to having a first derivative that rapidly tends toward zero as a neuron saturates. The sigmoid function is defined as: $\frac{1}{1+e^{-x}}$. The Rectified Linear Unit (ReLU), which the most generally used activation function in deep learning models. The function returns 0 if it receives any negative input, but for any positive value x it returns that value back. Hence, it can be written as $f(x) = \max(0, x)$. and other modern activation functions have larger gradients and saturate less quickly, thus avoiding the vanishing gradient problem more effectively [26].

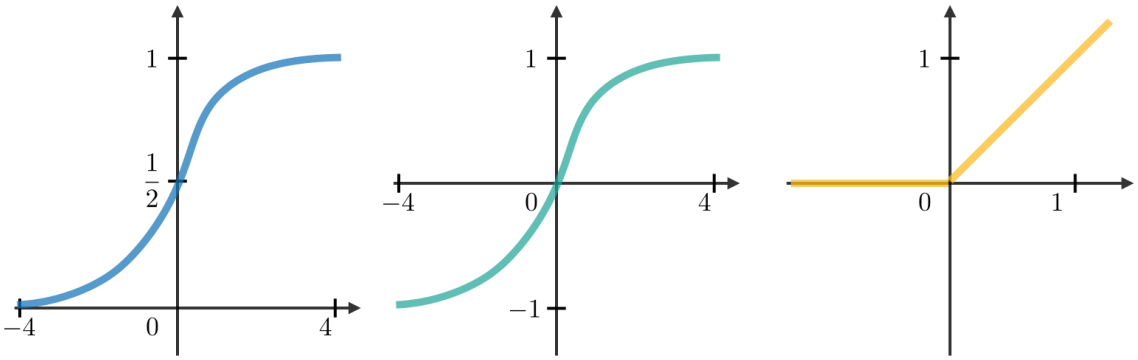


Figure 2.2: Activation functions; (a) Sigmoid (logistic) function (b) Tanh function and (c) ReLU function.

- *Dropout and other regularizers:* Large networks typically require large volumes of training data, and there may not be enough data available to train different networks on different subsets of the data. In other words, deep neural nets with a large number of parameters are potent machine learning systems. Moreover, training many architectures is difficult because finding optimal hyperparameters for each architecture is a daunting task, and training each extensive network requires much computation. Dropout is a technique that addresses both these concerns. The term "dropout" describes dropping out units in a neural network [27].

  Dropout is a technique in which a random set of neurons from each layer is omitted from both updating and classification during a training pass through the data. This successfully allows a single model to act as an ensemble, a group of classifiers that act in union to produce a classification. Additionally, since different sets of neurons will be participating from one pass to the next, dropout avoids training data's direct memorization. In other words, it prevents overfitting and provides a method of approximately combining exponentially many different neural network architectures efficiently. Overfitting is a severe problem in such networks. It is the phenomenon of learning patterns that

happen to be present in the training data by random chance and not present in general, is a constant threat in deep learning due to the immense numbers of parameters involved.

2.2 Related Work

Various papers have recently developed transfer learning frameworks tightly coupled with supervised and unsupervised transfer learning. This section gives a brief review of the associated work in these fields.

2.2.1 Unsupervised Transfer Learning

Hua Zuo *et al.* [1] proposed a fuzzy-regression transfer-learning method based on fuzzy rules to manage the problem of approximating the value of the target for regression. A Takagi-Sugeno fuzzy-regression model was created to transfer knowledge from a source domain to a target domain. Experimental results confirmed that the proposed fuzzy-regression transfer-learning method significantly improved the performance of existing models when solving regression problems in the target domain. The methods of [1] solve regression problems in the target domain when only a small amount of data is available.

In [28], a novel clustering framework was designed, and the transfer learning-based maximum-entropy clustering (TL MEC) algorithm was proposed. Historical cluster centers are employed in the TL MEC algorithm and in the membership of past data, which are used as references to guide the clustering of the current data, which distinctly improves its performance in two areas: clustering effectiveness and privacy protection.

In [29], a transfer-learning algorithm for document analysis based on the density-based spatial clustering of applications with noise (DBSCAN) algorithm was devel-

oped. Documents are classified into different classes using weight-adjustment strategies. DBSCAN is used to extract feature clusters; then, the dataset is clustered using KNN. The proposed algorithm showed a better performance in the experimental results.

The authors in [30] proposed a novel multitask clustering paradigm that performs multiple related clustering tasks together and employs the relationships among these tasks to improve clustering performance. The authors' goal in [30] was to learn the subspace shared by all clustering tasks to enable the transfer of information from different tasks to other tasks. This multitask clustering method was extended to the transductive transfer classification.

P. Qian *et al.* [31], inspired by transfer learning, devised a cluster prototype and fuzzy membership jointly leveraged (CPM-JL) framework for classic cross-domain maximum entropy clustering (CDMEC) to propose a corresponding algorithm, CPM-JL-CDMEC. Moreover, the authors proposed the dedicated validity index, *i.e.*, a fuzzy membership-based cross-domain difference measurement (FM-CDDM), to aid self-adaptive parameter setting in CPM-JL-CDMEC. The new algorithm shows acceptable clustering effectiveness and robustness in the experimental results.

A clustering-based approach was developed by Thendral *et al.* [32]to overcome the problem of data sparsity by using knowledge from a denser associated domain. The authors focus on finding a reference in a sparsely rated domain by using the knowledge from a highly rated domain with the same users and then rating the items in both domains. The results of [32] proved that clustering is a transfer learning technique that can be used to mitigate the cold-start and sparsity problems in recommended systems.

Hua Zuo *et al.* [33] proposed an infinite Gaussian mixture model (IGMM) with active learning to improve the performance of the model. The authors used the IGMM to identify the source and target domains' data structure, thereby providing

a promising solution to the domain selection problem. In addition, [33] exploited the interactive query in active learning to correct imbalances in the knowledge and thus generalized the learning model. The idea of active learning in [33] leads to an increase in the number of labeled data in the target domain by actively labeling the target domain's most informative data.

2.2.2 Supervised Transfer Learning

Wenyuan Dai *et al.*, [34] addressed the issue of classifying text documents across different distributions utilizing transfer learning. The labeled training data are available, but the problem is that this data has a different distribution from the unlabeled test data. The authors developed a transfer-learning algorithm based on the Naive Bayes classifiers, called Naive Bayes Transfer Classifier (NBTC). Their solution is first to estimate the initial probabilities under a distribution D_l of one-labeled data set and then use an EM algorithm to revise the model for a different distribution D_u of the unlabeled test data.

Sinno Jialin Pan *etal.* [35] presented a location-estimation method based on Manifold co-Regularization, which is a machine learning technique for building a mapping function between data using transfer learning. The authors described LeManCoR, a system for adjusting the mapping function between the signal space and physical location space on different periods based on Manifold Co-Regularization. Moreover, they showed that LeManCoR could effectively transfer the knowledge between two time periods without requiring an excessively new calibration effort.

In recent years, interest in utilizing BCG has increased.This recovery is fundamentally made possible by the ongoing development of piezoelectric sensors, signal processing [36–38], and new technology to ensure noninvasive long-term vital signs monitoring. *Rosales etal.* [39] enhanced system abilities to capture a heartbeat signal from four subjects using a new hydraulic transducer configuration. [39] also presented

a new approach for detecting the accuracy of heartbeats from ballistocardiogram (BCG) signals by extracting three features based on the peak-to-valley amplitude differences in the BCG signals and then combining the fuzzy C means clustering and k-means approaches.

In [40] researchers examined the mechanisms for BCG waveforms' genesis and proved their proposed model capable of assisting during surgery. The authors also presented an algorithm to detect individual heart beats and beat-to-beat interval lengths in BCG using healthy subjects. Wang *etal.* [41] proposed a new technique for extracting respiratory signals from ballistocardiography. They also developed a structure to detect ballistocardiography signals without the subjects' awareness. Besides, [42]developed an adaptive interface cancellation algorithm to derive the respiration component, attenuating the uncorrelated noise, and improving the shared information. This method delivers an accurate means to monitor respiration components without the subjects' awareness.

In [43], the authors found that positional patients having most of their breathing abnormalities in the supine posture and who became non-positional patients had a significant gain in weight and a significant increase in the apnoea-hypopnoea index, mainly in lateral apnoea–hypopnoea index. On the contrary, non-positional patients who became positional patients had a significant decrease in weight and showed a significant improvement in the apnoea-hypopnoea index, again mainly in the lateral apnoea–hypopnoea index. Gardner *et al.* [44], have been shown that body movements, a prominent behavioral aspect of sleep, impressively changes with age.

Daulatzai *et al.* [45], recorded the number of apnea and hypopnea in the supine position and in the lateral position of sleep. The authors evaluated the number of apnea and hypopnea in responders and non-responders. They found that avoidance of sleep in the supine (horizontal) position positively influences on the frequency and severity of obstructive sleep apnea. OSA patients are "Responders" when they re-

spond to positional therapeutic measure, while those in whom sleeping vertically does not decrease apnea hypopnea index (AHI) and is referred to as "Non- Responders."

In [46] proposed a model that achieved the state-of-the-art performance for ECG heartbeat arrhythmia detection on the commonly used benchmark dataset from the MIT-BIH Arrhythmia Database. We then utilize our model in an active learning process to perform patientadaptive heartbeat classification tasks on the non-wearable ECG dataset from the MIT-BIH Arrhythmia Database and the wearable ECG dataset from the DeepQ Arrhythmia Database.

Long short-term memory recurrent neural networks improve over the general recurrent neural networks, which possess a vanishing gradient problem. As stated in Hochreiter *etal*. [22], LSTM RNNs address the vanishing gradient problem commonly found in ordinary recurrent neural networks by incorporating gating functions into their state dynamics. At each time step, an LSTM maintains a hidden vector h and a memory vector m responsible for controlling state updates and outputs. More concretely, Graves *etal*. [47] define the computation at time step t as follows :

$$
\begin{aligned}
g^u &= \sigma(\boldsymbol{W^u h_{t-1}} + \boldsymbol{I^u x_t}) \\
g^f &= \sigma(\boldsymbol{W^f h_{t-1}} + \boldsymbol{I^f x_t}) \\
g^o &= \sigma(\boldsymbol{W^o h_{t-1}} + \boldsymbol{I^o x_t}) \\
g^c &= tanh(\boldsymbol{W^c h_{t-1}} + \boldsymbol{I^c x_t}) \\
m_t &= \boldsymbol{g^f} \odot \boldsymbol{m_{t-1}} + \boldsymbol{g^u} \odot \boldsymbol{g^c} \\
h_t &= tanh(\boldsymbol{g^o} \odot \boldsymbol{m_t})
\end{aligned}
\tag{2.2}
$$

where σ is the logistic sigmoid function, $\odot$ represents elementwise multiplication, $\boldsymbol{W^u}, \boldsymbol{W^f}, \boldsymbol{W^o}, \boldsymbol{W^c}$are recurrent weight matrices and $\boldsymbol{I^u}, \boldsymbol{I^f}, \boldsymbol{I^o}, \boldsymbol{I^c}$are projection matrices. While LSTMs possess the ability to learn temporal dependencies in

sequences, they have difficulty with long term dependencies in long sequences. The attention mechanism proposed by Bahdanau *etal*. [48] can help the LSTM RNN learn these dependencies.

In [49], the authors implemented a long short-term memory (LSTM) network with a convolutional neural network (CNN) to automatically diagnose Coronary artery disease CAD ECG signals accurately. The system has the potential to be deployed in clinical settings to assist cardiologists in making an objective and reliable diagnosis of ECG signals.

Chapter 3

TLPCM: a Transfer-Learning Possibilistic C-Means Algorithm

3.1 Introduction

Cluster analysis has been widely used in many fields, such as image segmentation, data mining, and unsupervised pattern recognition, as an unsupervised data processing method. Many methods based on different concepts have been proposed for solving these problems. The most widely used soft clustering algorithm is fuzzy c-means (FCM) [50]. Nevertheless, FCM is sensitive to noise and outliers because of its probabilistic constraint. Krishnapuram and Keller [51] proposed a concept of typicality or possibility. Typicality values are the only constraint in this approach, and these values must lie in the interval [0, 1]. The resulting algorithm is known as the possibilistic c-means (PCM) algorithm. Furthermore, it has been referred to as a mode-seeking algorithm because good clusters (dense regions) can be shaped with a proper estimation of the scale parameters. PCM can cluster data even with an unknown number of clusters. Traditional clustering approaches, such as k-means and FCM, usually work well in an ideal condition where the data are sufficient and pure.

However, noise and interference data are omnipresent in the real world [52].

Thus, two main problems must be addressed when working with datasets. First, the data capacity may be strictly limited for specific reasons, such as those found in biomedical datasets or a lack of accumulated data, especially in some emerging fields. Second, too much noise can be a problem in the original data. Several advanced cluster models have been developed to address the problems of a lack of information and data impurity, such as co-clustering [53], multitask learning [54], semisupervised learning [55], and transfer learning [56, 57]. We believe that transfer learning is the most promising model due to its specific mechanisms. Transfer learning works in a minimum of two domains, *i.e.*, the source domain and the target domain. It first identifies useful information from the source domain, within its classification of either data or knowledge, and then it moves this information into the target domain to guide the training procedure. Unsupervised transfer learning works to improve the target domains' predictive function of the target domains when the current data of the source domain are insufficient or impure; it utilizes helpful information from related fields or previous studies. Here, we will illustrate the power of our algorithm by applying it to real-world biomedical data. Our data were collected at a large nursing home and consist of complex data, such as ballistocardiogram (BCG) data, used to detect early illness.

One application of this is in home monitoring systems for cardiovascular disease. More than 37% of the United States population is affected by the cardiovascular disease [16, 58, 59]. A home monitoring system has been developed to detect the early signs of cardiovascular abnormalities and to avoid the fatal consequence of advanced cardiovascular disease. Monitoring cardiac parameters during sleep can provide critical information about the subject's health. In this work, real-world data from a hydraulic bed sensor (HBS) are used to capture BCG signals. The HBS was positioned under the mattress and used to perform both cardiac and respiration

monitoring in the home. Overall activity, behavioral patterns, and in-home gait patterns were also captured using motion sensors and a depth camera. An automated health change alert system is currently installed in 75 senior apartments and regularly runs as part of a longitudinal study.

The hydraulic bed sensor was developed at the Center for Eldercare and Rehabilitation Technology (CERT) at the University of Missouri. The BCG device provides a non-invasive, low-cost, robust solution for capturing physiological parameters during sleep [39, 60–62]. The bed sensor has a pressure sensor, a transducer, and a water tube. The transducer is 50 cm long and 6 cm wide and is filled with 0.4 liter of water. One end of the transducer is connected to an integrated silicon pressure sensor for measuring the vibrations of the discharge hose. It is placed under the mattress to provide sleeping comfort. The outputs are connected to the filtering circuit (Maxim MAX7401) that consists of a 741 op-amp amplifier and an 8th-order integrated Bessel filter. The four-channel signal is sampled and quantized to 12-bit precision. The BCG signal acquired from the sensor is superimposed on the respiration signal. Four transducers are placed in a parallel alignment underneath the subject's mattress to guarantee sufficient coverage. The four matching transducers are independent; thus, the quality of data collected by these transducers might vary depending on the subject's sleeping position, the type of bed (*e.g.*, its material and thickness) and the physical characteristics of the subject (*e.g.*, age and body mass index (BMI)).

In this work, we take advantage of transfer learning and combine it with PCM's unique features to propose a new approach, transfer-learning possibilistic c-means (TLPCM), that works with a limited dataset. Transfer learning helps to strengthen its clustering robustness and promote its ability to deal with more complex data. Here, we validate our proposed algorithm with synthetic and real-world data. This is the first work that uses transfer learning with PCM and applies it to synthetic and

real-world data to the best of our knowledge.

Despite the achievements in the above research, there exist limitations associated with each that are worth mentioning:

Despite the achievements in the abovementioned research (see 2.2.1), there exist limitations associated with each that are worth mentioning.

- The numbers of clusters in the source domain and the target domain are the same.

- Most of the research based on fuzzy clustering methods with transfer learning has problems associated with the constraint on memberships. This constraint causes the fuzzy clustering methods to generate memberships that can be taken as degrees of sharing but not as degrees of typicality. Thus, the memberships of two data points in a given cluster that are equidistant from the cluster prototype can be significantly different, and the memberships of two data points in a given cluster that are randomly far from each other can be the same.

In this work, the transfer-learning possibilistic c-means algorithm (TLPCM) is proposed to address these challenges.

The rest of this chapter is organized as follows:

In Section 3.2, we briefly review the transfer learning concept. Section 3.3 presents some preliminaries on clustering. The details of our approach are presented in 3.4. In Section 3.5, the synthetic and real-world datasets used in this work are described. The experimental results are reported in Section 3.6. Finally, some concluding remarks and future directions are provided in Section 3.7.

3.2 Transfer Learning

The key idea of transfer learning is to improve learning in a target domain by acquiring information from a different but related domain called the source domain. Traditional machine learning strategies work under several assumptions that entail that the target and source domains have similar feature distributions. However, it is necessary to relax these assumptions in practice and allow that the two domains may have different feature distributions or that the classification task may have to change. In this situation, traditional machine learning techniques often fail to model the test data correctly. As shown in Fig. 3.1, transfer learning offers the ability to use previously acquired knowledge to improve learning in a related area. In addition, it can be applied to different task domains [56, 63, 64]. In each transfer-learning algorithm, a source task is related to the source domain, and the target task is linked to the target domain. The transfer learning system procedure has two steps: First, the source task is learned, and second, the knowledge is transferred from the first step and used to improve the learning of the target task. If the model is not modified for new situations, the prediction accuracy will drop significantly. Transfer-learning techniques have been proposed to handle these types of cases. The idea behind transfer learning is that the knowledge learned previously can be applied to solve a new problem with a better or faster solution. In the machine learning area, transfer learning has many benefits, such as saving time when learning new tasks, requiring less information from experts, and making the learned model more robust.

In the next section, we present preliminary elements (c-means, fuzzy c-means, and possibilistic c-means) necessary for our proposed algorithm, TLPCM.

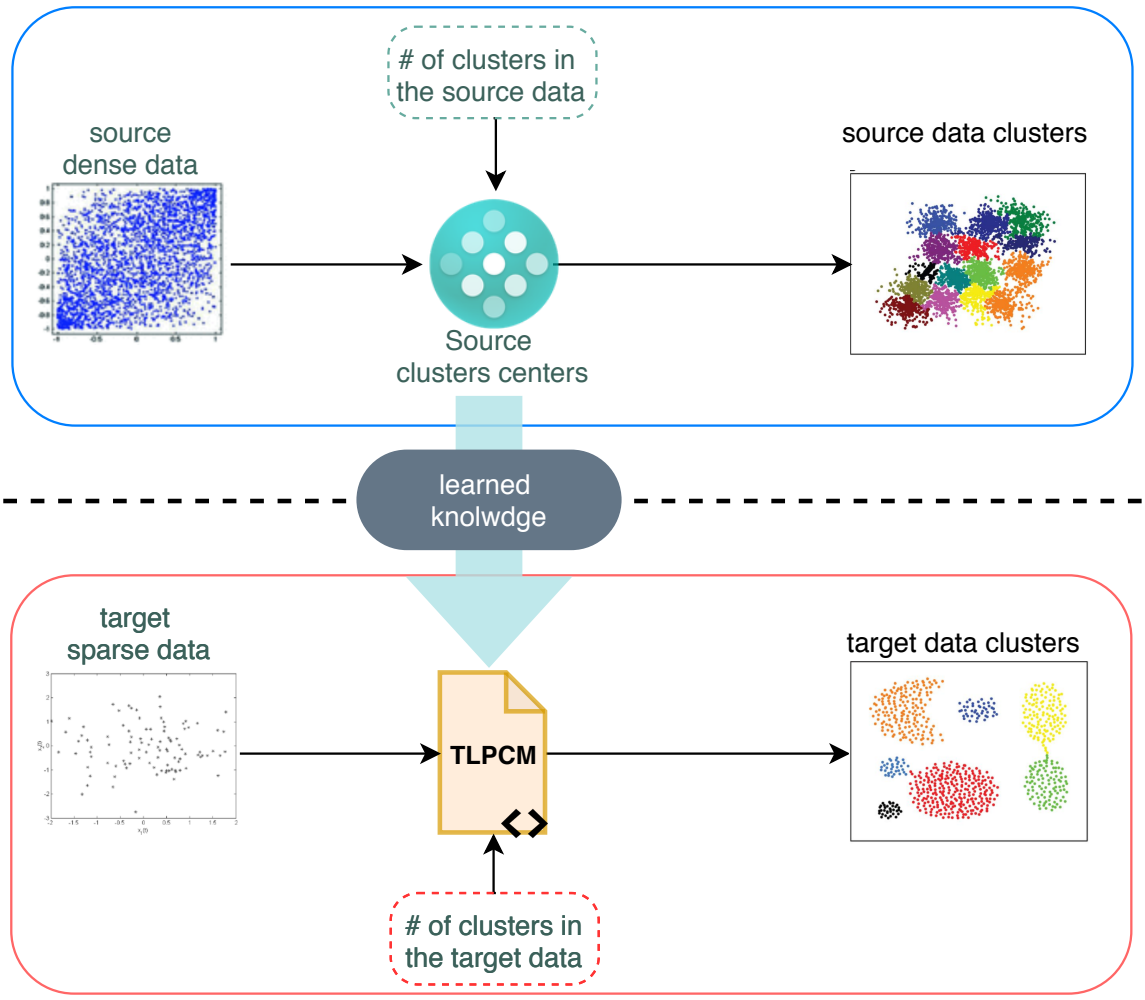


Figure 3.1: Overall framework of transfer learning clustering.

3.3 C-means, FCM, and PCM

There are many partition-based clustering methods, but the three well-known algorithms are the k-means clustering algorithm [65], the fuzzy c-means (FCM) clustering algorithm [50], and the possibilistic c-means (PCM) clustering algorithm [51]. These are the most popular approaches because of their general applicability to real life problems. k-means clustering is a typical example of an algorithm that achieves iterative modification of K cluster centers. The objective function used to derive the cluster center update is:

$$J(V,U) = \frac{1}{2}\sum_{n=1}^{N}\sum_{k=1}^{K} u_{nk}\|x_n - v_k\|^2, \tag{3.1}$$

where v_k is the k_{th} cluster's position vector, x_n is the data point being evaluated, and u_{nk} is the binary membership vector for data point x_n. The FCM algorithm [50], a generalization of the C-means (K-means), is an unsupervised clustering algorithm in which a dataset is grouped into C clusters with every point in the dataset belonging to every cluster to a certain degree [66]. The FCM algorithm is nearly identical to C-means except for one important point. Instead of a binary value of u, FCM has $u \in$ [0,1]. This means that a data point can have partial membership in multiple clusters. The objective function to minimize for FCM is:

$$FCM: \quad \min_{U,V} J_{FCM} = \sum_{j=1}^{C}\sum_{i=1}^{N} u_{ij}^{q}\,\|x_i - v_j\|^2 \tag{3.2}$$

$$\begin{aligned} s.t. \quad & u_{ij} \in [0,1],\ \forall_{i,j} \\ & 0 < \sum_{i=1}^{N} u_{ij} < N, \forall_j \quad \text{and} \\ & \sum_{j=1}^{C} u_{ij} = 1, \forall_i. \end{aligned}$$

Differentiating equation (3.2) with respect to u_{ij} and v_j and setting them to 0 leads to the equations

$$u_{ij} = \frac{1}{\sum_{k=1}^{C}\left(\frac{d(x_i,v_j)}{d(x_i,v_k)}\right)^{1/(q-1)}}, \tag{3.3}$$

$$v_j = \frac{\sum_{i=1}^{N} u_{ij}^q x_i}{\sum_{i=1}^{N} u_{ij}^q}, \tag{3.4}$$

where C is the number of clusters ($j = 1, 2, ..., C$), N is the number of data points, and $x_i \in R^d$ is the i^{th} data sample, where ($i = 1, 2, ..., N$). $V = [v_1, ..., v_C]^T$ is the matrix of C cluster centers with $v_j \in R^d$. $U = [u_{ij}]_{N \times C}$ is the fuzzy partition matrix whose element u_{ij} denotes the membership of the i^{th} data sample belonging to the i^{th} class for all $i = 1, \ldots, n$. Here, the fuzzifier parameter $q \in (1, \infty)$. A standard approach for optimizing the FCM model is to randomly initialize V and then alternately update U and V using the necessary conditions for the extrema of J_{FCM}.

FCM is a fuzzy clustering model; moreover, it is primarily a partitioning algorithm. Consequently, it will find a fuzzy C partition of a given dataset regardless of how many clusters are actually present in the dataset. In other words, each component of the partition may or may not correspond to a cluster.

In contrast, PCM is a mode-seeking algorithm; *i.e.*, each component generated by PCM corresponds to a dense region in the dataset. In PCM, the prototypes are automatically attracted to dense regions in the feature space as the iterations proceed [67]. Additionally, PCM has a significant advantage for anomaly detection compared with FCM. The noise points and outliers are often very distant from the primary clusters. In PCM, the farther a noise point is from a dense area, the smaller the membership degree. Noise points and outliers will be assigned a small degree of membership when using PCM, which gives them little influence on the estimation of the prototypes and the final partition. Merely relaxing the constraint will lead to a trivial solution of the memberships; the criterion function is minimized by assigning all memberships as zero. Therefore, a constraint is added to the objective function of PCM to make representative data points have high membership and unrepresentative

points have low membership. The objective function that satisfies the requirements can be formulated as:

$$PCM: \quad \min_{U,V} J_{PCM} = \sum_{j=1}^{C}\sum_{i=1}^{N} u_{ij}^{q} \, \|x_i - v_j\|^2 + \sum_{j=1}^{C} \eta_j \sum_{i=1}^{N} (1-u_{ij})^q \tag{3.5}$$

$$s.t. \quad u_{ij} \in [0,1], \ \forall_{i,j}$$

$$0 < \sum_{i=1}^{N} u_{ij} < N, \forall_j \quad \text{and}$$

$$\max_j u_{ij} > 0, \forall_i.$$

The update equation (a necessary condition for a minimum) for the typicality value was found to be:

$$u_{ij} = \frac{1}{1 + \left(\frac{d^2(x_i, v_j)}{\eta_j}\right)^{1/(q-1)}}, \tag{3.6}$$

where $U = [u_{ij}]_{N\times C}$ denotes the possibilistic partition matrix, u_{ij} denotes possibilistic membership, and η_j is a "scale" parameter, which corresponds to the size of the cluster or "zone of influence" [67].

The first term of J_{pcm} is the same as the FCM objective function, which leads to the minimization of the weighted distances, and the second term, which acts as a penalty, is used to avoid the trivial solution of $u_{ij} = 0 \ \forall_{i,j}$.

Although the PCM algorithm effectively solves the noise sensitivity problem of FCM,

some new problems have been caused by the PCM clustering model. PCM is sensitive to initialization. If there is poor initialization, PCM might converge to an insignificant partition where some or all of the clusters are coincident and other clusters may go unobserved.

3.4 The Proposed Method, TLPCM

The existing literature and many applications have demonstrated the relatively good performance of classic PCM. However, PCM effectiveness strictly depends on the preconditions, *i.e.*, abundant data and sufficient information. Without this precondition, PCM might be invalid. This is a strong incentive to strive for accuracy and thoroughness in subsequent PCM research. In this work, we adopt transfer learning, construct an objective function by utilizing the historical matrices of cluster centers $\hat{v}$, and propose the corresponding transfer-learning possibilistic C-means (TLPCM) algorithm. The historical matrices of cluster centers $\hat{v}$ are employed to construct the novel objective function

$$\begin{aligned} TLPCM : \min_{U,Z,V} J_{TLPCM} = & \sum_{i=1}^{N}\sum_{j=1}^{C_t} u_{ij}^{q_t}\|x_i - v_j\|^2 \\ & + \sum_{j=1}^{C_t}\eta_j \sum_{i=1}^{N}(1-u_{ij})^{q_t} \\ & + \Gamma \sum_{j=1}^{C_t}\sum_{k=1}^{C_s} z_{jk}^{q_s}\|\hat{v}_k - v_j\|^2 \\ & + \Gamma \sum_{j=1}^{C_t}\beta_j \sum_{k=1}^{C_s}(1-z_{jk})^{q_s} \end{aligned} \tag{3.7}$$

$$
\begin{aligned}
s.t. \quad & u_{ij} \in [0,1], \forall_{i,j}, \\
& z_{jk} \in [0,1], \forall_{j,k}, \\
& 0 < \sum_{i=1}^{N} u_{ij} \leq N, \forall_j, \\
& 0 < \sum_{k=1}^{C_s} z_{jk} \ \leq \ C_s, \forall_j, \\
& \max_j \ u_{ij} > 0, \forall_i, \text{and} \\
& \max_k \ z_{jk} > 0, \forall_j,
\end{aligned}
$$

where:

N: number of data samples

C_t: number of cluster centers in the target data

C_s: number of cluster centers in the source data

$\hat{v}_k$: source cluster center

v_j: target cluster center

q_t: target data fuzzifier

q_s: source data fuzzifier

u_{ij}: possibilistic membership between the target data and target centers

z_{jk}: possibilistic membership between the source centers and target centers

Γ: coefficient of the term of the historical cluster centers

η_j and β_j: suitable positive numbers.

The first term requires that the distances from the feature vectors to the target data prototypes be as small as possible, and the second term forces u_{ij} to be as large as possible, consequently avoiding a trivial solution. The first and second terms are inherited directly from the original PCM algorithm. The third term is used to

learn the knowledge from the source domain; it demands that the distances from the source data prototypes to the target data prototypes be as small as possible. The target domain will gain more knowledge from the source domain if the j^{th} cluster in the target domain and k^{th} cluster in the source domain are more similar. The fourth term forces z_{jk} to be as large as possible, hence avoiding the trivial solution.

It is worth noting that the first two constraints ensure that no cluster in the source or target data is empty, while the last two guarantee that no point has zero typicality in all clusters.

We propose TLPCM clustering algorithms whose general form is given in Algorithm 1. Most of the research discussed in Section 2.2 assumed that the number of clusters in the source data and target data are the same. However, in most practical applications, this assumption does not always hold. The beauty of our approach is its ability to implement transfer clustering with different numbers of clusters in the source and target domains. By using a similar optimization strategy in PCM, the novel update equations for u_{ij}, z_{ij}, and v_j in TLPCM can be determined as described below.

3.4.1 Update the Membership u_{ij} Equation

$$\frac{\partial J}{\partial u_{ij}} = q_t u_{ij}^{(q_t-1)} \|x_i - v_j\|^2 - \eta_j q_t (1-u_{ij})^{(q_t-1)}$$

Let $\frac{\partial J}{\partial u_{ij}} = 0$. Thus, we obtain

$$\begin{aligned}
\left(\frac{u_{ij}}{1-u_{ij}}\right)^{q_t-1} &= \frac{\eta_j}{\|x_i - v_j\|^2} \\
\frac{1-u_{ij}}{u_{ij}} &= \left(\frac{\|x_i - v_j\|^2}{\eta_j}\right)^{1/(q_t-1)} \\
\frac{1}{u_{ij}} &= \left(\frac{\|x_i - v_j\|^2}{\eta_j}\right)^{1/(q_t-1)} + 1
\end{aligned}$$

$$u_{ij} = \frac{1}{1 + \left(\frac{d^2(x_i, v_j)}{\eta_j}\right)^{1/(q_t - 1)}}. \tag{3.8}$$

3.4.2 Update the Membership z_{jk} Equation

$$\frac{\partial J}{\partial z_{jk}} = \Gamma q_s z_{jk}^{q_s - 1} \|\hat{v_k} - v_j\|^2 - \Gamma \beta_j q_s (1 - z_{jk})^{q_s - 1}$$

Let $\frac{\partial J}{\partial z_{jk}} = 0$.

$$\begin{aligned}
\left(\frac{z_{jk}}{1 - z_{jk}}\right)^{(q_s - 1)} &= \frac{\beta_j}{\|\hat{v_k} - v_j\|^2} \\
\frac{1 - z_{jk}}{z_{jk}} &= \left(\frac{\|\hat{v_k} - v_j\|^2}{\beta_j}\right)^{1/(q_s - 1)} \\
\frac{1}{z_{jk}} &= \left(\frac{\|\hat{v_k} - v_j\|^2}{\beta_j}\right)^{1/(q_s - 1)} + 1
\end{aligned}$$

$$z_{jk} = \frac{1}{1 + \left(\frac{d^2(\hat{v_k}, v_j)}{\beta_j}\right)^{1/(q_s - 1)}} \tag{3.9}$$

3.4.3 Update the Equation of the Cluster Centers v_j

$$\frac{\partial J}{\partial v_j} = -2 \sum_{i=1}^{N} u_{ij}^{q_t} \|x_i - v_j\| - 2\Gamma \sum_{k=1}^{C_s} z_{jk}^{q_s} \|\hat{v_k} - v_j\|$$

Let $\frac{\partial J}{\partial v_j} = 0$.

$$-\sum_{i=1}^{N} u_{ij}^{q_t} x_i + \sum_{i=1}^{N} u_{ij}^{q_t} v_j = \Gamma \sum_{k=1}^{C_s} z_{jk}^{q_s} \hat{v_k} - \Gamma \sum_{k=1}^{C_s} z_{jk}^{q_s} v_j$$

$$v_j(\sum_{i=1}^{N} u_{ij}^{q_t} + \Gamma \sum_{k=1}^{C_s} z_{jk}^{q_s}) = \sum_{i=1}^{N} u_{ij}^{q_t} x_i + \Gamma \sum_{k=1}^{C_s} z_{jk}^{q_s} \hat{v}_k$$

$$v_j = \frac{\sum_{i=1}^{N} u_{ij}^{q_t} x_i + \Gamma \sum_{k=1}^{C_s} z_{jk}^{q_s} \hat{v}_k}{\sum_{i=1}^{N} u_{ij}^{q_t} + \Gamma \sum_{k=1}^{C_s} z_{jk}^{q_s}} \tag{3.10}$$

The values of η_j and β_j need to be chosen depending on the desired bandwidth of the possibility distribution for each cluster. If all clusters are expected to be similar, these values can be the same for all clusters. In addition, η_j determines the relative degree to which the second term is essential compared with the first; A target cluster with larger values of η_j and mostly empty will have more freedom to move as the iterations proceed since more data will lie on its domain.

In TPCM, β_j determines the "zone of influence" of a target cluster. Hence, a source cluster center $\hat{v}_k$ will have little influence on the estimates of the prototype parameters of a target cluster v_j if $d^2(\hat{v}_k, v_j)$ is large when compared with . Conversely, "close" source cluster centers to a target cluster center relative to β_j exert more influence on the target typicalities and hence prototype movement. β_j decides also the relative degree to which the fourth term is necessary compared with the third. As in the PCM, these parameters can be set as constants or can be iteratively updated as weighted averages of the distances between the source and target cluster centers at each iteration.

In this work, η_j is estimated using the following equation [67]:

$$\eta_j = \frac{\sum_{i=1}^{N} u_{ij}^{q_t} d^2(x_i, v_j)}{\sum_{i=1}^{N} u_{ij}^{q_t}}. \tag{3.11}$$

In the same way, we estimate β_j using the following equation:

$$\beta_j = \frac{\sum_{k=1}^{C_s} z_{jk}^{q_s} d^2(\hat{v}_k, v_j)}{\sum_{k=1}^{C_s} z_{jk}^{q_s}}. \tag{3.12}$$

Algorithm 1 lists the steps of the TLPCM method. Like any clustering algorithm, TLPCM has a few hyper-parameters that need to be set ahead of time: the number of clusters in the source and target data C_s, C_t; the source and target fuzzifiers q_s, q_t; the partition matrices u_{ij}, z_{jk}; and the regularization coefficient Γ. However, all these parameters, except Γ, can be chosen using strategies also employed in the PCM algorithm. Hence, the only additional parameter that we need to initialize, and it is not part of the PCM algorithm, is the regularization coefficient Γ. This coefficient Γ controls the influence of the cluster prototypes from the source domain on those in the target domain, that is, the higher the coefficient Γ, the greater the influence of the source values. The values of fuzzifiers determine the fuzziness of the final possibilistic C-partition that affects the shape and overlap among the resulting membership functions. When q_s or $q_t \to 1$, the membership function is hard, and when q_s or $q_t \to \infty$ the memberships are maximally fuzzy. By removing the last constrain of FCM (see equation 3.2), the membership functions of c clusters become independent of each other. This makes PCM and TLPCM sensitive to the initialization of the partition matrices since nothing prevents the algorithms from converging to degenerate partition matrices where all clusters are identical or similar to each other. Therefore, the FCM algorithm is called to initialize the partition matrix u_{ij} according to Krishnapuram and Keller's recommendation [51], whereas uniform random numbers are utilized to initialize z_{jk}. Ultimately, the values of η_j and β_j are calculated utilizing equations 3.11 and 3.12, respectively. In Section 3.6, a couple of experiments were conducted to show the effect of η_j and β_j on the results.

Algorithm 1: TLPCM

Input: X= Dataset, $\hat{V}$=Source centers, q=Fuzzifier, d= Equ. distance, *StopThresh*, C_s=No. source centers, C_t=No. target centers, MaxIter.
Output: V= Target centers, U= Partition matrix

Initialization
begin
  Obtain the cluster prototypes $\hat{V}$ in the source data with the original PCM algorithm; (Skip this step if the cluster prototypes $\hat{V}$ in the source data are known beforehand)
  while *diff*> *StopThresh && iter* < *MaxIter* **do**
    foreach $x_i \in X$ **do**
      foreach $v_j \in V$ **do**
        calculate d(x_i,v_j)
      end
    end
    foreach $\hat{v_k} \in \hat{V}$ **do**
      foreach $v_j \in V$ **do**
        calculate d($\hat{v_k}$,v_j)
      end
    end
    foreach $x_i \in X$ **do**
      foreach $v_j \in V$ **do**
        update η_j utilizing equation (3.11)
      end
    end
    foreach $v_j \in V$ **do**
      foreach $\hat{v_k} \in \hat{V}$ **do**
        update β_j utilizing equation (3.12)
      end
    end
    foreach $v_j \in V$ **do**
      foreach $\hat{v_k} \in \hat{V}$ **do**
        update possibilistic membership between source centers and target centers using equation (3.9)
      end
    end
    foreach $x_i \in X$ **do**
      foreach $v_j \in V$ **do**
        update partition matrix utilizing equation (3.8)
        if $d == 0$ **then**
          $d = \epsilon$
        end
      end
    end
    foreach $v_j \in V$ **do**
      update target cluster centers utilizing equation (3.10)
    end
    diff = norm$(V_{iter-1}$ - $V)$
  end
  Output the final cluster prototypes V and memberships U in the target domain.
end

3.5 Dataset Description

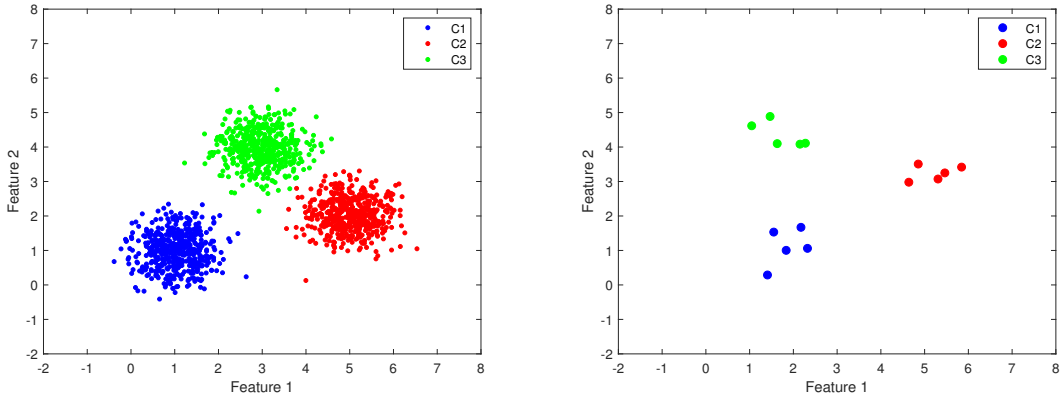
3.5.1 Synthetic Dataset

To assess the abilities of our new approach, we created three synthetic datasets. The first synthetic dataset is S_1, which has 1500 instances in the source domain and 15 instances in the target domain, with three clusters in both the source and target datasets. Fig. 3.2 shows that there is a good partitioning for the target data. To show the differences between the source data and the target data, the input data in both domains are displayed in Fig. 3.3. The S_1 datasets is similar to the one used in [1]. The mean values and covariance matrices of the source data and target data are shown in Table 3.1

The second synthetic dataset isS_2 , which has 4000 instances with four clusters in the source data and 18 instances with six clusters in the target data, which are shown in Fig. 3.4. In the second synthetic dataset, we chose the case where the target dataset has a different distribution than the source dataset from which we want to enable knowledge transfer. The third synthetic dataset is S_3 (see Fig. 3.5), where target clusters had different densities and a larger spread; this dataset is used to test η_j and β_j parameters.

Table 3.1: Distributions of Source Data and Target Data for S_1 [1].

Source data		Target data	
Mean values	Covariance	Mean values	Covariance
$\mu_1 = [1 \quad 1]$	$\sigma_1 = \begin{bmatrix} 0.5^2 & 0 \\ 0 & 0.5^2 \end{bmatrix}$	$\acute{\mu_1} = [2 \quad 1]$	$\sigma_1 = \begin{bmatrix} 0.4^2 & 0.1 \\ 0.1 & 0.4^2 \end{bmatrix}$
$\mu_2 = [5 \quad 2]$	$\sigma_2 = \begin{bmatrix} 0.5^2 & 0 \\ 0 & 0.5^2 \end{bmatrix}$	$\acute{\mu_2} = [5 \quad 3]$	$\sigma_2 = \begin{bmatrix} 0.5^2 & 0.1 \\ 0.1 & 0.5^2 \end{bmatrix}$
$\mu_3 = [3 \quad 4]$	$\sigma_3 = \begin{bmatrix} 0.5^2 & 0 \\ 0 & 0.5^2 \end{bmatrix}$	$\acute{\mu_3} = [2 \quad 4]$	$\sigma_3 = \begin{bmatrix} 0.5^2 & 0.1 \\ 0.1 & 0.5^2 \end{bmatrix}$



(a) (b)

Figure 3.2: Synthetic dataset S_1 ; (a) source data and (b) target data

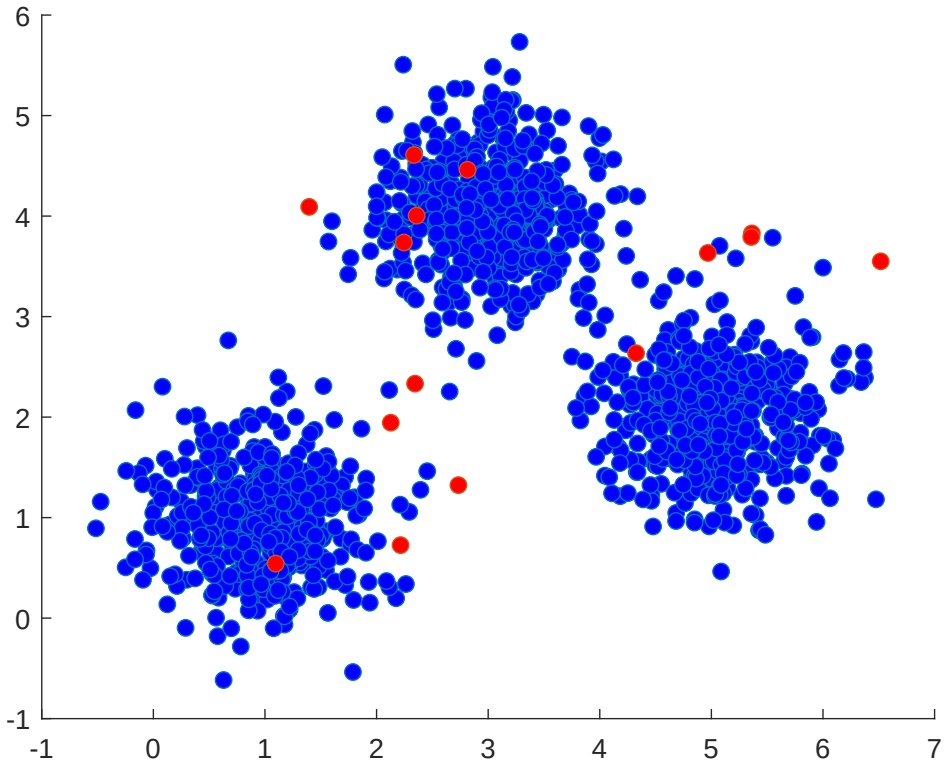
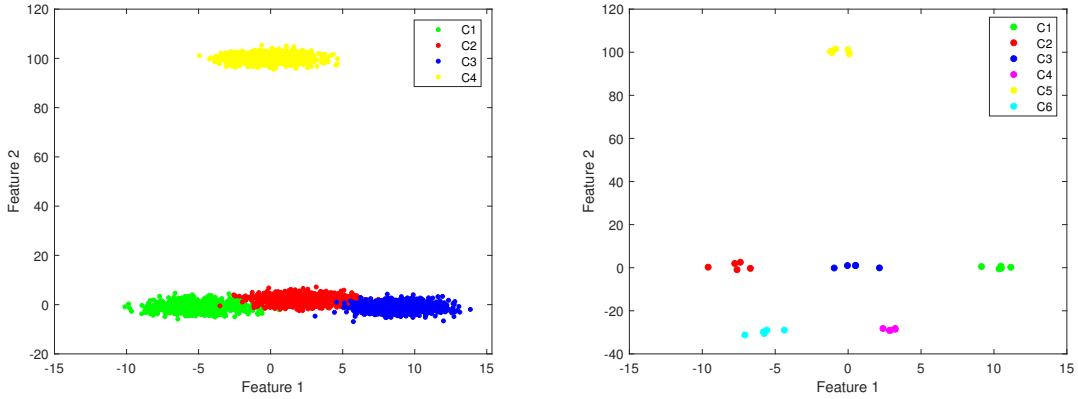
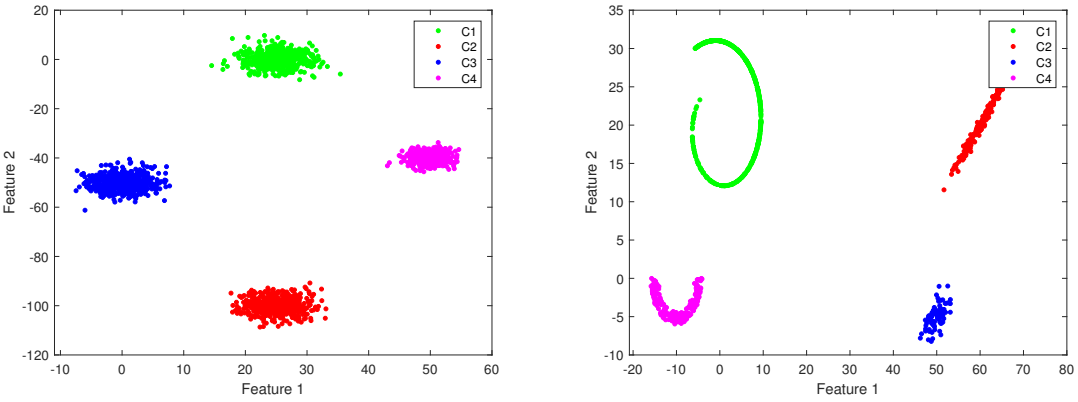


Figure 3.3: Differences between the source data and the target data; source (blue) and target (red).



(a) (b)

Figure 3.4: Synthetic datasetS_2 ; (a) source data and (b) target data



(a) (b)

Figure 3.5: Synthetic datasetS_3 ; (a) source data and (b) target data

3.5.2 Real-World Datasets

Three different kinds of real-world datasets are used in this work to prove the validity of our algorithm:

3.5.2.1 Bed Sensor Dataset

The sensor data of three different TigerPlace residents with different numbers of normal and abnormal days were used, as shown in Table 3.2. Technology has a tremendous impact on the elderly by providing them with full creative and independent

lives. Sensor technology is utilized in the TigerPlace facility to help elderly residents manage their illness, stay healthy, and be as independent as possible. The Americare Corporation of Sikeston, MO with the help of the Missouri Sinclair School of Nursing (SSON), established TigerPlace, a senior living community that was opened in mid-2004 just a few miles from the MU campus [94].

Table 3.2: Data for three residents in TigerPlace.

Resident No.	**Total records**	**Positive days ("feel good")**	**Negative days ("feel bad")**
Resident1	441	360	81
Resident2	744	709	35
Resident3	499	164	335

An integrated monitoring system was placed in 47 TigerPlace apartments with the University of Missouri's IRB approval. Only non-wearable sensors were used for monitoring because they are more acceptable to older adults and are unobtrusive. The monitoring began in the fall of 2005, accumulating , as of 2020, an average of two years' worth of data for each resident. There is a data logger for each resident's apartment, which is used to collect the data from wireless sensors. The data logger tags the data with the date and time, then logs it in a document that is sent to a database on a protected server via a wired network connection. Fig. 3.6 shows the architecture of our monitoring system.

The main components of the monitoring system are a data logger, sensor network, and electronic health record (EHR) system; a reasoning system for decline detection and recognition; a secure Web-based interface to display the data for clinicians and researchers; and an alert manager to inform clinicians of possible problems. Each sensor network consists of several types of sensors placed in the resident's apartment, including passive infrared motion (PIR) sensors, a Microsoft Kinect depth sensor and bed sensors. Newer apartments have in-house built depth sensors. The PIR

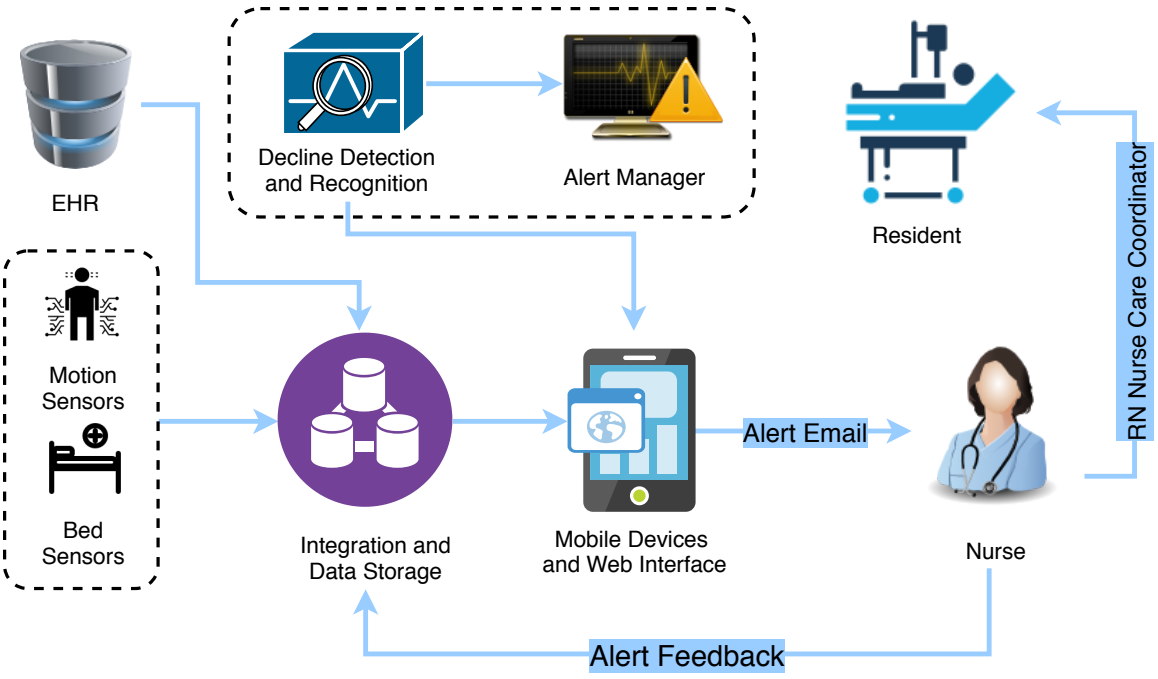


Figure 3.6: The TigerPlace sensor network architecture comprises a data logger, sensor network and EHR system, a reasoning system, a secure Web-based interface, and an alert manager.

motion sensors are set in various places, such as the bedroom, the kitchen, the living room, and the bathroom. In addition, some of the residents have sensors of this type installed on the door of their refrigerator, kitchen cabinets, and even drawers.

A resident's movement within the apartment is captured by the PIR sensors that produce a signal as long as there is movement around them. The bed sensors are arranged in sets of four pneumatic (which was replace by a hydraulic version later) sensor strips placed under the bed mattress. Unlike the motion sensors, the pneumatic bed sensor captures three types of signal (heart rate, respiration and bed restlessness), which are discretized into three or four levels of severity. The newer hydraulic sensor produces continuous sensor values, instead. Sensitivity is the key for early illness recognition. If the resident does not feel well, the sleep and motion pattern will be altered.

Four features are used in this study to represent residents' behavior: the total number of motion sensors firing, restlessness, pulse rate, and breathing for each hour

of the day. The bed sensor can continue to track the residents' restlessness, pulse, and breathing as long as the person lies on the bed. The number of features is doubled by dividing each feature into a day and night hours. Fig. 3.7 shows pairwise relationships in a Resident1 dataset with layered kernel density estimate (KDE). It can be noticed from this figure that breathing in the day hours has a high and wide kernel density range compared to the other feature. Moreover, the two classes interfere with each other, which makes them more challenging in clustering or classification.

All clinical records for these residents were collected. These records included their medication, nursing visits, and hospitalizations. The labels for each hour (normal and abnormal) were added manually based on the nursing visit reports and other clinic records. These labels represent the ground truth for our data, which was later used to calculate the Rand index and thereby check the accuracy of our proposed algorithm. Data processing was rather simple. The steps were as follows: first, we aggregated the sensor data. Features 1-4 were the sum of the sensor data for the night hours (12 a.m. to 7 a.m.), and features 5-8 provided the sum of the sensor data for the day hours (7 a.m. to 12 a.m.) and represented the sensor activity prior to a nursing visit. Fig. 3.8 shows 3D visualization for the three residents. Then, the data were normalized for the three residents, after which the data were passed through our algorithms as described in Section 3.4.

3.5.2.2 Forensic Glass Dataset

This dataset includes 214 fragments of glass that were initially collected by B. German for a study of the context of a criminal investigation [68]. Each fragment has a measured refractive index and chemical composition (weight percent of oxides of Na, Mg, Al, Si, K, Ca, Ba and Fe) [69, 70]. After cleaning the data, 14 fragments were discarded, and 200 were used in this work. The glass types are as follows: 1. building windows, float processed; 2. building windows, non-float processed; 3. vehicle

Figure 3.7: Pairwise relationships in a Resident1 dataset with layered kernel density estimate (KDE).

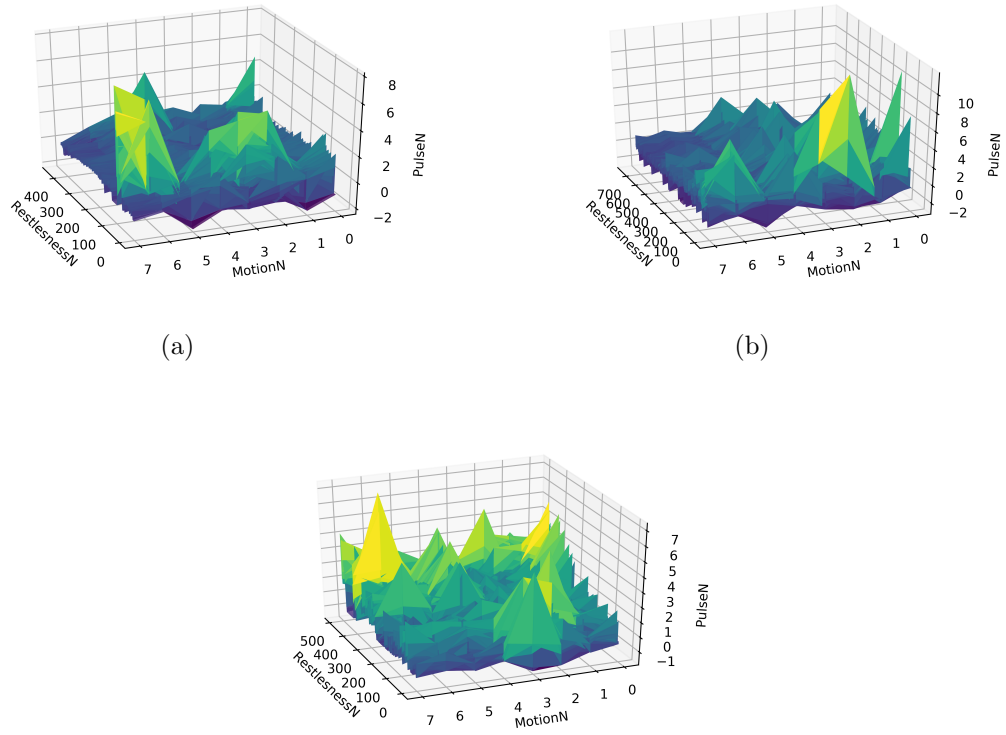


Figure 3.8: 3D visualization for the three residents ; (a) Resident1 (b) Resident2 and, (c) Resident3.

windows, float processed; 4. vehicle windows, non-float processed; 5. containers; 6. tableware; and 7. headlamps. The glass type "windows, non-float processed" is not available in this dataset. Therefore, we ultimately used 6 types of glass in this work. It is worth mentioning that these data are unbalanced, as shown in Fig. 3.9, which provided us with a good challenge to test the validity of our algorithm.

3.5.2.3 Chronic Kidney Disease Dataset

The third real-world dataset used in this work is the chronic kidney disease dataset taken from the University of California Irvine machine learning repository [70]. This set of data was collected over two months in India with 400 rows and 25 features in

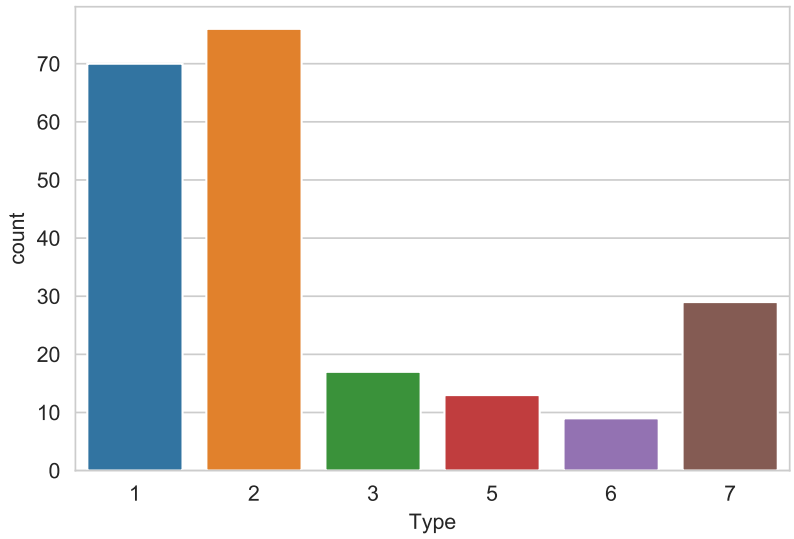


Figure 3.9: Glass dataset distribution.

total, among which 14 features were categorical and 11 were numeric. A total of 400 instances were stored in this dataset, which was collected from the Apollo Hospitals, Karaikudi, Tamilnadu, India [71,72]. The features include age, specific gravity, blood pressure, albumin, sugar, red blood cells, pus cell clumps, pus cells, bacteria, sodium, blood glucose random, blood urea, serum creatinine, potassium, hemoglobin, packed cell volume, red blood cell count, white blood cell count, hypertension, diabetes mellitus, coronary artery disease, appetite, pedal edema, and anemia [73]. The output variable has only two values, “ckd” and “notckd”, where ”ckd” denotes chronic kidney disease. Every attribute contained missing values except the class attribute. Among the 400 instances, there were 250 instances with the “ckd” class, which is 62.5% of the aggregate data, and the remaining 150 instances were labeled as the “notckd” class, which is 37.5% of the whole data.

3.6 Experimental Results

In this section, we describe the experiments conducted with synthetic and real-world datasets to validate the proposed method. Specifically, we evaluated the impact of the TLPCM and the active learning technique on the performance of the constructed models. The Rand index is used for performance evaluation, and the experimental setup is first described. Then, the performance of the proposed algorithms on synthetic and real-world datasets is reported and discussed.

3.6.1 Experimental Results on the Synthetic Datasets

3.6.1.1 Increase the Number of Data Samples

In the first set of experimental studies, we determine the effect of increasing the number of data samples on both the PCM and TLPCM algorithms. The S_1 dataset is used by increasing the number of data samples in the target dataset gradually and applying both the PCM and TLPCM algorithms each time until the optimum clustering for our data is reached. Fig. 3.10 shows Target data with final cluster centers found with TLPCM for Experiment with five samples on target data. Fig. 3.11 shows that TLPCM reached a Rand index of approximately one when it had 4 data points per cluster. However, PCM needed 14 data points to cluster the data correctly, which is three times more than what TLPCM needed.

3.6.1.2 Shift the Location of the Clusters

The second experiment studies the effect of moving the centers of the S_1 target dataset in Fig. 3.2 apart and then moving them closer, as shown in Fig. 3.12. To move the centers precisely, we calculated the main center of the three centers. Next, we calculated the slope and the angle between the main center and the center of each

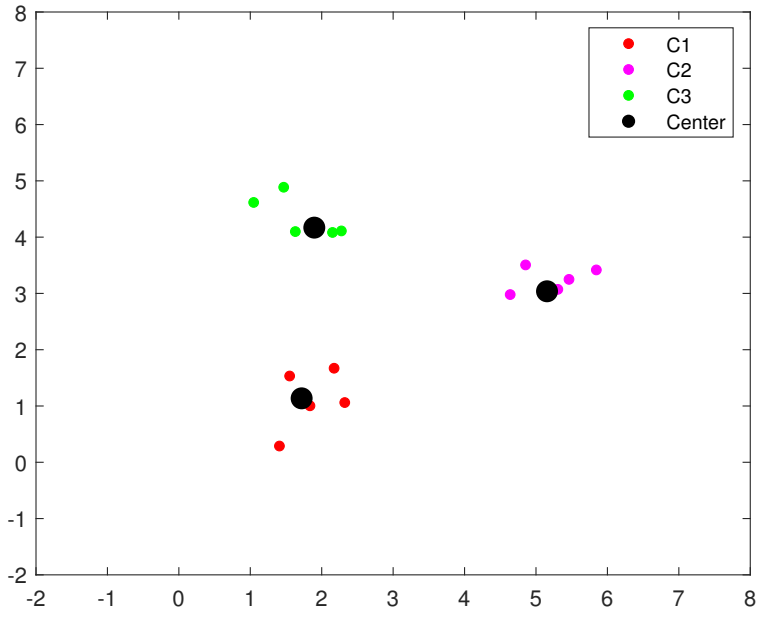


Figure 3.10: Target data with final cluster centers found with TLPCM for Experiment.

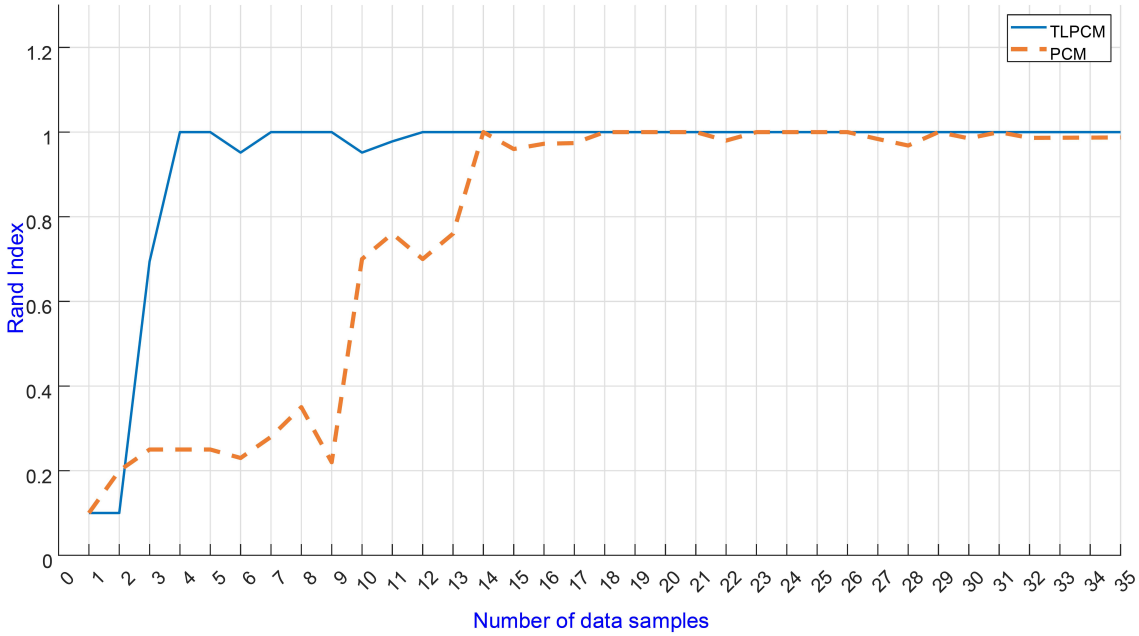
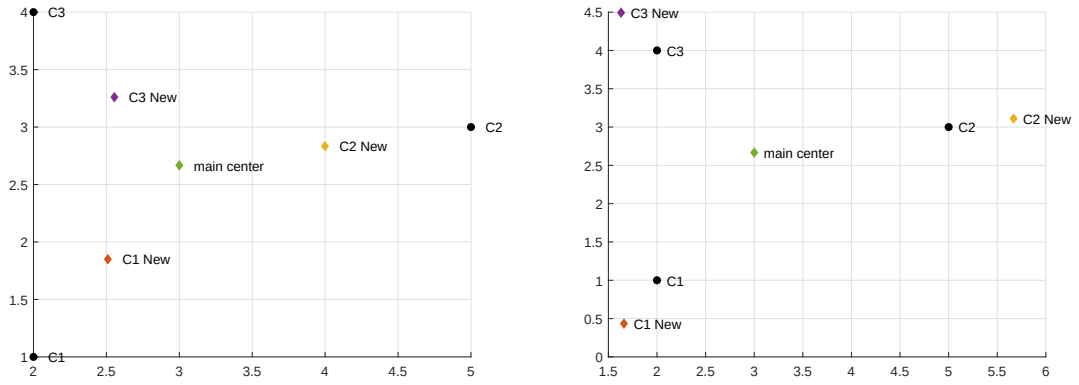


Figure 3.11: Rand index versus the number of data samples per cluster using PCM and TLPCM; our algorithm reached a Rand index of approximately one when it had 4 data points per cluster.



(a) (b)

Figure 3.12: Moving centers. (a) Moved centers inside (converged) and (b) moved centers outside (diverged)

cluster. The distances between the initial three centers were then calculated to obtain the percentage distance needed to move the centers to the new locations. Finally, we calculated the centers' new locations and generated a Gaussian dataset based on the slope, angle, and distances determined above. The general form of these steps is described in Algorithm 2.

The proposed TLPCM algorithm and the original PCM algorithm were applied to the S_1 dataset, with 14 samples per cluster in the target data at each cluster location, starting from very close clusters, which overlapped each other with 50 percent of the original distance between the centers. Fig. 3.13 shows that both algorithms failed to cluster the data because the clusters overlapped. However, when we increased the distance between the clusters, TLPCM was able to cluster the data when the distance between clusters was 25% less than the distances between clusters in the original dataset.

Algorithm 2: Converging and Diverging Centers

Input: V= target centers, d= Euclidean distance, MD = movement direction

Output: x_new,y_new= x and y positions for the new center location.

begin
  $Load(V)$
  $main\_center \Leftarrow mean(V)$
  foreach $v_j \in V$ **do**
    calculate the slope and the angle between $main\_center$ and v_j.
    calculate the shifting distance (d).
    if $MD==1$ **then**
      $x\_new = v_j + d * cos(\phi_i)$
      $y\_new = v_j + d * sin(\phi_i)$
    else
      $x\_new = v_j - d * cos(\phi_i)$
      $y\_new = v_j - d * sin(\phi_i)$
    end
  end
end

3.6.1.3 Coincident Clustering Feature

Coincident clustering is one of the PCM algorithm features since it relaxes the probabilistic constraint in the fuzzy c-means (FCM) clustering algorithm. Hence, an experiment is created here to test our new approach with this feature, as shown in Fig. 3.14. Thus, the TLPCM algorithm is initialized "mistakenly" with 4 clusters for the target data. However, the real data set S_1 has 3 clusters only in the source and target data.

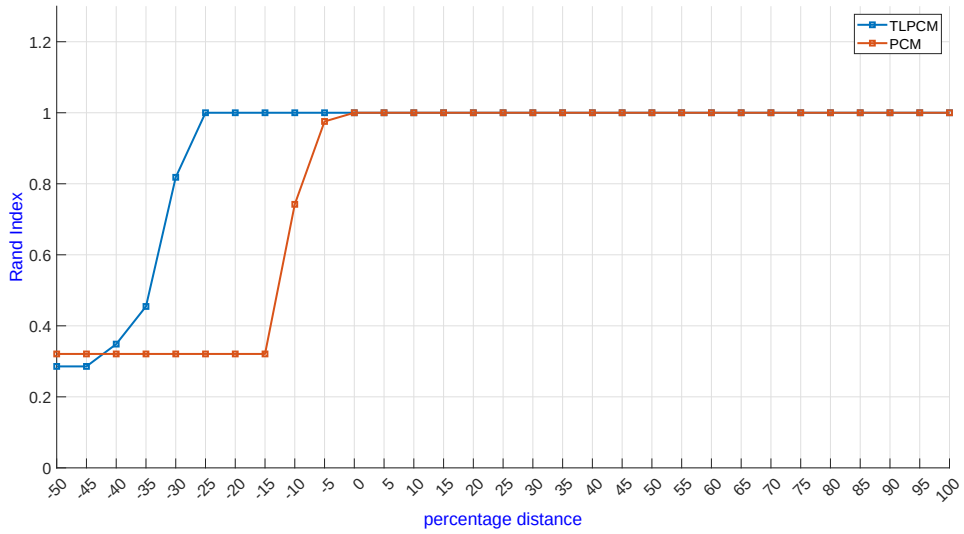


Figure 3.13: Rand index versus percentage distance using PCM and TLPCM; our approach, TLPCM, was able to cluster the data when the distance between clusters was 25% less than the distances between clusters in the original dataset.

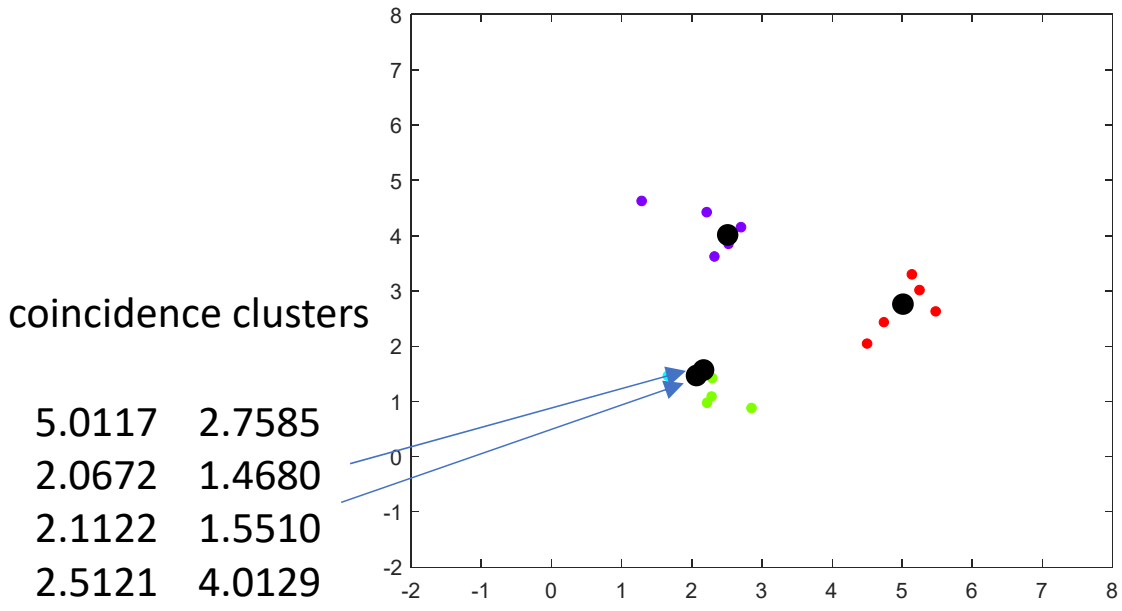


Figure 3.14: Experiment of testing the TLPCM algorithm with coincidence clustering feature..

3.6.1.4 Different Initializations

PCM might not guarantee a unique clustering result if the partition matrix and initial cluster number are chosen randomly. For this reason, the results of the clustering are

not consistent. Concerning cluster consistency, the TLPCM algorithm was compared with the original PCM algorithm by applying different initializations of FCM on theS_2 dataset on both algorithms. Fig. 3.15 shown target data with 6 small clusters and final cluster centers found with TLPCM. Table 3.3 and Fig. 3.16 show that TLPCM surpassed PCM in terms of consistency and accuracy. The average Rand index of PCM is 0.7304 with a standard deviation of 0.0514, while the average Rand index of TLPCM is 0.977 with a standard deviation of 0.0245. A smaller standard deviation means greater consistency, quality, and predictability.

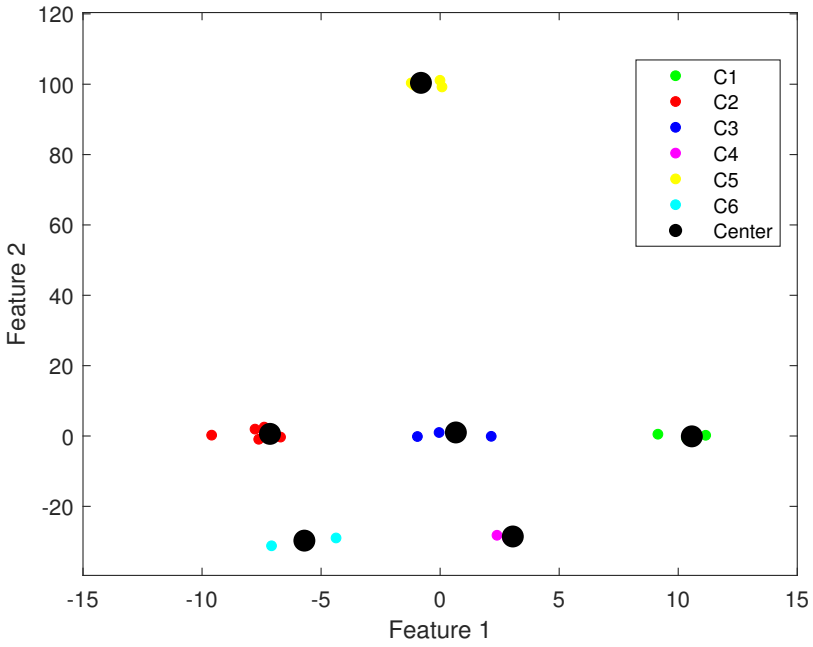


Figure 3.15: Target data with 6 small clusters and final cluster centers found with TLPCM.

Table 3.3: Consistency of TLPCM and PCM.

TLPCM	**PCM**
Mean = 0.9778	Mean = 0.7304
STD = 0.0245	STD = 0.0514

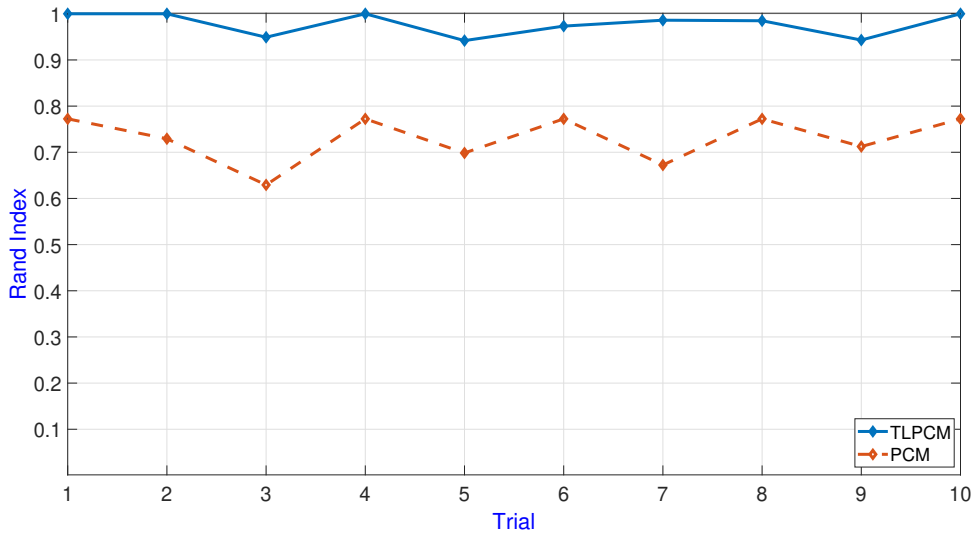


Figure 3.16: Rand index with different initializations for PCM and TLPCM; our algorithm has a higher average Rand index and a smaller standard deviation than PCM, which means greater consistency.

3.6.1.5 Increase the Number of Cluster Centers

This experiment studies the effect of increasing the number of clusters on our proposed TLPCM algorithm, comparing the original PCM algorithm results. TheS_2 dataset, as shown in Fig. 3.4, is used to carry out this experiment. We started our experiment by making the number of clusters in the target dataset less than that in the source dataset. Then, we increased the number of clusters until we reached 10 clusters in the target dataset.

This was greater than the number of clusters in the source dataset, which had four clusters. Fig. 3.17 shows that PCM could not cluster the data when we set the number of clusters at 10. In fact, we had only two clusters in the very first trial. The accuracy of PCM increased as the actual number of clusters increased and continued to do so until we reached our preset number of clusters, which was 10. Nevertheless, TLPCM could cluster the data, because we inherited the information from a similar source dataset but not the same target dataset. In this experiment, we attempted to fix the initialization and changed the actual number of clusters in the

target data; then, we observed how both the PCM and TLPCM methods performed. The TLPCM was most accurate when the source and the target data had the same number of clusters (four clusters).

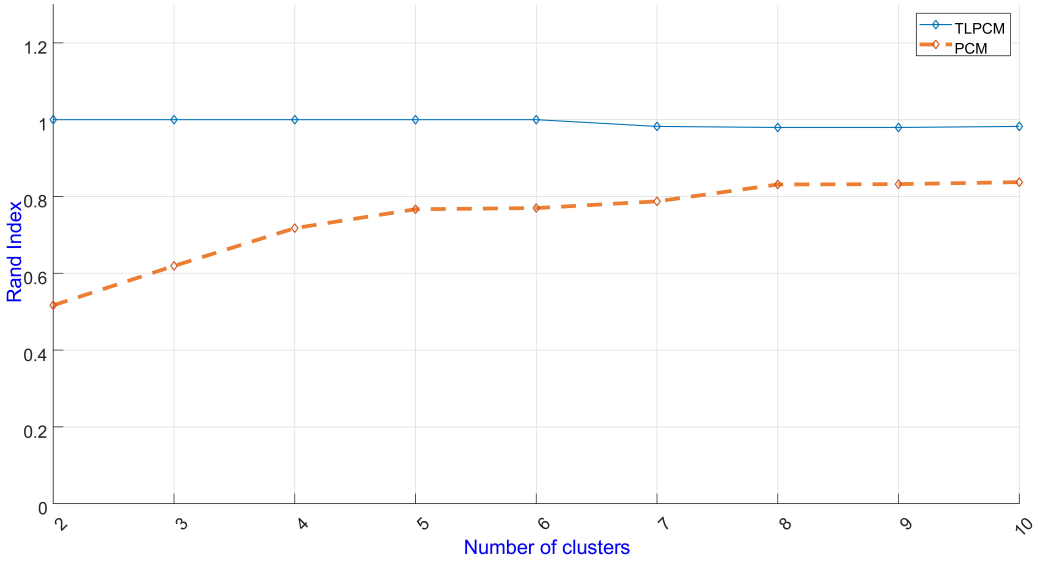


Figure 3.17: Rand index versus the number of clusters using PCM and TLPCM; TLPCM was able to cluster the data regardless of the number of clusters that had been set.

3.6.1.6 Experiments on η_j and β_j Parameters

I provide here a few experiments to help understand the update of both η_j and β_j. In each experiment, we initialized each η_j and β_j to a value of 2 and allowed equations 3.11 and 3.12 to update their values.

Because the cluster data and hence cluster center locations in the source and target are consistent, there is not much change that occurs in the evolution of η_j and β_j, and as shown in Fig 3.18 and Fig. 3.19 respectively.

To show how the adaptation of η_j and β_j for a target data set with more clusters than existed in the source, consider the source from data set S_2 . From Fig. 3.20, we see that the η_j values are stable after a few iterations at fairly small values. This is due to the fact that η_j controls the local cluster spread. The clusters in Fig. 3.4 are well

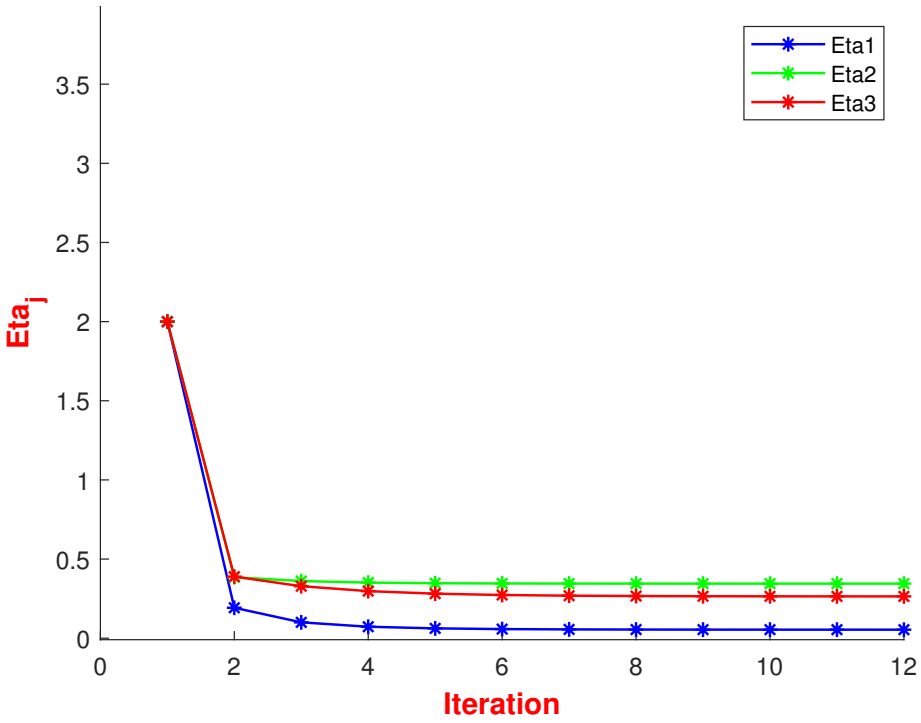


Figure 3.18: Evolution of η_j vs iteration on target data for all clusters using S_1 dataset.

separated and compact, so the points themselves are close to their respective cluster centers, keeping η_j small. The evolution of the values for each β_j are also stable, but the range is more varied. This is because a few of the source cluster centers need to "stretch" into the target space to influence the extra structure there. Besides, it can be seen from equation 3.12 that β_j is proportional to the average fuzzy distance between the source and target cluster centers. The target cluster with a higher β_j value will have greater mobility because it achieves more source centers and therefore moves as the iteration proceeds. This explains why β_5 is significant compared to the β_j of other clusters (see Fig. 3.21). The same idea can be seen for η_j in Fig. 3.20, but with smaller range because η_j is working with the local target cluster spread (see equation 3.11).

As a final observation, what happens when the target clusters are quite different

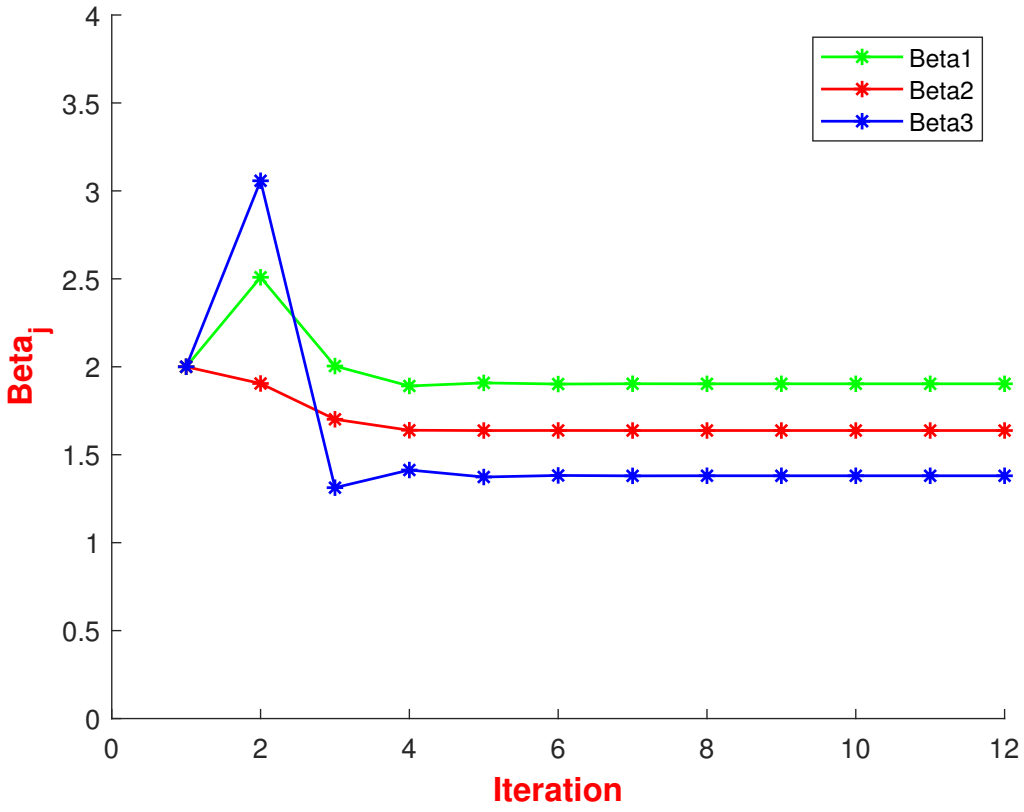


Figure 3.19: Evolution of β_j vs iteration on target data for all clusters using S_1 dataset.

from those in the source? Consider the source and target data in Fig. 3.5. Note the difference in both location and shape between these clusters. As a result, even though relatively stable (see Fig. 3.23 and Fig. 3.24), there is a larger variance in both of the parameter sets with η_j controlling the within cluster spreads, and β_j controlling the match between source and target cluster centers. Fig. 3.22 shows the target data with discovered cluster centers using TLPCM.

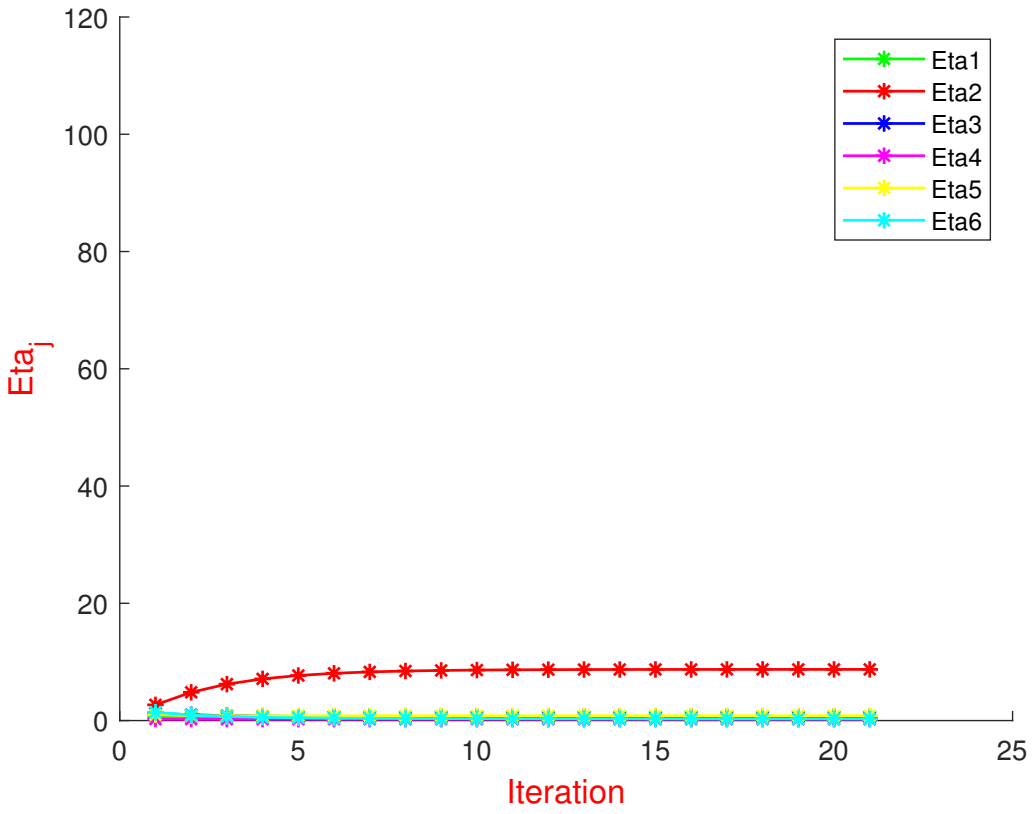


Figure 3.20: Evolution of η_j vs iteration on target data for all clusters using S_2 dataset.

3.6.2 Experimental Results on the Real-World Datasets

3.6.2.1 Experimental Results on the Bed Sensor Dataset

To validate our approach, we used data from three different TigerPlace residents with different numbers of normal and abnormal days, as shown in Table 3.2. We conducted three experiments using the resident data to test the performance of our algorithms for clustering on both normal and abnormal days. Conventional mathematical models (such as classifiers) for early illness recognition are not transferable from one person to another due to different disease-behavior associations that can vary among people. However, our transfer-learning possibilistic C-means approach allowed us to take advantage of one person's data to cluster a few weeks of data from another person's

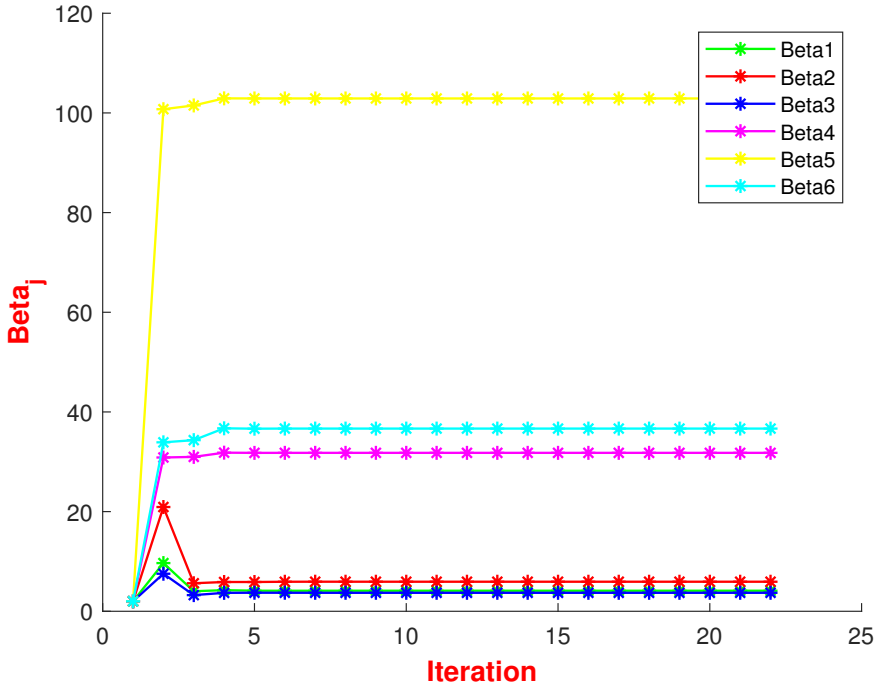


Figure 3.21: Evolution of β_j vs iteration on target data for all clusters using S_{21} dataset.

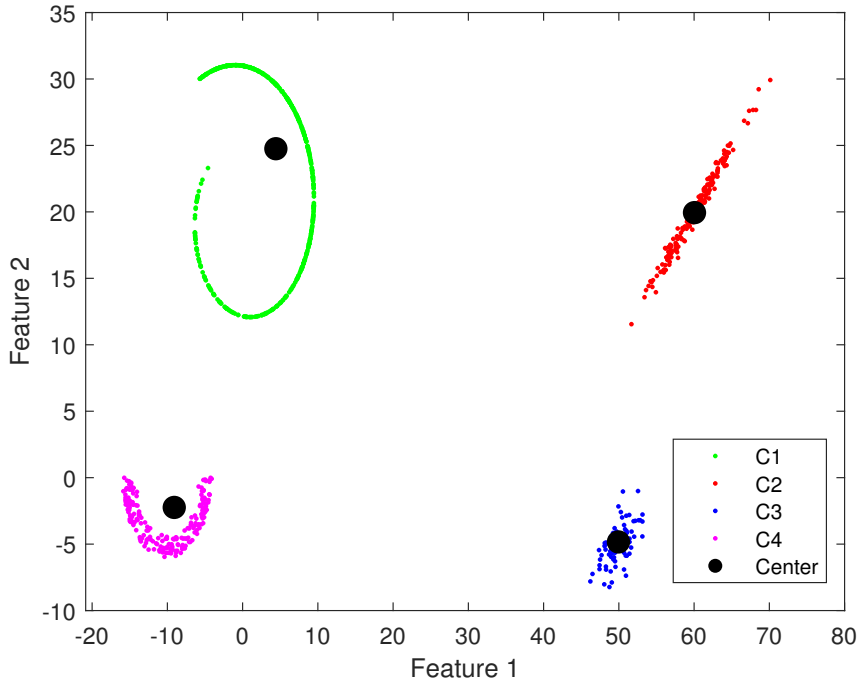


Figure 3.22: Target data S_3; discovered cluster centers with TLPCM .

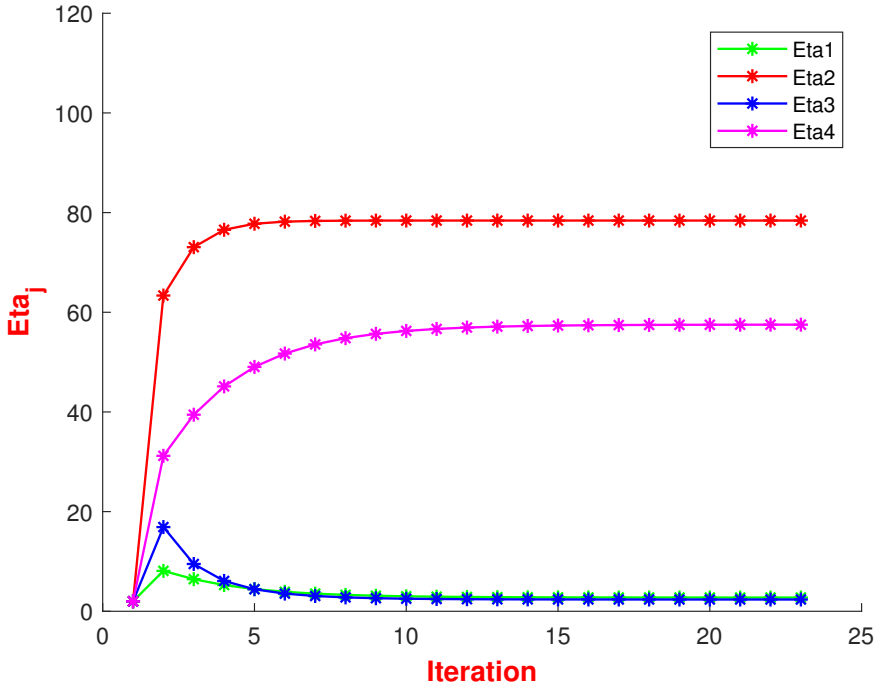


Figure 3.23: η_j vs iteration on target data for all clusters using S_3 dataset.

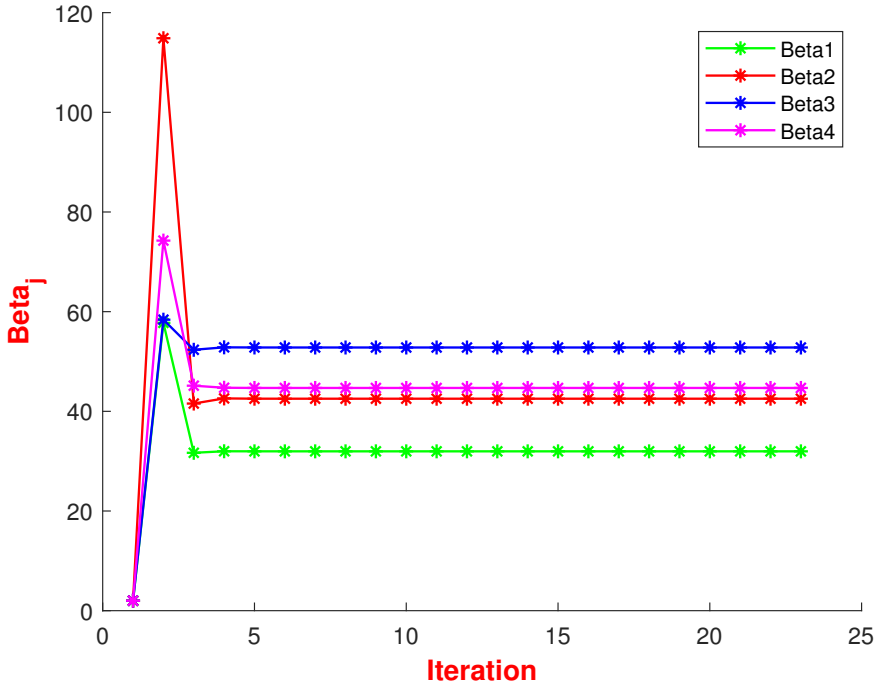


Figure 3.24: β_j vs iteration on target data for all clusters using S_3 dataset.

data.

To confirm our labels for normal and abnormal days, we ran the original PCM algorithm on each resident's data 1000 times and then averaged the partition matrices and compared the predicted labels with the nurse's notes. Once we had our ground truth, we tested our approach and calculated the Rand index. We used Resident 1 as the source data for our algorithm and Resident 2 as the target data. We fed our proposed algorithm with target data weekly and calculated the Rand index each time.

Fig. 3.25 shows that TLPCM could correctly cluster the new resident data after 10 weeks. However, PCM could not cluster the data of Resident 2 until we had 17 weeks of data. The same experiment was performed with the data of the other residents by considering Resident 1 as the source data and Resident 3 as the target data; then, we considered Resident 2 as the source data and Resident 3 as the target data, as shown in Fig. 3.26 and 3.27, respectively. Clearly, Fig. 3.27 has the best Rand index curve for TLPCM compared with the other two figures. The reason is that Resident 1 is the youngest and healthiest of the three. Furthermore, Resident 1 does not use a walker, which makes his routines different on normal and abnormal days. On the other hand, Resident 2 and Resident 3 use walkers that affect their walking. Since Residents 2 and 3 have similar activities, TLPCM was able to cluster target data in fewer weeks (7 weeks), as shown in Fig. 3.27. However, PCM used directly on Resident 3 could not stabilize on the desired cluster structure until approximately 21 weeks of data.

Another experiment was performed using the same real-world dataset in case we needed the ground truth—whether the resident would have a normal or abnormal day. Instead of computing the ground truth by applying the original PCM algorithm 1000 times, the following steps were used: As shown in Fig 3.28, in the first step, we considered the predicted labels of the first week as the ground truth of the first week in the data for the second step; then, we considered the predicted labels of the two weeks in the second step as the ground truth of the third step's first two weeks of data,

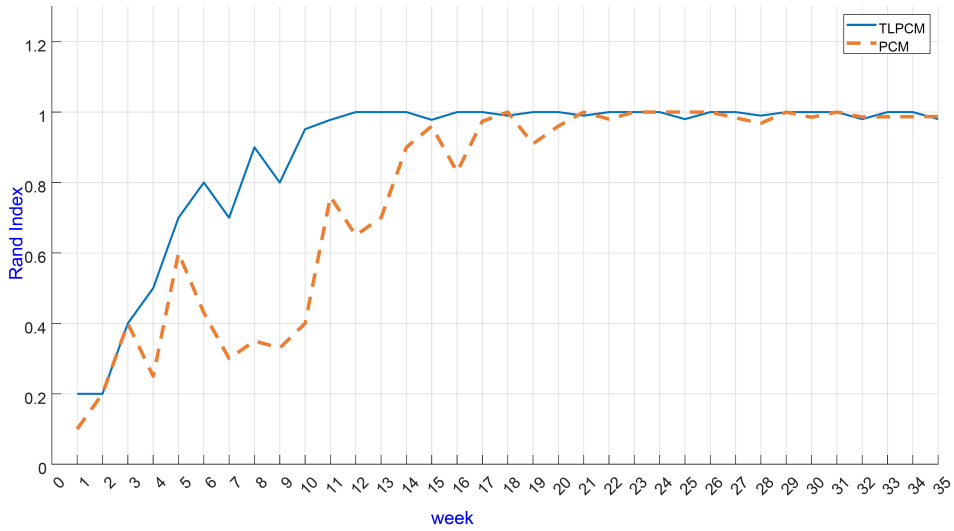


Figure 3.25: Weekly Rand index using Resident 1 as the source data and Resident 2 as the target data; TLPCM was able to reach a Rand index of approximately one with fewer weeks of data.

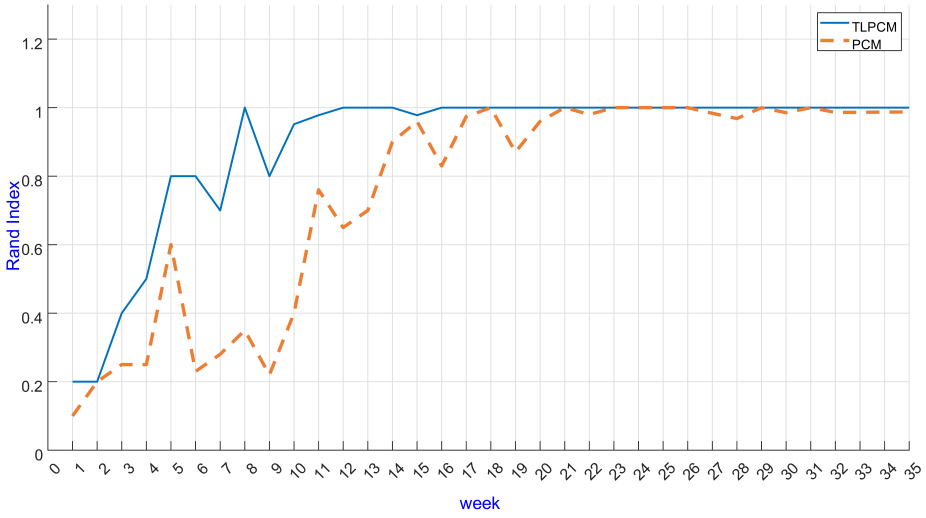


Figure 3.26: Weekly Rand index using Resident 1 as the source data and Resident 3 as the target data; TLPCM was able to reach a Rand index of approximately one with fewer weeks of data.

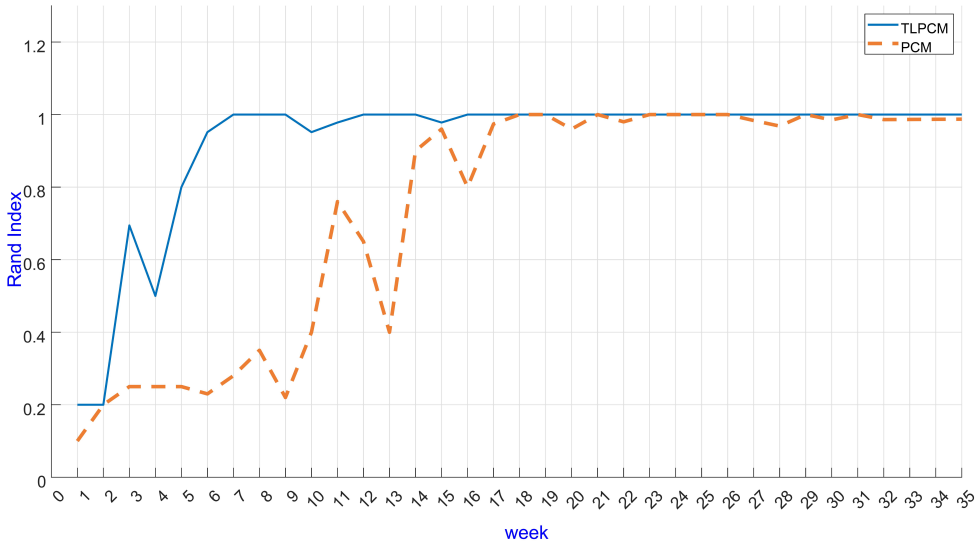


Figure 3.27: Weekly Rand index using Resident 2 as the source data and Resident 3 as the target data; TLPCM needs fewer data than in the previous two cases due to the similarity between Residents 2 and 3 (source and target data).

and so on. Using our algorithm, residents are consistently checked on a weekly basis. If there is an increase in cluster numbers, this is abnormal and could be indicative of a potential health issue, as shown in Fig. 3.29.

All of the above experiments on the bed sensor data demonstrate how to use our proposed algorithm to transfer the knowledge gained from one resident to another. In the next three experiments, we tested our algorithm by combining the data of two residents as the source data and then using the third resident as the target data. Two of the residents (Residents 2 and 3) use a walker, and they are close in age, while the third (Resident 1) is healthier and younger and does not use a walker. We took different combinations of these three residents as the source and target data and checked the effect of combining different sources of data on the clustering results of the target data. We combined the data of Residents 1 and 2 to form our source data and used resident 3 as the target data. In other words, we combined healthy and non-healthy residents for the source data and used a non-healthy resident as target data. We started feeding our proposed model with the target data weekly, as we did

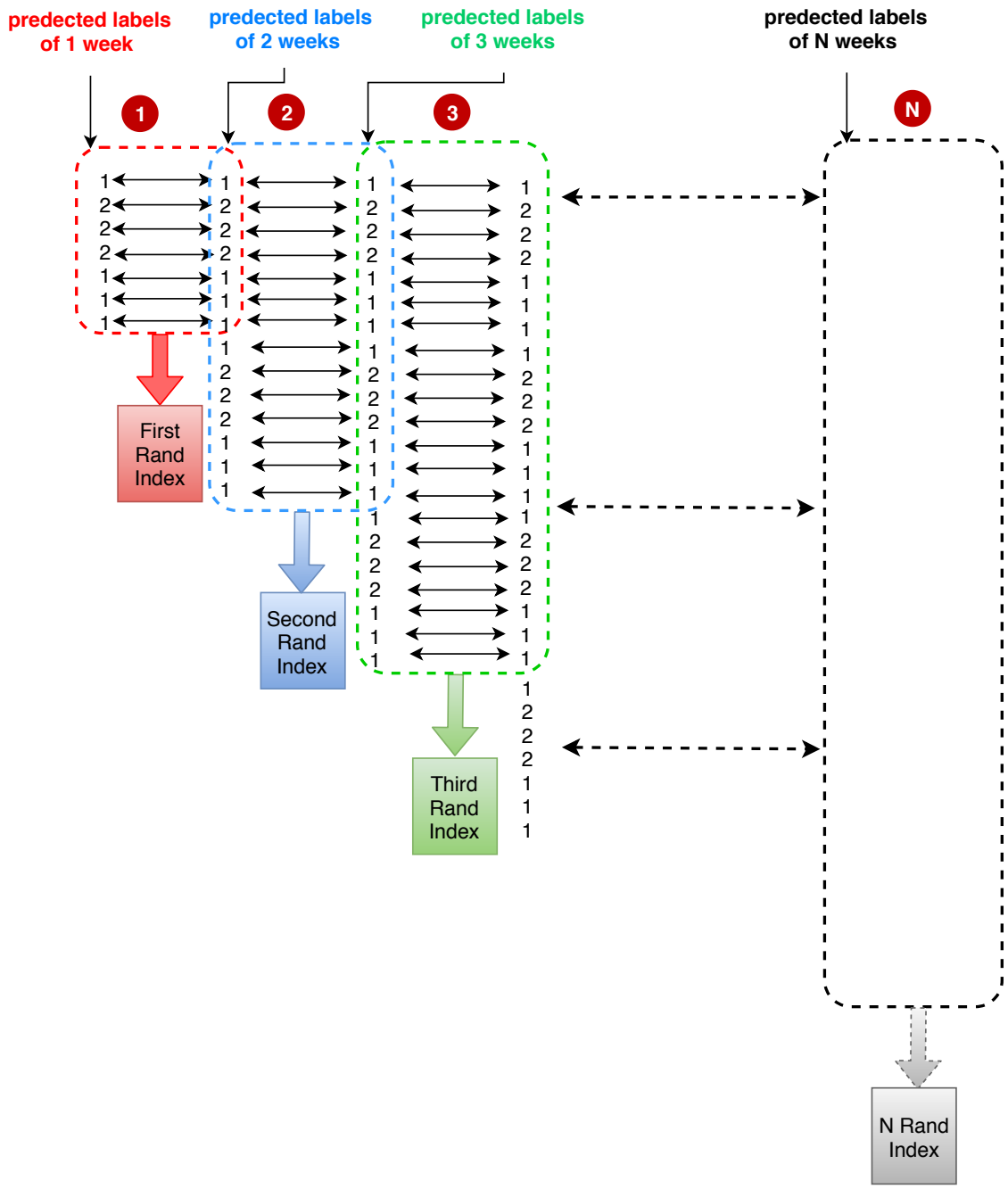


Figure 3.28: The block diagram illustrates how to detect a new health issue using the ground truth from earlier weeks' data.

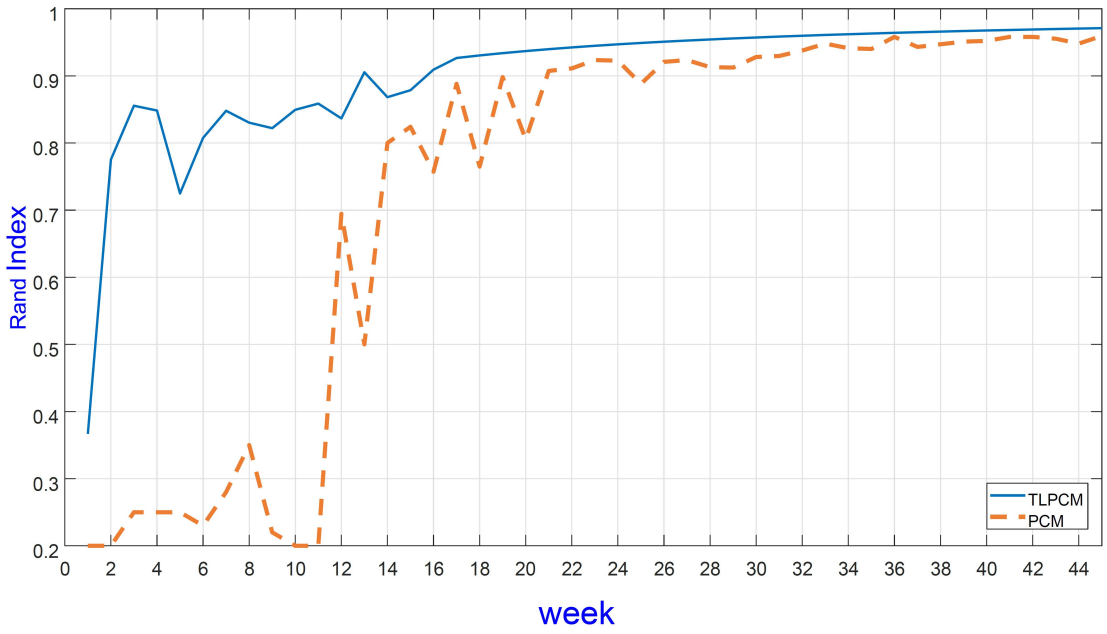


Figure 3.29: Detecting a new health issue by consistently checking on a weekly basis using our algorithm with a small number of data samples compared to those in the PCM algorithm.

in the previous experiments. Then, we calculated the Rand index each time. Fig. 3.30 (the solid blue curve) shows that TLPCM could correctly cluster the data in approximately eight weeks of target data. The results are compared with those of the other two experiments, as shown in Fig. 3.30 (the green dashed and red dotted curves), where the source data include only one of the residents (either Resident 1 or 2) and we kept the target data as it was (Resident 3). When we combined the healthy and non-healthy residents in the source data and used a non-healthy resident as the target data, the results were better than they were when we had only healthy-resident source data, because approximately half of the source data were similar to the target data. However, the results were less accurate than when all of the source data were similar to the target data, as shown in Fig. 3.30.

The same experiment was applied to the other two cases: when Residents 1 and 3 provided the source data and Resident 2 provided the target data, and then when Residents 2 and 3 provided the source data and Resident 1 provided the target data.

Fig. 3.31 and Fig. 3.32 show the results of the second and third cases, respectively. We can conclude that adding more source data similar to the target data will improve the clustering results, because the transfer-learning technique depends on the historical information that we obtain from the source data. Moreover, adding source data different from the target data will reduce the efficiency of the algorithm in clustering the target data.

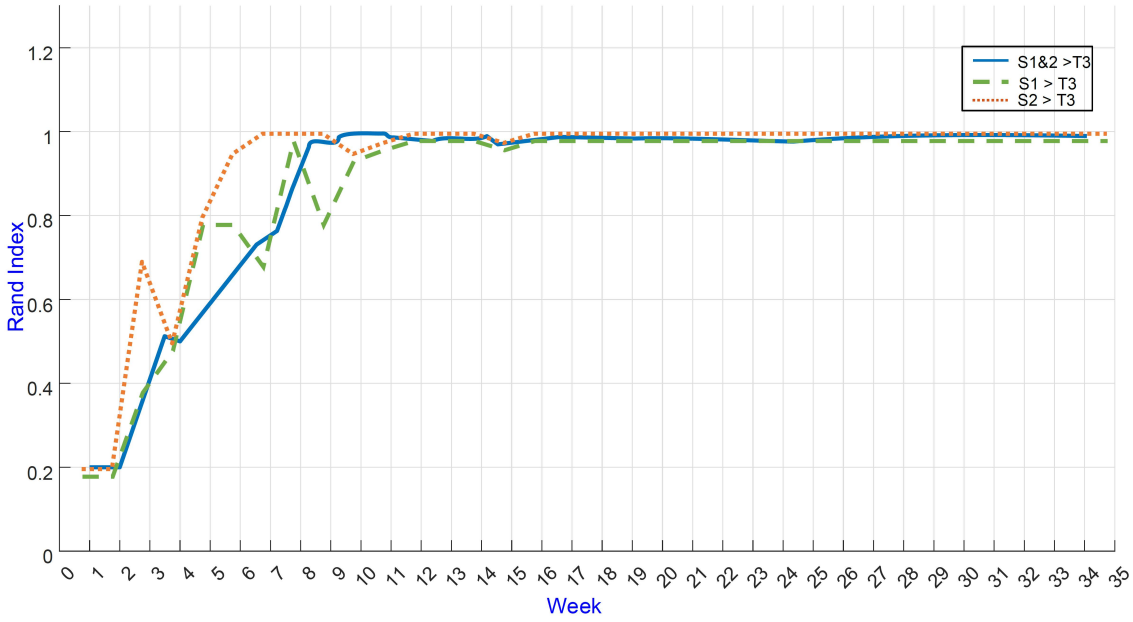


Figure 3.30: The weekly Rand index for TLPCM using Residents 1 and 2 as the source data and Resident 3 as the target data (solid blue curve) compared with the Rand index for a single resident (Resident 1 or 2) as the source data and a single resident (Resident 3) as the target data (green dashed and red dotted curves).

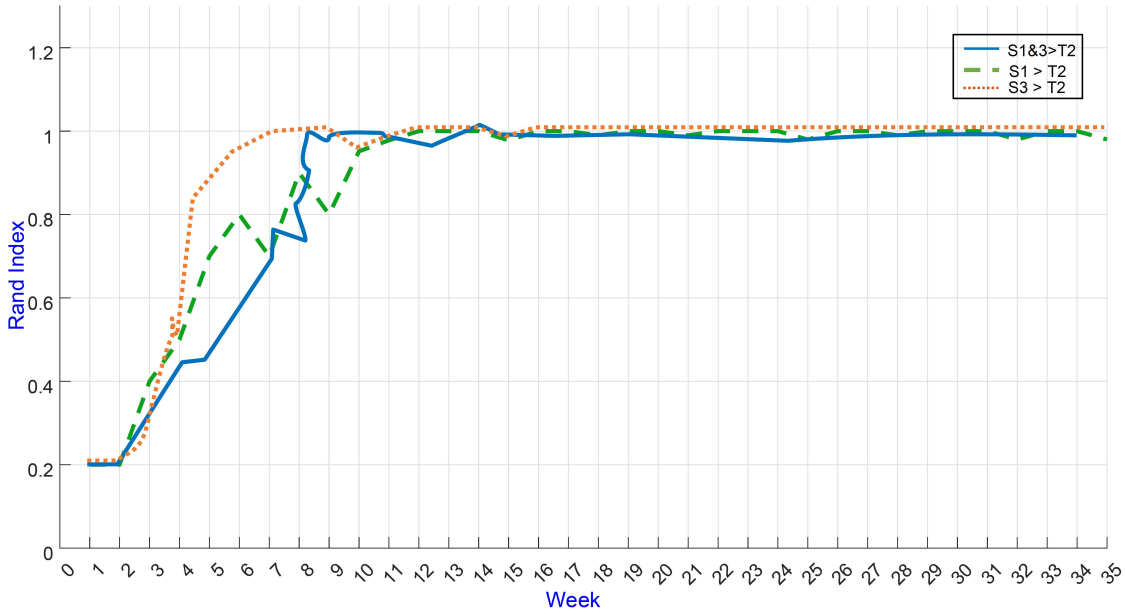


Figure 3.31: The weekly Rand index for TLPCM using Residents 1 and 3 as the source data and Resident 2 as the target data (solid blue curve) compared with the Rand index for a single resident (Resident 1 or 3) as the source data and a single resident (Resident 2) as the target data (green dashed and red dotted curves).

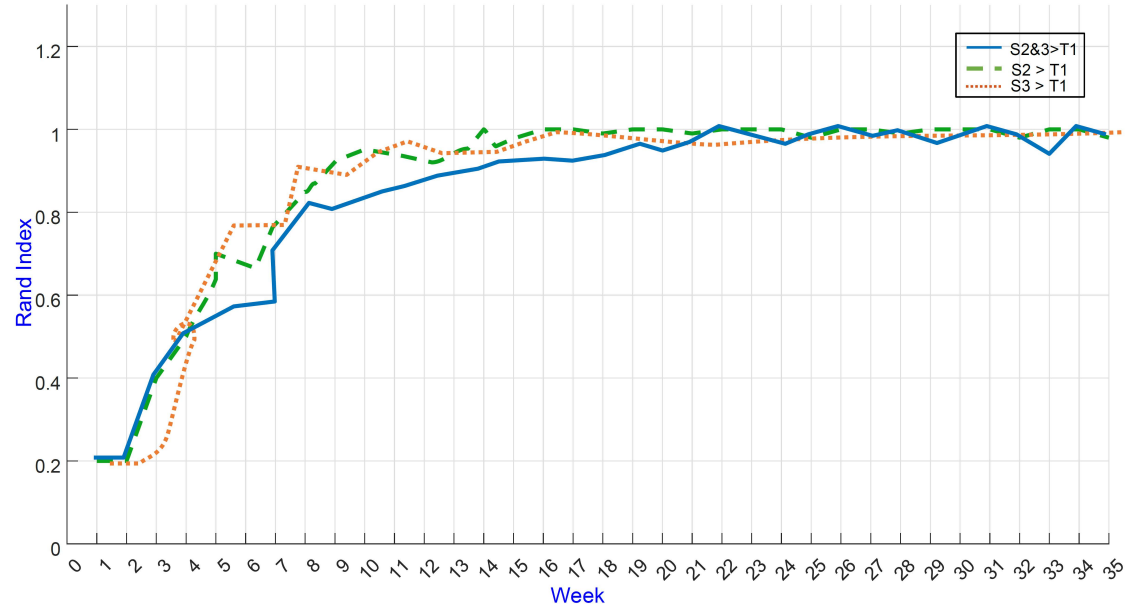


Figure 3.32: The weekly Rand index for TLPCM using Residents 2 and 3 as the source data and Resident 1 as the target data (solid blue curve) compared with the Rand index for a single resident (Resident 2 or 3) as the source data and a single resident (Resident 1) as the target data (green dashed and red dotted curves).

3.6.2.2 Experimental Results on the Forensic Glass Dataset

This experiment analyzes our algorithm's validity in finding the right number of clusters in unbalanced data and compares the original PCM algorithm results. These data, described in Section 3.5.2.2, are unbalanced, as shown in Fig. 3.9, and it will not be easy to find the right number of clusters with a few samples of data. First, I examine a heatmap of the correlations, as shown in Fig. 3.33. It seems to be a strong positive correlation between RI and Ca. This could be a hint to perform principal component analysis (PCA) [74] in order to decorrelate some of the input features. Before applying PCA, I plot the cumulative explained variance, as shown in Fig. 3.34, and it appears that about 99 % of the variance can be explained with the first five principal components.

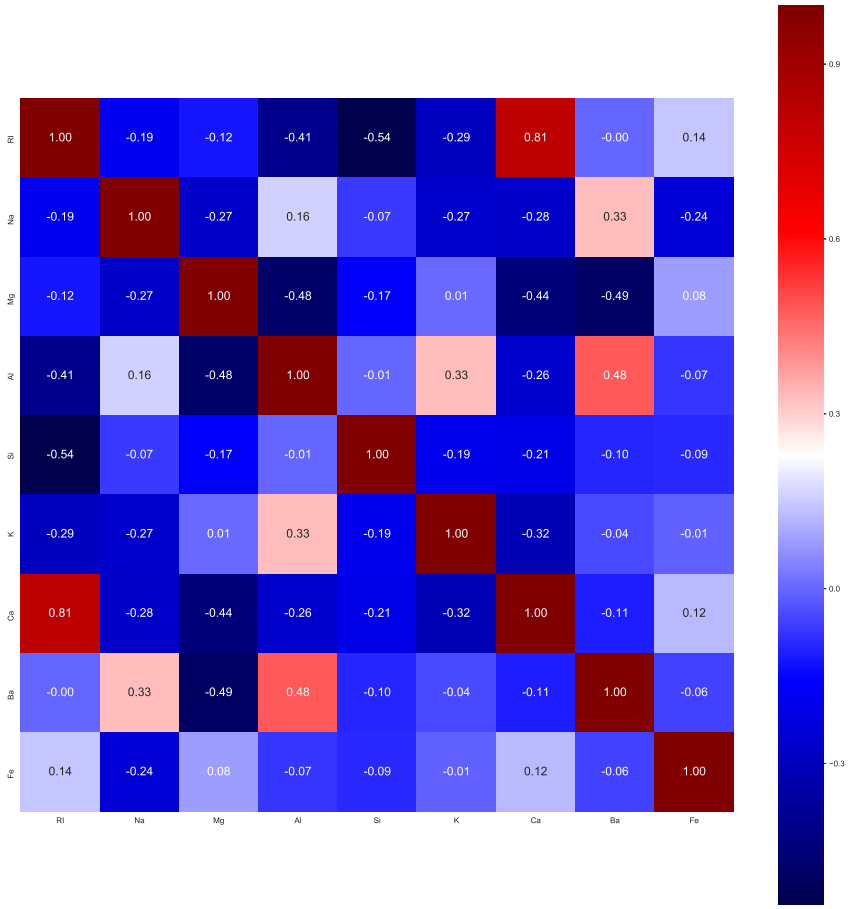


Figure 3.33: Heatmap of the correlations glass dataset.

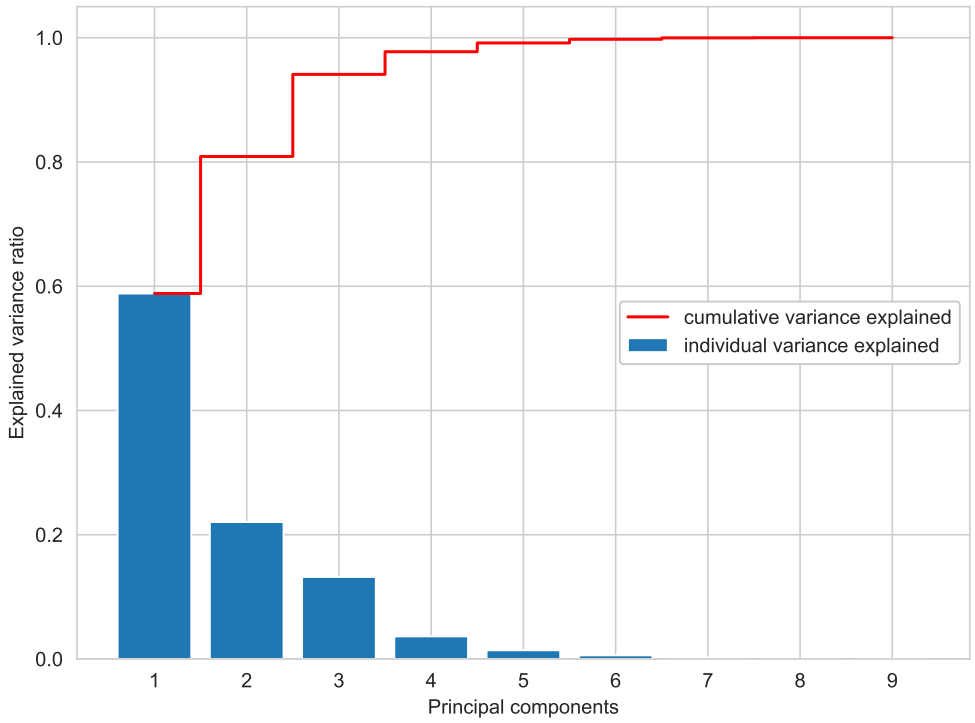


Figure 3.34: Principal component analysis performance for glass dataset features.

Therefore, after normalizing the data by utilizing the standard score normalization technique, the PCA algorithm is used to reduce the number of variables in our data by extracting the essential variables. Half of the data samples are used as a source domain to obtain the cluster centers (knowledge), and then the target data are fed to the PCM and TLPCM models gradually by adding 20% of the total target data each time. As shown in Table 3.4, there is consistency in the number of clusters in TLPCM due to the knowledge inherited from the source data, while PCM is tricked into finding the right number of clusters.

Table 3.4: Glass dataset Results.

% Target data	Ground truth	PCM NO. of Clusters	PCM Rand Index	TLPCM NO. of Clusters	TLPCM Rand Index
20%	6	3	0.56	6	0.76
40 %	6	2	0.58	6	0.78
60%	6	4	0.60	6	0.80
80%	6	5	0.64	6	0.80
100%	6	5	0.763	6	0.808

3.6.2.3 Experimental Results on the Chronic Kidney Disease Dataset

The kidney dataset, described in Section 3.5.2.3, is collected from 400 patients (250 "CKD", 150 "notckd"). Every individual person has certain features that differ from the others, which makes this a good test for the transfer-learning technique in our algorithm. Moreover, this dataset contains many missing values and categorical values that need to be preprocessed. Thus, before utilizing the dataset in our proposed algorithm, the data are first cleaned. Then, I plot the Box-plot as shown in Fig. 3.35 to see if all the data features are on the same scale.

However, the features are inconsistent in terms of the range of the values (*i.e.*, they have various scales). Thus, the features are then normalized, as shown in Fig. 3.36. Next, I plot the cumulative explained variance, as shown in Fig. 3.37, and it appears that variance can be explained with almost all principal components. Hence, the PCA algorithm is not applied to this data. After the data carpentry phase, half of the data samples are used as a source domain to determine the cluster centers (knowledge).

The proposed algorithm (TLPCM) is then tested with various sizes of the target dataset and compared with the PCM algorithm (see Table 3.5). First, both PCM and TLPCM are fed 20% of the target dataset. The Rand index was 0.607 and 0.78,

respectively. It can be seen from the Rand index that TLPCM was able to achieve a better performance than PCM. Next, we steadily increased the target dataset's size by 20% of the actual data size until the full size of the target dataset was reached. As expected, increasing the size of the target dataset improved the performance of the algorithm. With the full target dataset, the Rand index was 0.802 and 0.906 for PCM and TLPCM, respectively. It is worth noting that TLPCM could significantly better cluster results than PCM with even a small portion of the dataset. Thus, the core principle of transfer learning is satisfied since our algorithm does not need an abundance of data to cluster correctly.

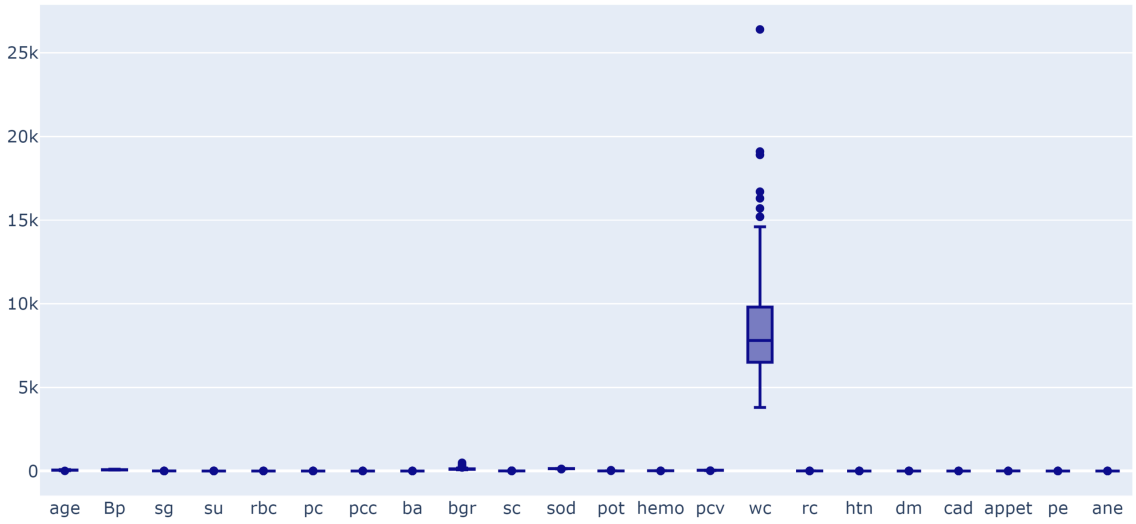


Figure 3.35: Kidney disease dataset features before normalization.

Table 3.5: Kidney dataset Results.

Target data %	**Ground truth**	**PCM NO. of Clusters**	**PCM Rand Index**	**TLPCM NO. of Clusters**	**TLPCM Rand Index**
20%	2	1	0.607	2	0.78
40%	2	2	0.605	2	0.80
60%	2	1	0.708	2	0.89
80%	2	2	0.802	2	0.872
100%	2	2	0.852	2	0.906

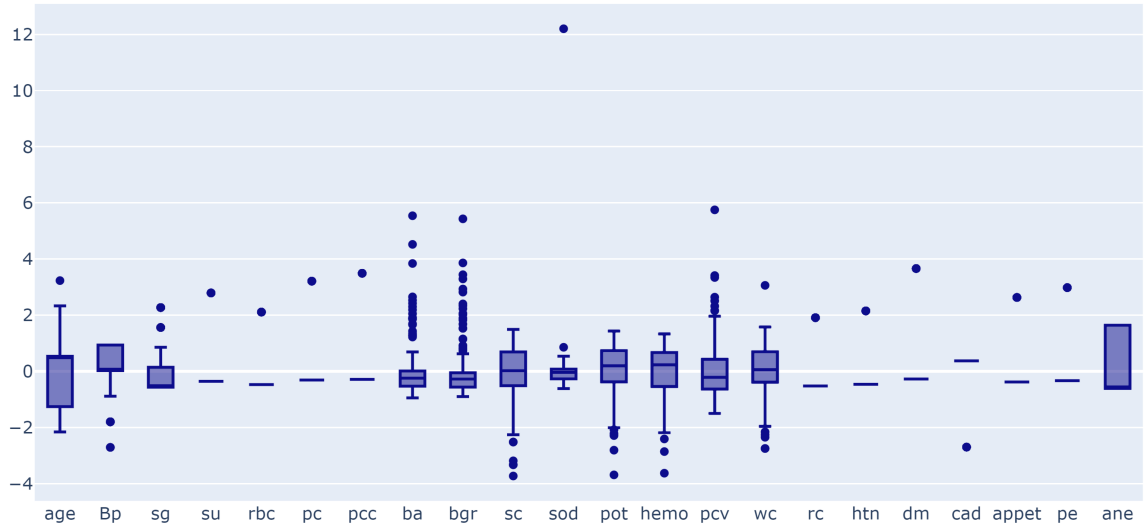


Figure 3.36: Kidney disease dataset features after normalization.

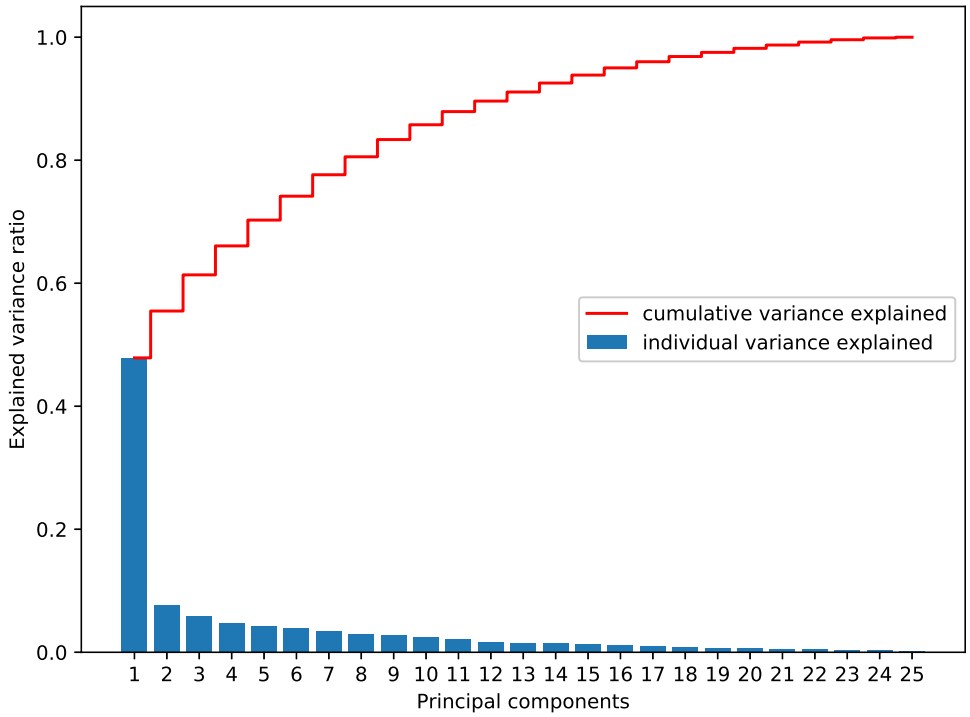


Figure 3.37: Principal component analysis performance for Kidney disease features.

3.7 Conclusion

Based on the original PCM algorithm, and inspired by transfer learning, we designed a novel clustering framework called the transfer-learning possibilistic c-means (TLPCM) clustering algorithm. TLPCM works in applications where data is limited and insufficient for useful clustering or is polluted by unknown noise or outliers. When the experimental results are compared with those of the original PCM algorithm, we can see the attractiveness and efficacy of the proposed TLPCM algorithm in both artificial and real transfer scenarios.

The real-world data showed that for early illness recognition and analyzing the Forensic Glass and Chronic Kidney datasets, the proposed method outperformed individual learning in terms of consistency and the amount of data needed for clustering. As more information was added to the algorithm, the performance steadily increased, but only up to a certain limit. Once that limit was reached, performance plateaued. Since data labeling is difficult in a real-world setting such as TigerPlace, transfer learning can help train illness recognition classifiers across residents. Early signs of illness could be predicted in elderly residents of TigerPlace based on unobtrusive monitoring sensors through the use of transfer learning in TLPCM. Further study based on this research will focus on transferring membership values from the source domain instead of cluster prototypes.

Chapter 4

Non-Invasive Classification of Sleep Stages with a Hydraulic Bed Sensor Using Deep Learning

4.1 Introduction

Sleep is a critical physiological phenomenon for recovery from mental and physical fatigue. Lately, there has been much interest in the quality of sleep, and research is actively underway. In particular, it is vital to have a repetitive and regular sleep cycle for good sleep. Nevertheless, it takes much time to determine sleep stages using physiological signals by experts. A person with a sleep disorder such as apnea will stop breathing for a while throughout sleep. If it regularly happens, sleep disorders can be dangerous for health. An early step in diagnosing these disorders is the classification of sleep stages [75, 76].

Unfortunately, sleep disorders have been affecting many people around the world in different ways. Whatever the cause of these disorders, the consequences can be severe. The quality of sleep depends on the number and order of these stages. The names of these stages, are Wake, Non-REM1, Non-REM2, Non-REM3, and REM.

Detection of any sleep disorder, such as sleep apnea, insomnia, or narcolepsy, requires a correct staging of sleep [77, 78].

Classification of sleep stages is also essential for managing the quality of sleep. Sleep studies depend on manual scoring of sleep stages from raw polysomnography signals, which is a tedious visual task. Thus, research efforts to develop an automatic sleep stage scoring based on machine learning techniques have been carried out in the last several years [79]. Convolutional neural networks (CNN) [80] and Long-Short Term Memory Recurrent Neural Networks (LSTM) [81] provide an interesting framework for automated classification of sleep based on raw waveforms. In the past few years, Deep Neural Networks (DNNs) have accomplished tremendous success for time series tasks compared to traditional machine learning systems. Recently, further improvements over DNNs have been obtained with alternative types of neural network architectures. CNNs, LSTMs, and DNNs are individually limited in their modeling capabilities, and we believe that time series data classification can be improved by combining these networks in a unified framework.

The classification of time series signals presents many challenges that make it a uniquely difficult problem in machine learning. Many feature extraction approaches in time series face issues related to the signal's non-stationary nature when the probability distribution does not change over time. Accordingly, features such as mean and variance will not change. Furthermore, the physiological signals are very noisy, susceptible to posture, mood, physical movement, and external noise [82]. Lack of comparability between experiments is another issue that can be faced in this field. Unlike in image classification, there are no standard time series datasets used as performance benchmarks [21, 83]. Some approaches use models for individuals, while others try to make a general model, training, and testing with samples from all individuals at one time.

In this work, we propose a method for classifying sleep stages based on the CNN,

LSTM and DNN with the help of transfer learning. More specifically, we use a transfer learning technique to train our network model with sleep posture data for 56 subjects (source dataset) and use it for sleep stage classification (target data). The sleep data was obtained from 5 subjects and it was collected in the Boone Hospital Center (BHC) in Columbia, MO, USA under the University of Missouri IRB approval, project number 2008526. The main contribution of this work is developing a new deep model architecture that utilizes CNNs and LSTMs to classify sleep stage data. The CNNs are trained to learn filters that extract time-invariant features from the BCG signals while the LSTMs are trained to encode temporal information such as sleep stage transition rules.

4.2 Sensors and Datasets

A home monitoring system using a ballistocardiography (BCG) hydraulic sensor has been developed to monitor sleep at home (see Fig. 4.1). The hydraulic bed sensor has been developed at the Center for Eldercare and Rehabilitation Technology (CERT) at the University of Missouri. A BCG device provides a noninvasive, low-cost, and robust solution for capturing physiological parameters such as heart rate and respiration rate, during sleep [39, 60, 61]. The BCG sensor has four transducers. Each transducer is composed of a water tube with a pressure sensor placed at one end. The water tube is 50 cm long and 6 cm wide and it is filled with about 0.4 liters of water. The BCG sensor is placed under the mattress, parallel to the body direction, to provide sleeping comfort and not to disturb a person's normal sleep pattern.

The pressure outputs are coupled to a Maxim MAX7401 which is a filtering circuit that consists of a 741 operational amplifier and an 8th-order integrated Bessel filter [16, 62]. The four-channel signal is sampled and quantized to 12-bit precision. The BCG signal acquired from the sensor is superimposed over the respiration signal. The

four matching transducers are independent; consequently, the data quality collected by those transducers might vary depending on the subject's sleeping position, type of bed (*e.g.*, material, thickness, *etc.*) and the physical characteristics of the subject such as age and body mass index (BMI). In this study, two kinds of datasets have been collected utilizing our bed sensor: sleep posture and sleep stages.

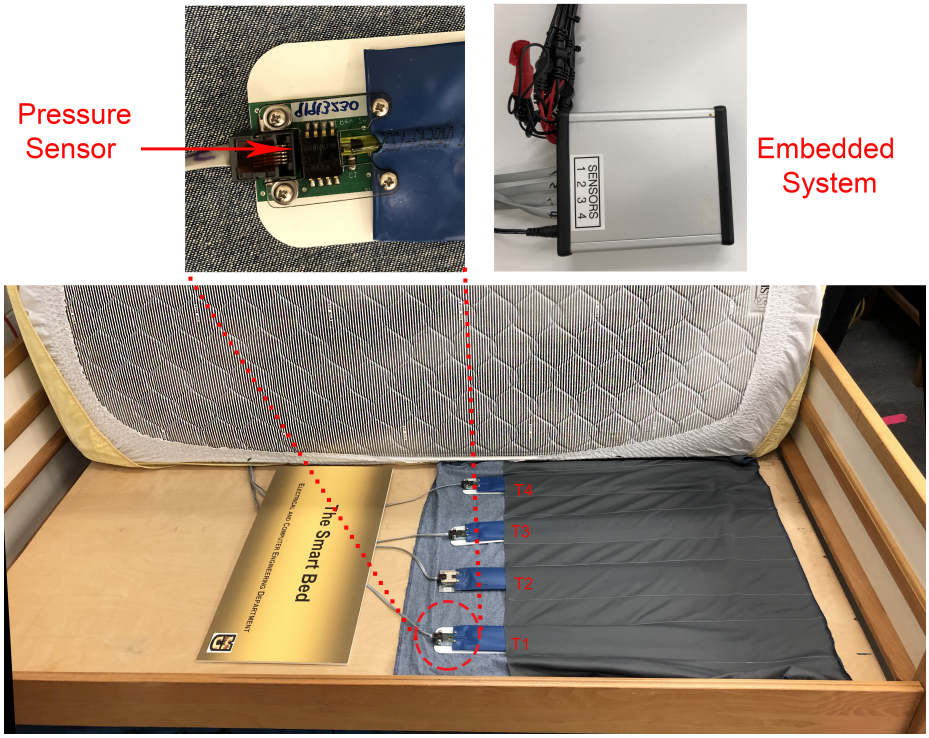


Figure 4.1: Hydraulic Bed Sensor System.

4.2.1 Posture Dataset

A total of 56 young healthy subjects (75.8% of males and 24.2% of females) were asked to lie still in our lab for one minute on each of the four main postures, supine, prone, left lateral and right lateral. The data collection procedure was approved by the University of Missouri Institutional Review Board (MUIRB). The bed sensor produced data sampled at 100 Hz. The subjects age ranged between 18 and 49 years

(mean 29.3 years); the weight ranged between 48 and 127 kg (mean 75.9 kg); the height ranged between 156 and 190 cm (mean 174.4 cm); the average body mass index was 24.8 kg/m^2, ranged between 18.1 and 37.9 kg/m^2. Fig. 4.2 shows the characteristics of 56 participants.

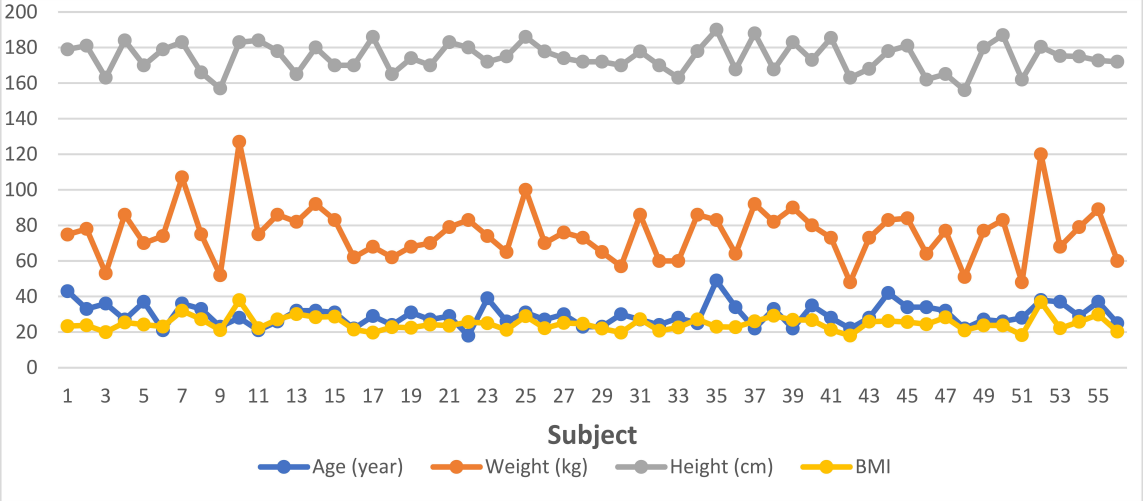


Figure 4.2: Characteristics of the 56 source participants.

4.2.2 Sleep Stage Dataset

Sleep stage data was collected from consenting patients by a sleep-credentialed physician during the regularly scheduled PSG studies conducted in the sleep lab of the Boone Hospital Center (BHC) in Columbia, MO, USA. In addition to the regular PSG equipment, we placed our hydraulic bed sensor under the study bed mattress. The scoring system for staging the sleep was based on the American Academy of Sleep Medicine Manual (AASM) [84], which is the standard for scoring sleep stages and it provides guidelines for associated events during sleep. Sleep stage scoring in 30-second epochs is required for the AASM protocol. A program produced by Natus SleepWorks (Natus Medical Inc., San Carlos, CA, USA) was used in the BHC sleep lab to assist the staff in monitoring a patient throughout the night. SleepWorks not only collects the PSG data but also performs a video recording of the entire night.

The technician can view the patient's sleep video if they have any uncertainty about the data. SleepWorks can also provide an initial analysis of the collected data and can generate a report, which helps the sleep physicians who make treatment recommendations.

Fig. 4.3 shows the PSG signals visualized in the Natus SleepWorks interface; the orange bar is the occurrence of obstructive apnea that annotated by a technician. The BHC sleep lab provides de-identified polysomnography (PSG) data with a sample rate of 256 Hz. PSG is a multiparametric recording method used in sleep labs to monitor physiological changes throughout sleep. It is a consistent tool for diagnosing sleep disorders, and it can similarly help adjust treatment. Moreover, a technician scores each patient's clinical events (e.g., limb movements, respiratory and cardiac events, and arousals) using the American Academy of Sleep Medicine (AASM) standards. PSG is used in this research as ground truth for our bed sensor signals. The hypnogram is one PSG outcome that displays the sleep stages as a function of time (Fig. 4.4). Sleep stages are annotated in 30-second epochs. From top to bottom, the sleep stages are Wake (W), REM(R), NREM1 (N1), NREM2 (N2), and NREM3 (N3). For the patient shown in Fig. 4.4, 742 epochs were monitored during sleep. REM sleep happened between epoch 392 and epoch 485, also highlighted in red.

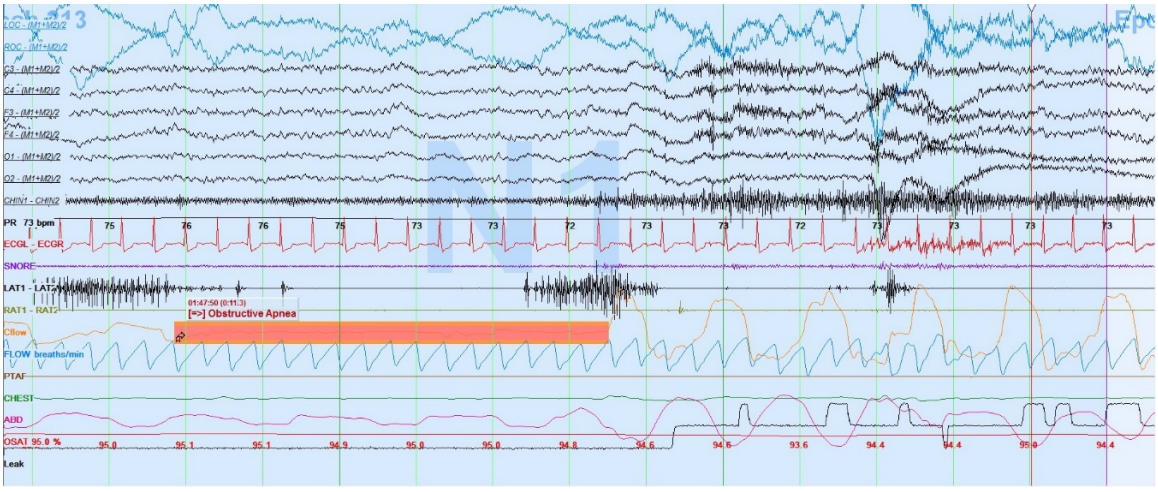


Figure 4.3: PSG signals visualized in the Natus SleepWorks interface.

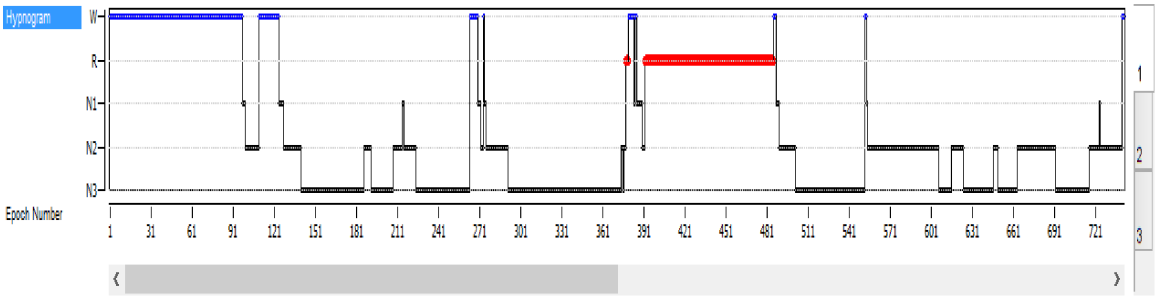


Figure 4.4: Hypnogram of an entire night exported from a PSG system.

Those who participated in the sleep study are likely to have a sleep disorder. The partial and the complete collapse of the airways are called hypopnea and apnea, respectively. Lack of airflow will affect breathing patterns and then influence sleep stage sequence. The Apnea-hypopnea index (AHI) represents the number of apnea and hypopnea events per hour during sleep. Based on the AASM, the mild sleep apnea AHI is between 5 and 15. In this study, we did not use the data collected from patients with severe apnea symptoms. For this study, we selected five sleep lab patients with a low Apnea-hypopnea index (AHI), so that each one has sufficient REM, NREM, and Wake sleep stages during the night.

4.2.3 Data Preprocessing

We present the preprocessing steps for our data. For the posture data, we truncate the signals utilizing the percentile method [46] to remove any noise that was introduced due to the desynchronization between the time of leaving the bed and stopping recording the signal. After denoising, we normalize the data by utilizing standard score normalization technique. Fig. 4.5 shows the these stages of cleaning and normalizing the data. Then we down-sample the data to 100 Hz and shuffle it to prepare it for net training.

To account for the varying length of the signals, we divide the preprocessed bed sensor signals into 5-second segments with 80% overlap (see Algorithm 3). Each

output label corresponds to a segment of the input. Together the output labels cover the full sequence. The above two algorithms are applied on posture and sleep stage data. Another algorithm is applied for sleep stages only, which are the rules from [85] that help to improve the accuracy about 1%. These rules can be illustrated as followed (see Algorithm 4):

- For three consecutive 30s epochs, if the center one is not the same as the other two and the other two are the same, change the center one to the same stage as the other two epochs. However, the center epoch is not changed if it is an Awake stage.

- For every three 30s epochs, if the sleep stages are all different, the center epoch is removed from the recording. Similarly, this rule does not apply to the Awake stage.

- If REM stages show up in the first hour, these stages are removed from the recording.

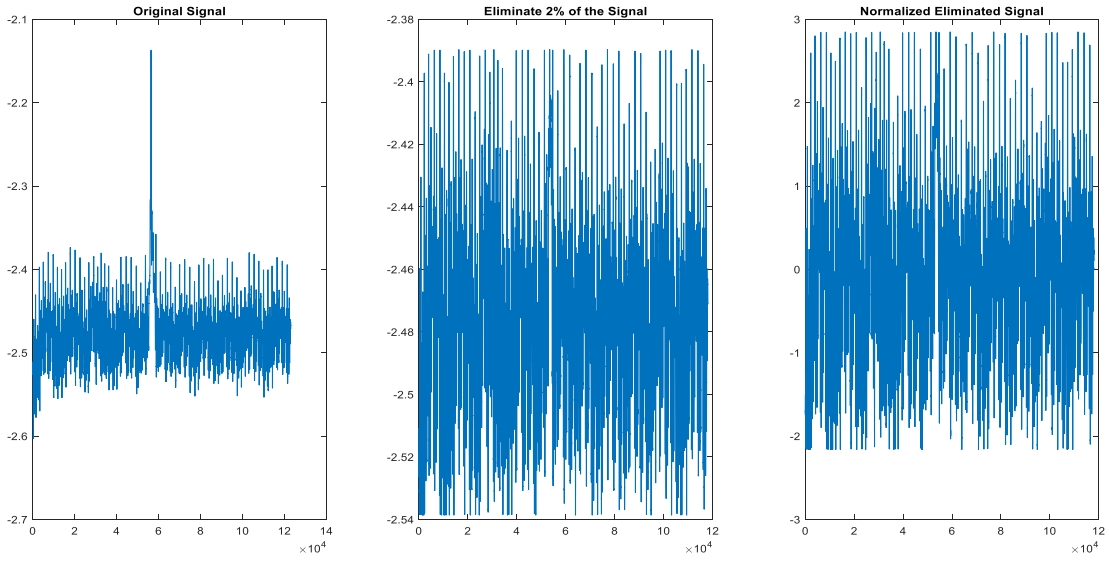


Figure 4.5: Sample of BCG signal with cleaning and normalizing stages.

Algorithm 3: Extract Segments

1

Input: $\widetilde{bcg}$:= Processed bed sensor Data, *win_size* : windows size.

Output: *Seg*:= segments, *LB*:=labeles, *GR*:= groups, *T_Seg*:= total segments, *T_LB*:= total labels, *T_GR*:=total groups.

2 **Initialization**

3 **begin**

4 Load($\widetilde{bcg}$)

5 **foreach** $\widetilde{bcg_i} \in \widetilde{bcg}$ **do**

6 **while** *start < length of* $\widetilde{bsd}$ **do**

7 end = start + win_size

8 start=start + stride

9 **end**

10 **foreach** *(start, end) in slice(*$\widetilde{bcg_i}$*, win_size)* **do**

11 **if** *size of* $\widetilde{bcg_i}$ *(start to end) == win_size* **then**

12 signal $\Leftarrow$ $\widetilde{bcg_i}$(start to end)

13 **if** *Seg is None* **then**

14 segments = signal

15 **else**

16 stack signal in Seg

17 **end**

18 LB $\Leftarrow$ $\widetilde{bcg_i}$(0,end-1)

19 GR $\Leftarrow$ $\widetilde{bcg_i}$(0,end-2)

20 **end**

21 **end**

22 *T_LB* $\Leftarrow$ stack LB

23 *T_GR* $\Leftarrow$ stack GR

24 *T_Seg* $\Leftarrow$ stack Seg

25 **end**

26 **end**

Algorithm 4: Preprocessing

Input: bsd:=Bed Sensor Data
Output: $\widetilde{bsd}$:= Processed Bed Sensor Data
begin
  Load(bsd)
  Clean(bsd)
  foreach $\delta_i \in bsd$, *where* $i = \{2, size(bsd)\}$ **do**
    Perc $\Leftarrow$ Percentiles (δ_i)
    $\delta_i(\delta_i \leq \text{Perc}(1)) \Rightarrow$ eliminate;
    $\delta_i(\delta_i \geq \text{Perc}(2)) \Rightarrow$ eliminate
  end
  foreach $\delta_i \in bsd$ **do**
    Normlize the data

$$\overline{\delta_i} = \frac{\delta_i - mean(\delta_i)}{std(\delta_i)} \tag{4.1}$$

    $\overline{\delta} \Leftarrow \overline{\delta_i}$
  end
  $\breve{\delta} \Leftarrow$ Downsample if need it($\overline{\delta}, 100Hz$)
  $\dddot{\delta} \Leftarrow$ Label&Group($\breve{\delta}$)
  $\widetilde{bsd} \Leftarrow$ Shuffle($\dddot{\delta}$)
end

4.3 Architecture Design

We propose a new architecture to classify BCG bed sensor data (see Fig. 4.6.). The network takes as input the windows of time series of filtered BCG signal and outputs a sequence of label predictions. The basic block is a convolution layer followed by a batch normalization layer [86], Rectified Linear Unit (ReLU) activation layer, and max-pooling layer. We similarly employ shortcut connections to those found in the Residual Network architecture [87] to make the optimization of such a network manageable. The shortcut connections between neural network layers enhance training by

permitting information to propagate well in deep neural networks. The convolutional layers in the first and second blocks have 8 filters; this number of filters doubles in successive blocks until it reaches 32 in the last block.

Moreover, the Max Pooling layer helps to subsample the input to become one-fourth of the input sample at the top layer since Max Pool is set to size two. When a block subsamples the input, the corresponding shortcut connections also subsample their input using a Max Pooling operation with the same subsample size. We next pass the CNN block output to LSTM layers with 128 units, which are appropriate for modeling the signal in time.

Regularization techniques are employed in our architecture to reduce over-fitting effects during training. Regularization techniques help keep the model from becoming too complex and specific to the training data, thus reducing the tendency to overfit. In this work, two regularization techniques were used. The first technique was a dropout, which randomly sets the input value to zero with a certain probability [27]. A probability of 0.5 was used in the dropout layer after LSTM, as shown in Fig. 4.6.

The second technique was L2 weight decay, which adds a penalty term into the loss function. L2 regularization is a typical technique used in many optimization methods, in which the squared sum of the weights is applied as a penalty to the optimization function. In essence, this weighs the advantage of increased classification of the training data against model complexity. By preventing the model from becoming too complicated, memorization of the training data is reduced, and a more generalizable model is developed.

4.4 Training and Experimental Results

Transfer learning is employed in the experiments of this work to classify sleep stages. Transfer learning is a technique wherein the knowledge gained from training a model

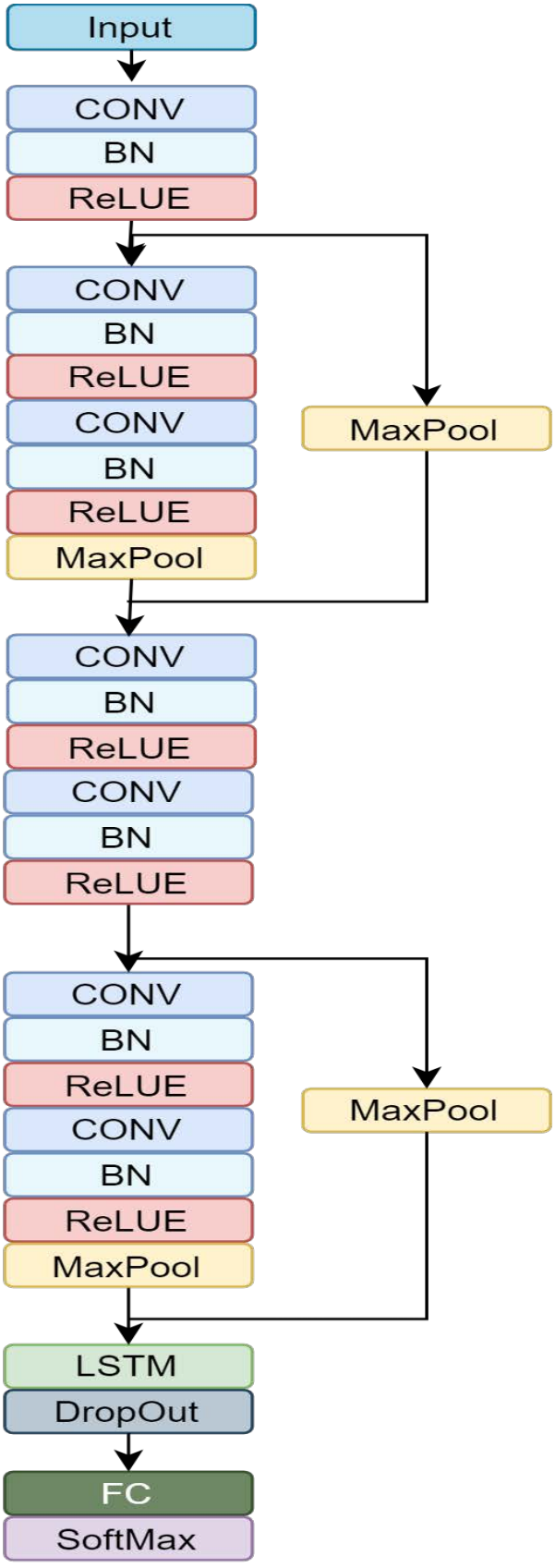


Figure 4.6: The proposed architecture of the network

on a dataset (source) can be reused as a starting point, when fine-tuning the model on another dataset (target). This is usually done when the target dataset is not as rich as the source, and hence, a complex deep network would have more trouble training only on the target (see Fig. 4.7). Consequently, we split the network model training into two distinct phases. First, our model is trained on the class-balanced training posture dataset of 56 people using five-fold cross-validation. Then, the network weights obtained in the first phase are used as an initialization for the second phase of training on sleep stage data subjects.

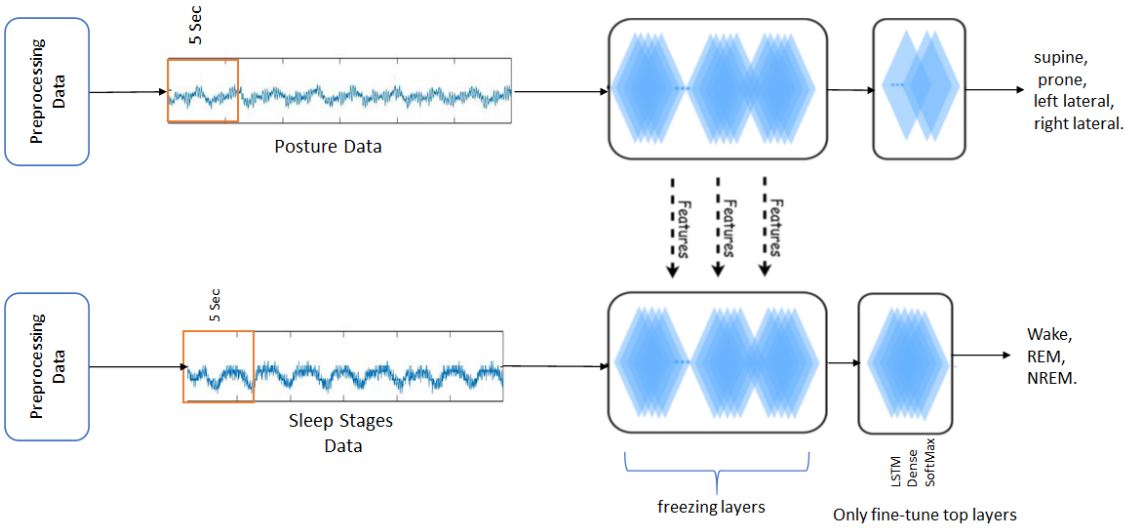
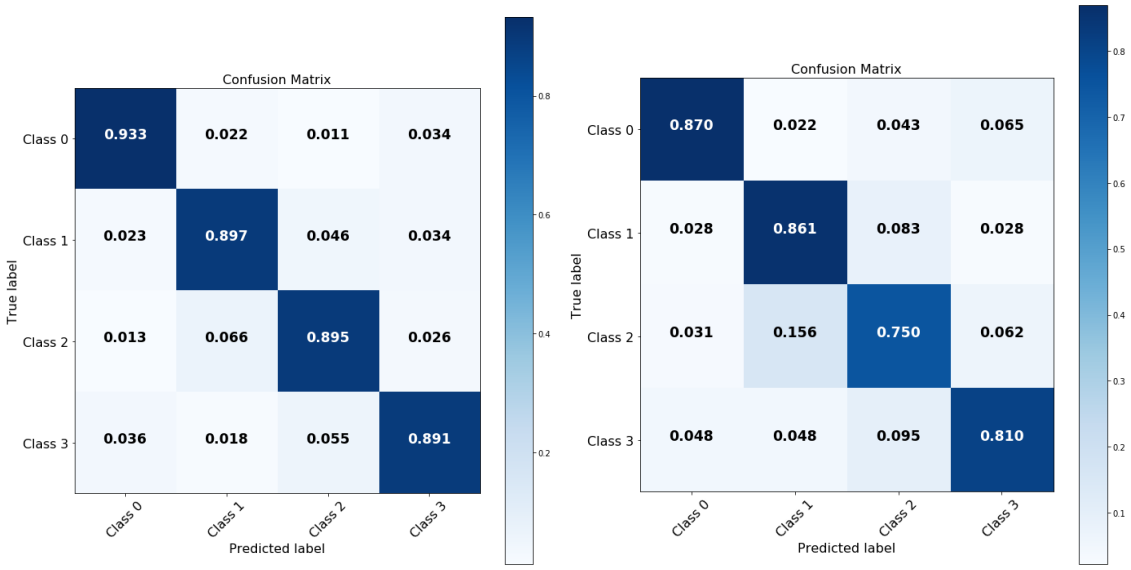


Figure 4.7: System framework, the features of posture data were utilized to train the sleep stages data and fine-tune the top layers of the model.

The model is trained and tested on the posture data only to check the validity of the architecture. Five-fold cross validation is conducted to train and test posture data. Each time (fold), 12 patient was left out for testing while the remaining 44 patients were used to train the DNN networks. Fig. 4.8 (a) and (b) show the training and testing phase respectively for the 4 postures Supine, Left Lateral, Right Lateral, and Prone posture. Our model was able to classify the posture data better than the results reported by Enayati, *et al.* [88] which were 72% the average accuracy for the four postures; the previous work has been done using the same data and the same

sensor, but with traditional machine learning algorithms.



(a) (b)

Figure 4.8: Confusion matrix of 56 patients five-fold non-hierarchical posture data classification ; Class 0: Supine posture, Class 1: Left Lateral, Class 2:Right Lateral, Class 3: Prone posture (a) training phase (b) testing phase.

After testing our architecture with posture data, a five-fold LOSO-CV is conducted to fine-tune the sleep stage data. Each time (fold), one patient was left out for testing while the remaining four patients were used to train the DNN networks. In the testing part, we utilize a hierarchical classification scheme for our automatic sleep stage classification system. This structure is composed of two layers as shown in Fig. 4.9.

The first layer separates wake from sleep, which is a union of REM and NREM, in a binary classification problem. Next, all the epochs classified as sleep are fed into the next layer to further classify the REM and NREM epochs. The last block of the architecture, shown in Fig. 4.6 is used to fine-tune the sleep stage data. In the first phase, the optimal hyperparameters for the model are carefully chosen for a given dataset.The model is at that time fine-tuned on the given dataset with these hyperparameter settings.

Training overnight raw sleep stage data using DNN is time-consuming which might takes weeks or months. Moreover, the ground truth of the posture data is more reliable than the ground truth of sleep stage data. Deep learning models with transfer learning can, however, successfully converge on short-time data (sleep posture data) split across multiple subjects. By training a model over multiple posture data subjects, we obtained an initialization that is then used for fine-tuning our model on the sleep stage data. Based on Williams *et al.* [21], this process is hypothesized to make convergence on an individual's data more likely; it should also provide more robust general filters in the first layers of the neural network.

Fig. 4.10 (a) and (b) show the average fine-tuned training five-fold LOSO- CV sleep stage data to classify wake from sleep and classify REM from NREM, respectively. Fig. 4.10 (c) shows the average of hierarchical fine-tuned testing five-fold LOSO-CV sleep stage data. Fig. 4.11 (a) and (b) shows PSG sleep stages ground truth for 5 subjects and sleep stages predicted labels for 5 subjects respectively . Clearly, our model was able to classify the sleep stage classes better than the results reported by Yi, *et al.* [89] which were 79.9%, 78.8%, and 88.8% sensitivity for wake,

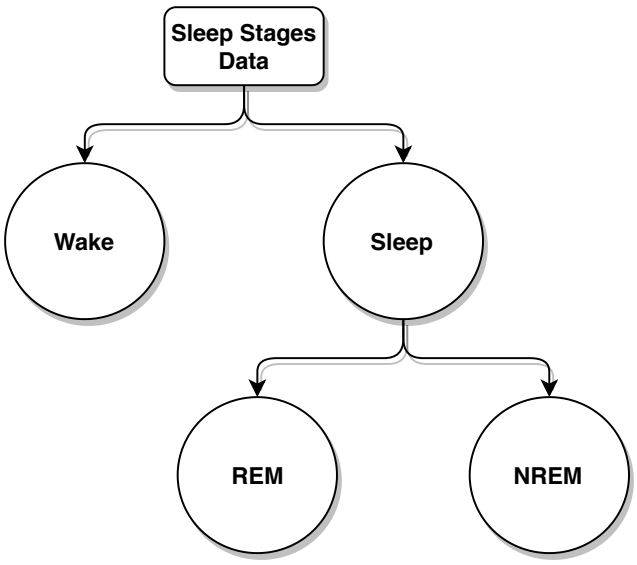


Figure 4.9: Sleep Stages hierarchical classification steps.

REM, and NREM respectively; the previous work has been done using the same data and the same sensor, but with traditional machine learning algorithms. Table 4.1; shows the percentage of epochs that are correctly classified for each class Wake, REM,

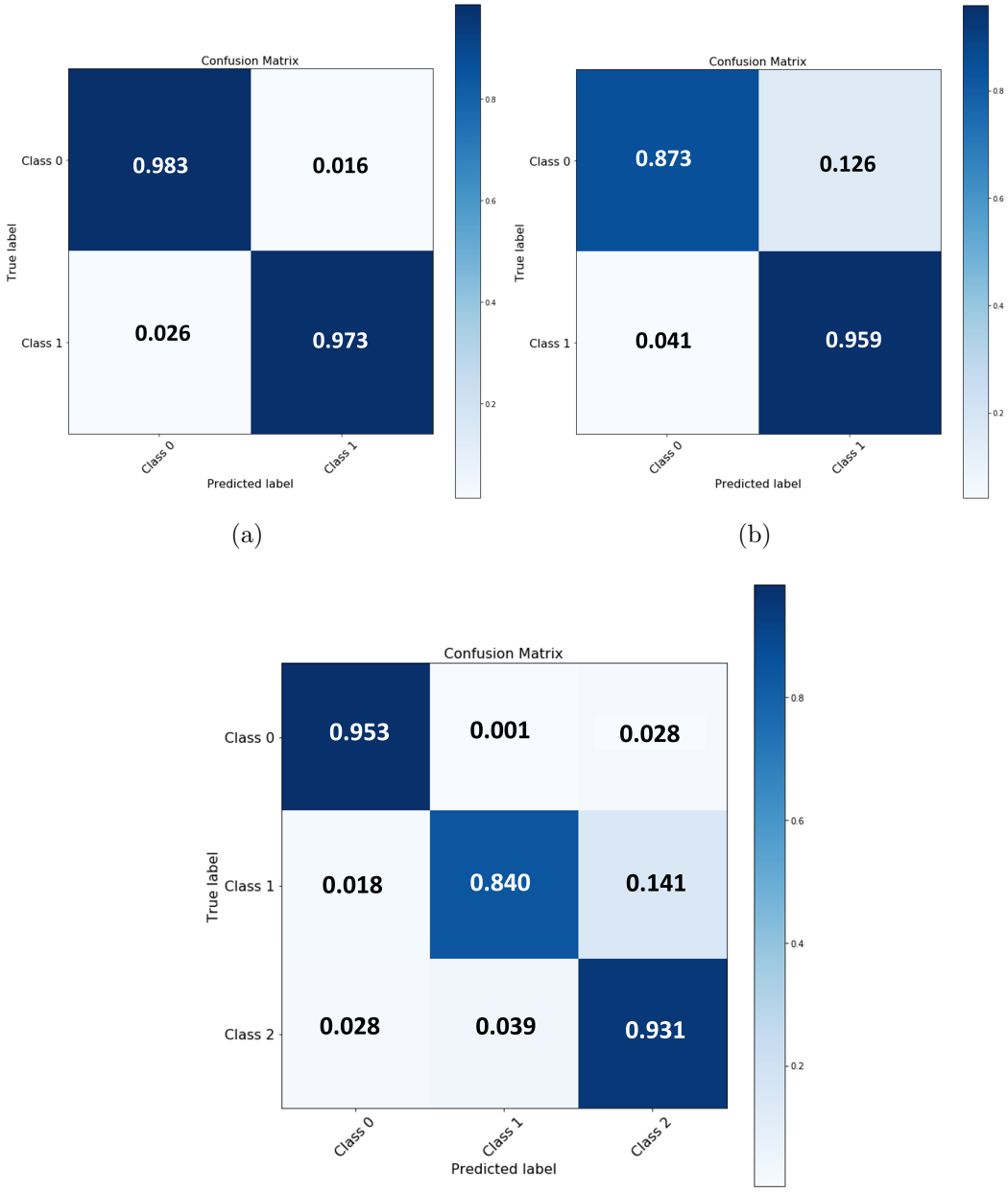
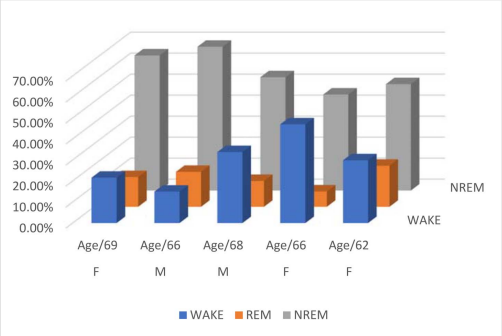
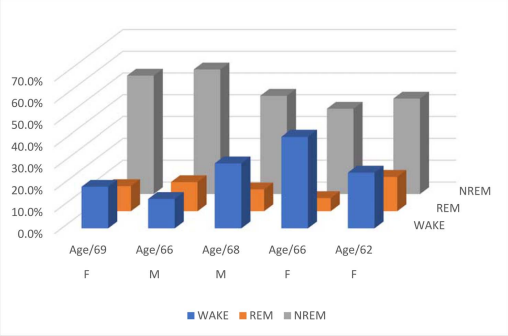


Figure 4.10: Confusion matrix of 5 patients leave-one-subject-out hierarchical sleep stages classification utilizing posture data knowledge (a) Training phase, Class 0: Wake, Class 1: Sleep (b) fine-tuning training, Class 0: REM, Class 1: NREM (c) Hierarchical fine-tuning testing phase Class 0: Wake, Class 1: REM, Class 2: NREM



(a)



(b)

Figure 4.11: Experimental results; (a) PSG sleep stages ground truth for 5 subjects and (b) Sleep stages predicted labels for 5 subjects.

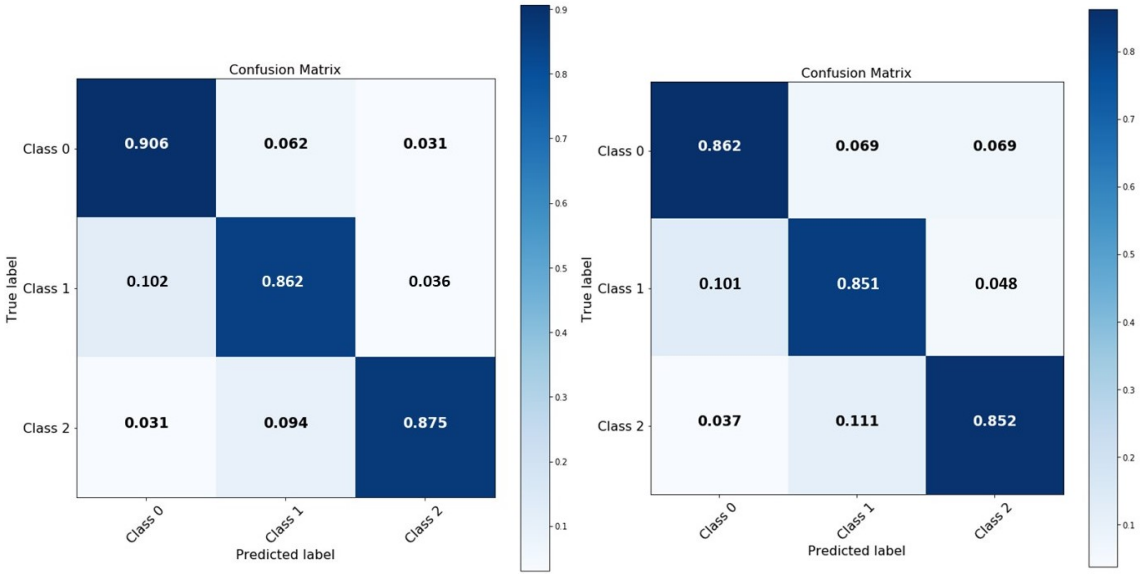
and NREM comparing to the percentage ground truth. From the results shown in Fig. 4.10, and Table 4.1, it is worth to mention the misclassification of the Wake as Sleep in the first layer in the hierarchical method is fed into the second layer, which has a cumulative impact on the second layer classification.

Table 4.1: Selected five sleep lab patients with a low Apnea-hypopnea index (AHI).

Information			Ground Truth			Predicted Labels		
SUBJECT	GENDER	AGE	WAKE(%)	REM(%)	REM(%)	WAKE(%)	REM(%)	NREM(%)
1	F	69	21.57	14.13	64.29	19.13	11.54	54.32
2	M	66	14.95	16.61	68.44	13.53	13.39	57.14
3	M	68	33.76	12.3	53.94	29.83	10.04	45.03
4	F	66	46.98	7.32	45.7	42.01	6.03	39.07
5	F	62	29.85	19.53	50.63	25.58	15.75	43.74

The same method is used to classify the data with the non-hierarchical classification technique. Fig. 4.12 (a) and (b) show the average of non-hierarchical fine-tuned training and testing five-fold LOSO-CV sleep stage data respectively to classify wake, REM, and NREM. The REM class results were a bit better in non-hierarchical classification than hierarchical classification. In contrast, the wake and NREM classes

were less accurate.

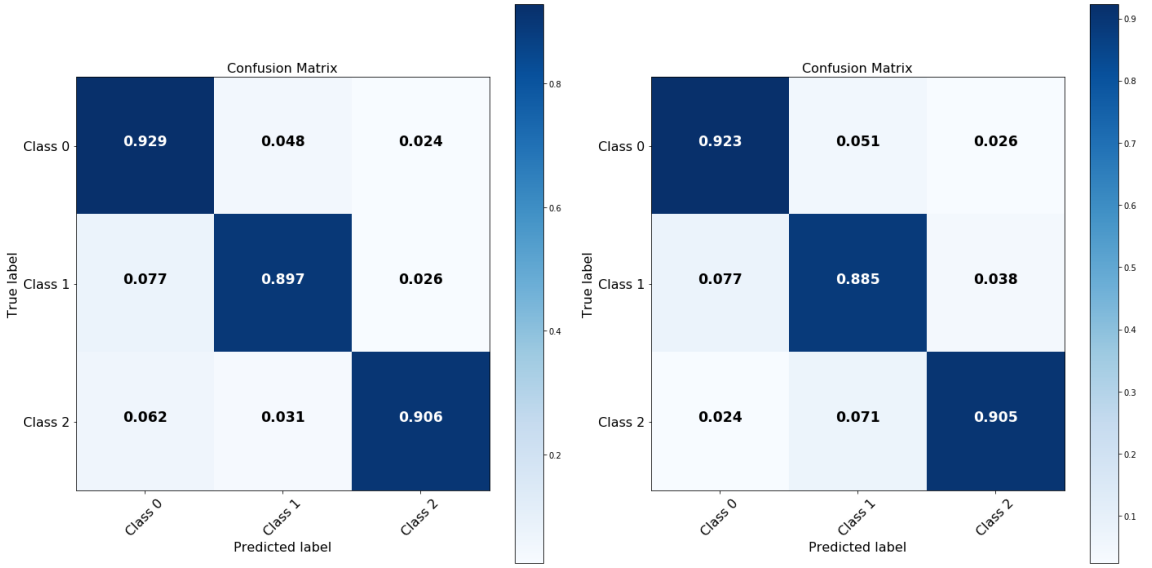


(a) (b)

Figure 4.12: Confusion matrix of 5 patients five-fold LOSO-CV non-hierarchical sleep stages classification utilizing posture data knowledge; Class 0: Wake, Class 1: REM, Class 2: NREM (a) fine-tuned training phase (b) fine-tuned testing phase.

The same five patients are used again with the LOSO-CV method directly without using posture knowledge. This experiment aims to see the difference between training the five subjects using the knowledge from the posture data and using the sleep stages data directly without. Fig. 4.13 (a) and (b) show the average confusion matrix of five-fold LOSO-CV for the training and testing phase, respectively. Although the results for training the sleep stages data directly without using posture knowledge is better, the time consuming for training the 8 hours/subject data is much more than using the posture knowledge, which was 4 minutes/subject.

Although the starting time of the two systems is synchronized, there were some interruptions during sleep. During the study, forty-four out of 71 patients went to the restroom. The number of restroom visits varied from one to five times. When a patient leaves the bed, the technician must detach all the devices worn on that person's body surface and suspend the PSG data collection. At the same time, the



(a) (b)

Figure 4.13: Confusion matrix of 5 patients LOSO-CV non-hierarchical sleep stages classification; Class 0: Wake, Class 1: REM, Class 2: NREM (a) Training phase (b) testing phase.

bed sensors are supposed to stop collecting the data. However, the bed sensors kept on collecting data while subjects got out of bed.

Fig, 4.14 shows an annotation example of the PSG system with the epoch number labeled at the top left of each epoch's starting. As shown in the figure, during epoch 55, the system was disconnected and then reconnected during epoch 62. The restroom visit timestamp in the bed sensor system is from 22:38:19 to 22:42:10, according to the results of data synchronization. Nevertheless, due to a hardware glitch, the bed sensors kept collecting data while the patient was away from the bed. Since the sleep stage is based on 30-second epochs, we do not want to keep partial sleep stages.

In the PSG system, we look for entire epochs before the disconnected time point. In this example, the whole epoch before disconnection is epoch 54, and the entire epoch after the reconnected time point is epoch 63. The incomplete epochs with the yellow background in Fig. 4.15 were removed. For the bed sensor system, all the data between epoch 54 and epoch 63 is removed. The remaining signals are the multiple

integral segments of 30-second epochs. We then concatenated all the signals together.

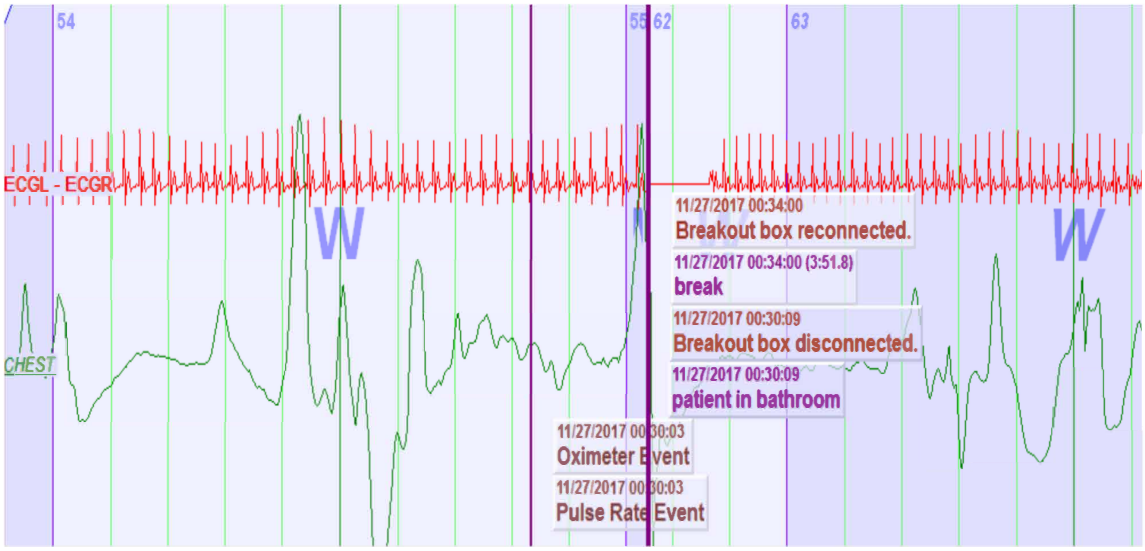


Figure 4.14: Annotation example of PSG system with break.

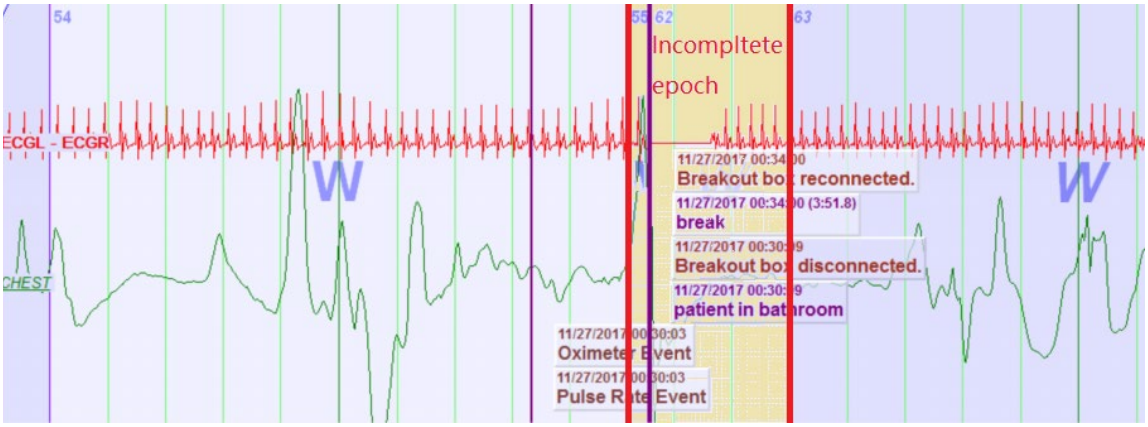


Figure 4.15: Incomplete epoch removed from PSG system.

Hence, The same training and testing procedure is applied again, but 27 patients of Boone hospital sleep stages data are used. None of these 27 patients has left the bed or interrupted the study. About 80% of them (22 patients) are used to train the architecture. Then five-fold LOSO-CV is conducted to fine-tune the rest of the data, about 20% (5 patients). Each time (fold), one patient was left out for testing while the remaining four patients were used to fine-tune the top layers of the DNN networks. Fig. 4.16 (a) shows the non-hierarchical training phase's confusion matrix-

ing 22 patients from Boone hospital. Fig. 4.16 (b) and (c) show the average of non-hierarchical fine-tuned training and testing five-fold LOSO-CV sleep stage data to classify wake, REM, and NREM. The results show an accuracy of 80%, 76%, and 78% for awake, REM, and NREM, respectively, on this group of patients from the sleep lab. Five patients' results were better than the 27 patients because they have a low Apnea-hypopnea index (AHI).

The algorithm is then applied to 71 patients; some of these people have left the bed, and others are not, as I explained earlier. Since the sleep stage is based on 30-second epochs, we do not want to keep incomplete sleep stages. In the PSG system, we look for complete epochs before the disconnected time point. All the data between the complete epoch before disconnection and the complete epoch after reconnecting are removed from the bed sensor system. The remaining signals are the multiple integral segments of 30-second epochs. Then all the signals are concatenated together.

A hardware malfunction caused the second exception. The bed sensor missed some part of the data, even after the patient returning from the restroom. For some hardware reason, the bed sensors sometimes experienced a seven-minute delay before they could start collecting data again. However, in the PSG system, sleep stages were labeled during this period. The synchronization process was similar to the previous one, and the complete epoch was found to be coincident with the timestamp data. Then, the sleep stage labels were removed to match the missing data.

Thus, 56 out of 71 patients (about 80%) are used to train the architecture. Then five-fold is conducted to fine-tune the rest of the data, which is 15 patients (about 20%). Each time (fold), three patients were left out for testing while the remaining 12 patients were used to fine-tune the top layers of the DNN networks.

Fig. 4.17 (a) shows the non-hierarchical training phase's confusion matrix using 71 patients from Boone hospital. Fig. 4.17 (b) and (c) show the average non-hierarchical fine-tuned training and testing five-fold sleep stage data to classify wake, REM, and

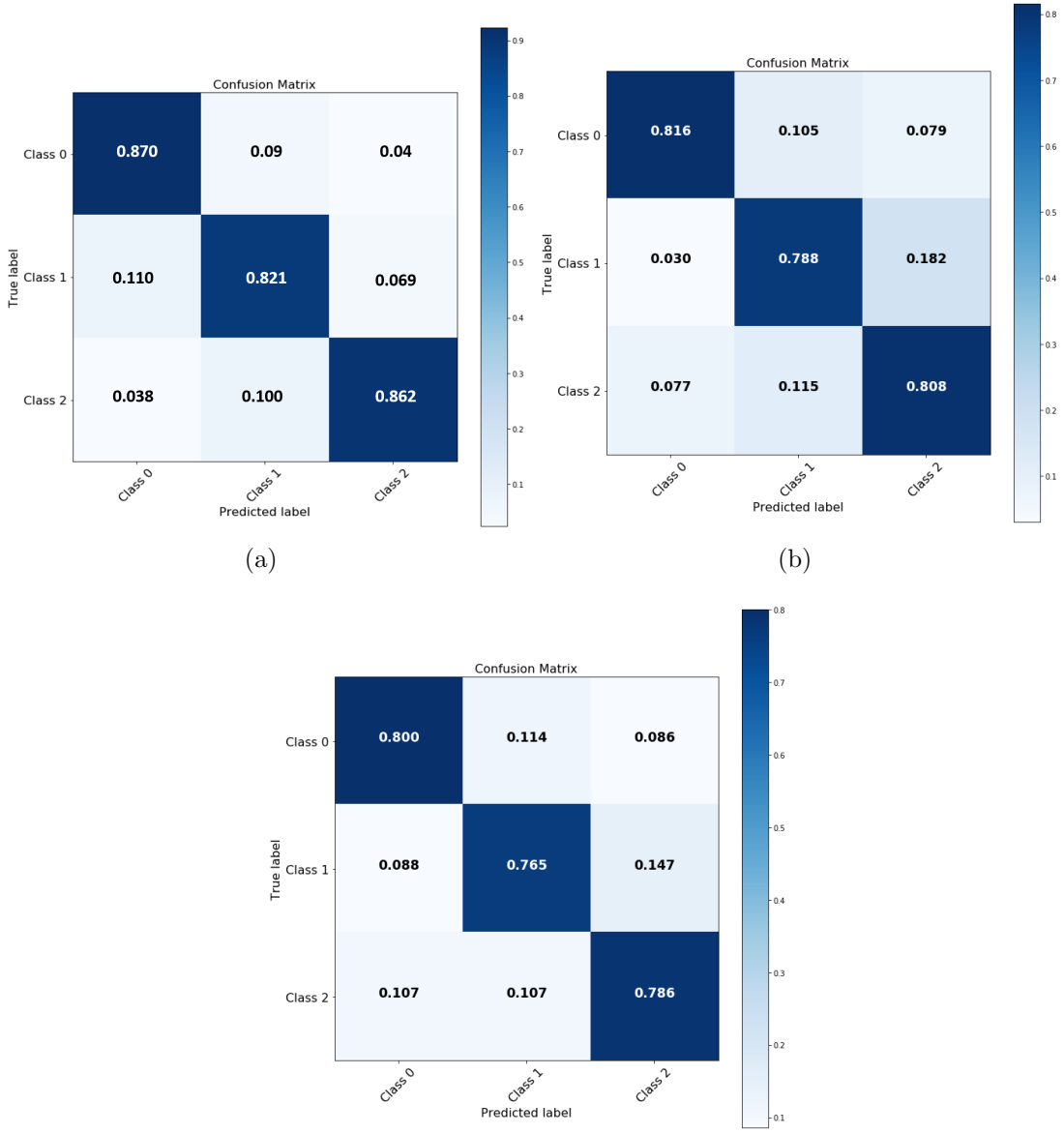


Figure 4.16: Confusion matrix of 27 patients non-hierarchical sleep stages classification; Class 0: Wake, Class 1: REM, Class 2: NREM; (a) training phase for 22 patients (b) fine-tuning training phase five-fold LOSO-CV for 5 patients (c) fine-tuning testing phase five-fold LOSO-CV for 5 patients.

NREM.

The results of 27 patients were better than those of 71 patients because some 71 patients have left the bed as many as five times in some cases. After the patients get back to bed, it will be hard for them to sleep, and in many cases, they stay awake

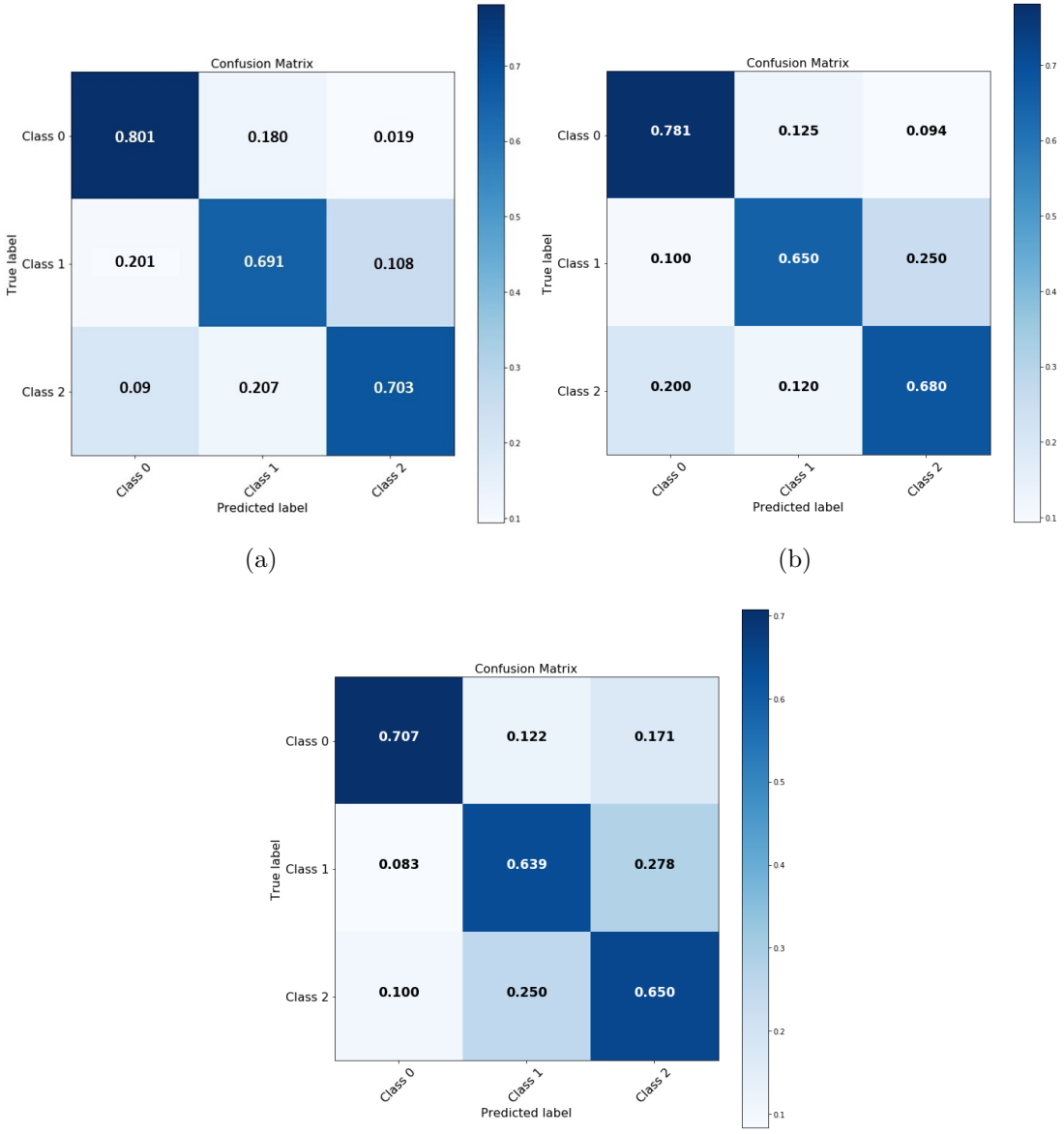
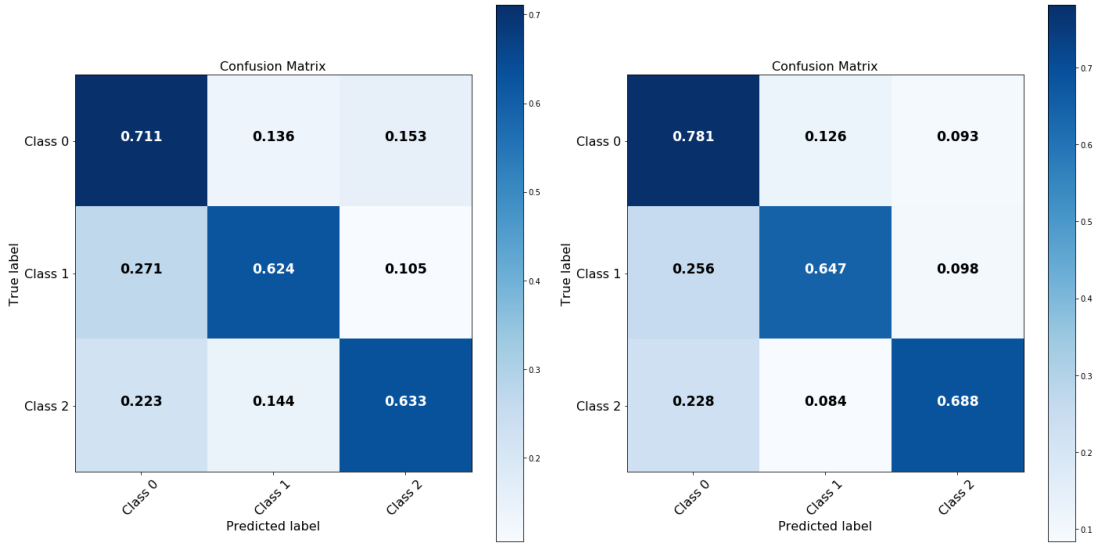


Figure 4.17: Confusion matrix of 71 patients non-hierarchical sleep stages classification; Class 0: Wake, Class 1: REM, Class 2: NREM; (a) training phase for 56 (about 80%) patients (b) fine-tuned training phase (c) fine-tuned testing phase.

until the morning. The testing accuracy of 71 patients were 70.07%, 63.9%, and 65%, respectively for the Wake, REM, and NREM, The accuracy of 27 patients were 80%, 76.5%, and 78.6% for the Wake, REM, and NREM, respectively (see Fig. 4.17 and Fig. 4.16). Moreover, the results of the 5 subjects was much better than both 27

and 71 patients because they have a very low Apnea-hypopnea index (AHI), so that each one has sufficient REM, NREM, and Wake sleep stages during the night. The accuracy of the 5 patients were 92.3%, 88.5%, and 90.5% respectively for the Wake, REM, and NREM (see Fig. 4.13).

The last experiment compares the transfer learning method utilizing posture data and the traditional training and testing method. The 15 patients (out of 71) are used in testing directly without fine tunning using the last experiment's network of the Boone hospital data. Hence, the architecture is trained on 56 patients (about 80%) and testing on 15 patients directly, as shown in Fig. 4.18 (a). Then, the same 15 patients are used one more time with posture network. However, the 15 patients are used in the five-fold-CV testing phase, as shown in Fig. 4.18 (b). The testing phase results for transfer learning and and traditional methods are close; however, training the posture data network (4 minutes subject) requires much less time than training the sleep stages data (8 hours patient).



(a) (b)

Figure 4.18: Confusion matrix of the testing phase for 15 patients; non-hierarchical sleep stages classification; Class 0: Wake, Class 1: REM, Class 2: NREM; (a) Confusion matrix of the testing phase for 15 patients directly (b) Confusion matrix of the five-fold CV testing phase for 15 patients utilizing posture knowledge.

4.5 Conclusion

This work has developed a deep learning-based hierarchical classification method for automatic sleep stage classification based on BCG data. The deep learning model is a state-of-the-art algorithm that consists of a stacked CNN-LSTM model. The proposed model was trained using a transfer learning approach that achieved a significantly improved performance in comparison to similar studies. The results of the leave-one-out cross-validation strategy showed potential in automatically classifying sleep stage epochs. Our sleep monitoring system based on the BCG can indeed provide a more natural way of diagnosing sleep problems, and make long-term sleep monitoring possible. However, to validate our algorithms, we used a selected, balanced dataset with a reduced noise level. We are currently working to increase our methodology's robustness and validate it for patients with reduced REM and high AHI.

Chapter 5

Early Illness Recognition in Older Adults Using Transfer Learning

5.1 Introduction

As the US population ages, there has been a significant increase in the development of health monitoring technologies. The likelihood of developing a health problem increases as a person ages, but many elderly adults in the US want to live independently for as long as possible despite conditions such as frailty and dementia and the increased risk of falling [90, 91]. Home sensor networks have emerged as a viable solution that can allow seniors to age in place.

Our living lab, called TigerPlace [92, 93], is an aging-in-place facility in Columbia, Missouri. Since 2005, 60 apartments at TigerPlace have been fitted with sensor networks that include motion sensors, bed sensors, and depth cameras. In this work, we use data from bed and motion sensors (PIR) deployed in various rooms of the apartment (e.g., bathroom, bedroom, kitchen, and living room). The motion sensors provide information related to resident activity around the apartment. The bed sensor provides information related to bed motion (low, medium, high), heart rate (low,

high) and respiration (low, high). The data provided by these sensors are securely sent to an off-site level-4 database. The sensor information can be visualized over a web interface by TigerPlace clinical personnel. The sensor data capture the behavior and physiological information of the resident, and this information is then linked to medical records in the TigerPlace EMR.

An EMR nursing report is said to describe negative or abnormal behaviors if the patient exhibits abnormal behaviors that can be linked to health problems, while the EMR report is said to be positive or normal if no such behaviors are exhibited. While this classification of behavior is imperfect, it was the only classification possible. The main goal of this work is to predict illnesses by detecting related behaviors. To this end, we use sensor data to train several classifiers to detect abnormal behaviors. While training/testing a classifier with data from only one resident (apartment) provides consistent results [94], it is not a realistic approach since the algorithm must produce consistent results based on previous training data when a new sensor network is deployed. There are two major challenges associated with our training approach: the dataset is strongly imbalanced, as there are not many abnormal events, and the distributions of the training data may vary greatly by resident due to differences in behaviors and diseases. To overcome these problems, we investigate a solution based on transfer learning. We compare our solution based on transfer learning with three other methods, namely, a regular support vector machine (SVM) [95] and two one-class classifiers, i.e., support vector domain data description (SVDD) and k-nearest neighbors data description (KNNDD) [96].

Traditional supervised machine learning techniques assume that the training data and test data are drawn from datasets with similar probability distributions and that the classification task is the same for both datasets [97]. However, in practice, it is often necessary to relax this assumption and consider the test information to have another probability distribution or to permit the classification task to change. In this

situation, traditional machine learning techniques often fail to classify the test data. Based on the motion and bed sensor data from a particular resident of TigerPlace, a model can be learned to classify the activity occurring in the apartment. Then, this model can be tested with a different resident in a different apartment or with different activity labels. If the model is not modified for the new situation, then the prediction accuracy will decrease significantly. Transfer learning techniques have been proposed to handle these types of cases. With transfer learning, the knowledge learned previously can be applied to obtain a better or faster solution. Compared with traditional machine learning algorithms, transfer learning has many benefits: it reduces the time it takes to learn new tasks, it requires little expert information, and it improves the robustness of the learned model [97].

5.2 Methodology

Transfer learning can be defined in many ways, but in general, through transfer learning, the training procedure is considered to be enhanced by using information other than that from the source dataset. In each transfer learning algorithm, a source task is related to the source domain, and the target task is linked to the target domain. The transfer learning process has two main steps: source task training and knowledge transfer [12, 98]. In regular linear SVM, the label of a data vector x is determined by the sign of a linear decision function, *i.e.*, $y = sgn(f(x)) = sgn(x^T w)$, where $w = [w_i]_i^M = 1$ are the model parameters. A linear SVM classifier is trained by considering the following optimization problem:

$$\min_{w,b} \|w\|^2 + C \sum_{i=1}^{N} max(0, 1 - y_i(w^T x_i + b)) \tag{5.1}$$

where w is perpendicular to the separating hyperplane and the scoring function

that predicts the label of the test sample z_i is $f(z_i) = sign(w^T Z_i)$. $\|w\|^2$ encourages margin maximization, and term hinge loss $\sum_{i=1}^{N} max(0, 1 - y_i(w^T x_i + b)))$ determines the location of the hyperplane by minimizing the misclassification error. The hyper-parameter C balances margin maximization and hinge loss.

Transfer learning support vector machine (SVM) is a model based on transfer regularization formulations with adaptive SVM [12,99] that was introduced for adapting SVM classifiers to new domains. The key idea is to learn from the source model w^s by regularizing the distance between the learned model w and w^s. The classifier is linear, and it is indicated by a format vector w, with a scoring function $w^T x$, where x is the feature vector. The task is to learn w for the target group using a few training instances x_i, and the source group detector w^s. The formulation is as follows:

$$L_A = \min_{w,b} \|w - \Gamma w^s\|^2 + C \sum_{i=1}^{N} max(0, 1 - y_i(w^T x_i + b)) \tag{5.2}$$

where Γ controls the measure of transfer regularization, C controls the weight of the loss function, and N the number of samples. The main difference in this formulation is the use of squared loss instead of hinge loss [100, 101]; squared loss is more sensitive to outliers, but it also provides an analytic least squares solution for the objective. Furthermore, it enables efficient leave-one-out cross-validation, which is used to optimize the hyperparameter Γ, i.e., the amount of transfer. Intuitive transfer regularization for an SVM is similar to a spring between Γw^s and w, and it is equivalent to providing training samples from the source class. The transfer can also be understood by expanding the regularization term. Assume that w^s is l2 normalized to 1; then

$$\|w - \Gamma w^s\|^2 = \|w\|^2 - 2\Gamma\|w\| \cos\theta + \Gamma^2 \tag{5.3}$$

where $\|w\|^2$ provides the normal SVM margin maximization, and $-2\Gamma\|w\|\cos\theta$ induces the transfer by maximizing $\cos\theta$, *i.e.,* by minimizing the angle $\cos\theta$ between w^s and w [102]. However, the term $-2\Gamma\|w\|\cos\theta$ also encourages $\|w\|$ to be larger, which reduces the cost and prevents margin maximization. Thus, Γ, which should define the amount of transfer regularization, balances margin maximization and knowledge transfer.

The support vector domain description algorithm by [103] is used in this work for comparison with transfer learning SVM. SVDD can be used for outlier recognition. A spherically shaped decision boundary around an arrangement of objects is built based on a set of support vectors describing the sphere boundary; this boundary can change the information to new feature spaces without significantly increasing the computational burden. For a dataset with N data objects, $x_i, i = 1, \ldots, N$, a description is required. We try to find the smallest sphere that contains as many of the data objects as possible. This problem is extremely sensitive to the most obvious outlier object in the target dataset. When one or a few remote objects are in the training set, a huge sphere is obtained that will not characterize the data very well. Therefore, we allow for some data points outside the sphere and the introduce slack variable ζ (analogous to [103]). The formulation below is used to maximize α_i:

$$L = \sum_i \alpha_i(x_i.x_i) - \sum_{i,j} \alpha_i\alpha_i(x_i, x_i) \tag{5.4}$$

with constraints $0 \leq \alpha_i \leq C, \sum_i \alpha_i = 1$, where the items in α_i are the Lagrange multipliers. The inner products of objects (x_i, x_j) can be substituted by a kernel function $K(x_i, x_j)$ when this kernel $K(x_i$, one-class) satisfies Mercer's theorem, which implicitly maps the objects xi into some feature space. When a suitable feature space is chosen, a better, tighter description can be acquired. No explicit mapping is required; the problem is expressed entirely in terms of $K(x_i, x_j)$. Therefore, all inner

products (x_i, x_j) are substituted by a proper $K(x_i, x_j)$, and the problem of finding a data domain description is now given by

$$L = \sum_i \alpha_i K(x_i.x_i) - \sum_{i,j} \alpha_i \alpha_i (x_i, x_j) \tag{5.5}$$

with constraints $0 \leq \alpha_i \leq C and \sum_i \alpha_i = 1$. A test object z is accepted when

$$K(z, z) - 2\sum_i \alpha_i K(z.x_i) + \sum_{i,j} \alpha_i \alpha_j K(x_i, x_j) \leq R^2 \tag{5.6}$$

Different kernel functions K result in different description boundaries in the original input space. In this work, radial basis function (RBF) kernel is used. The third method that we used in this work for comparison to our transfer learning SVM approach is the k-nearest neighbors data description (KNNDD) [96], which is a one-class classifier version of kNN. In this simplified version of kNN, only the distance to the kth nearest neighbor is utilized. In the one-class classifier KNNDD, a test object z is accepted when its local density is larger or equal to the local density of its (first) nearest neighbor in the training set $NN^{tr}(z) = NN_1^{tr}(z)$. For local density estimation, $k = 1$ is used [96]:

$$f_{NN^{tr}}(z) = NN^{tr}(z) = I\Big(\frac{\|z - NN^{tr}(z)\|}{V\|NN^{tr}(z) - NN^{tr}(NN^{tr}(z))\|} \leq 1\Big) \tag{5.7}$$

Thus, the distance from object z to its nearest neighbor in the training set $NN^{tr}(z)$ is compared with the distance from this nearest neighbor $NN^{tr}(z)$ to its nearest neighbor.

5.3 Datasets

5.3.1 Synthetic dataset

For the synthetic dataset, we chose the case in which the target dataset has a different distribution than the source dataset from which we want to transfer knowledge. To simulate this scenario, we generated two imbalanced datasets: a source (or training) dataset and a target (or testing) dataset. To generate data, we randomly sampled 500 and 200 source points from simple 2D Gaussian distributions. The first class had a mean of (2, 3), the second had a mean of (6, 3), and both had a covariance matrix of ((1, 1.5), (1.5, 3)). For the target dataset, we generated 500 and 50 imbalanced data points for the first and second class, respectively. The first class had a mean of (-10, 0), the second had a mean of (-9, 0) and both had a covariance matrix of ((0.05 ,0.1), (0.1 ,8)), as shown in Fig. 5.1.

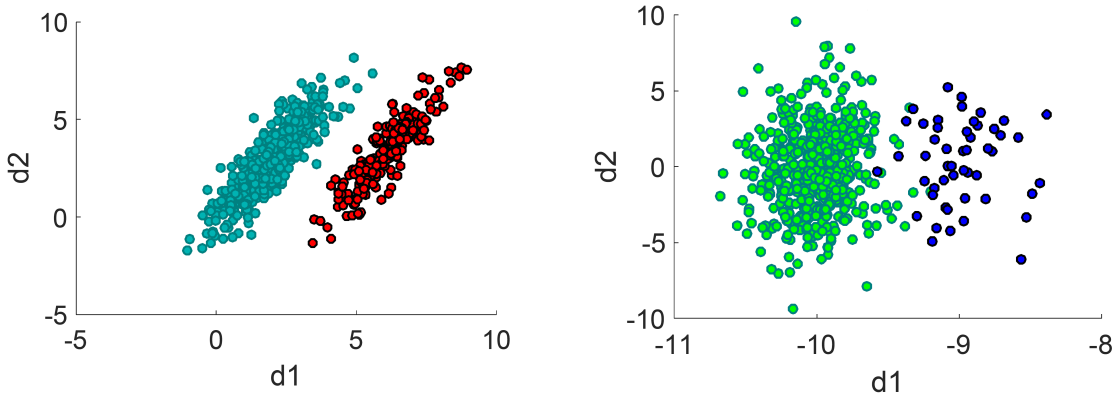


Figure 5.1: Synthetic dataset: (a) Source data and (b) Target data

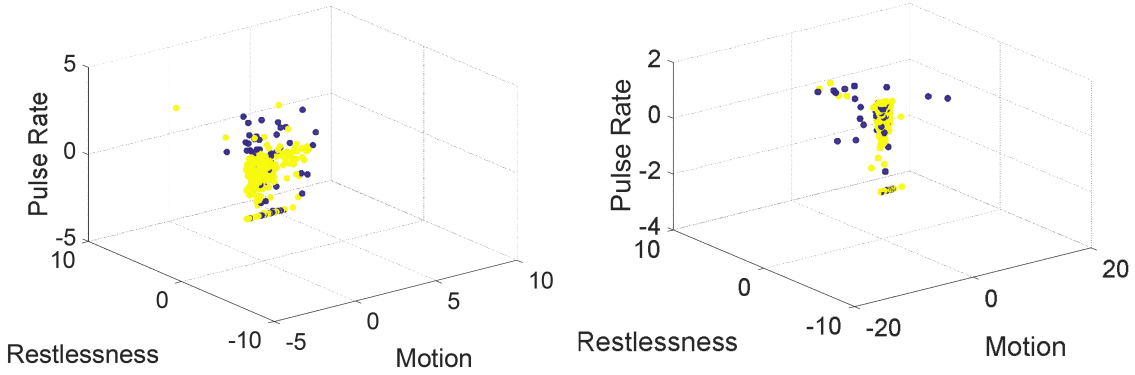
5.3.2 Real-world dataset

Sensor data of three residents from TigerPlace are considered in this work, as shown in Table 5.1. In addition to the sensor data, we have all the clinical record data

(medications, nursing visits, hospitalizations, etc.) for these residents. Each day was manually labeled by the authors as good or bad based on the nursing visit reports and other clinical records, and these labels served as the ground-truth information. Fig. 5.2 shows a three-dimensional (3D) feature visualization of the data of Resident1.

Table 5.1: Data for three residents in tiger place.

Resident No.	**Total records**	**Positive days (feel good)**	**Negative days (feel bad)**
Resdident1	441	360	81
Resdident2	744	709	35
Resdident3	499	164	335



(a) (b)

Figure 5.2: Visualization of three features for normalized data of Resident1: (a) Day data and (b) Night data.

5.4 System Architecture and Implementation

TigerPlace [92,93] is an independent living facility for seniors planned and created as part of a joint effort between the Sinclair School of Nursing, University of Missouri, and Americare Systems Inc. of Sikeston, Missouri. An essential objective of TigerPlace is to help the inhabitants to handle their sicknesses and to remain as healthy and independent as possible. Each resident included in the study has a data logger

in his or her apartment that collects data from wireless sensors. The data logger logs the data into a document that is sent to a database on a protected server via a wired network connection. The sensor network comprises of a few types of sensors that are mounted in various places throughout the residents' apartments, including counting motion sensors, bed sensors, and stove temperature sensors. The motion sensors are set in multiple spots, such as the bedroom, bathroom, kitchen, and living room, and some residents have this sort of sensor installed on the refrigerator, kitchen cabinets, and drawers.

These sensors capture resident movement over the apartment by producing a signal when there movement is sensed. The bed sensors include certain sets of sensors, such as a pneumatic sensor strip (on the bed) and a motion sensor (attached to the headboard). The sensor strip and motion sensor joined are connected, and they work in a similar way as the previously mentioned motion sensors: they identify action. Unlike the motion sensors, the bed sensor strip records three types of activities, which are organized based on three or four levels of severity. Early illness recognition assumes that, if the resident does not feel well, then his/her sleep and motion patterns will change. In this study, we used four features to represent resident behavior: the total number of motion records, bed restlessness, pulse rate, and breath, respectively, for every hour of the day preceding the nursing report (produced daily at 12 pm). The number of features was doubled by splitting the data into day and night sets.

Data processing was performed as follows. First, we aggregated the sensor data: features 1–4 were the sum of the sensor data during the night (7 pm–7 am), while features 5–8 were the sum of the sensor data during the day (7 am–7 pm) and represent the sensor activity prior a nursing visit. Then, the data are normalized for the three residents; after that, the information is passed through the transfer learning SVM algorithm described in Section 5.2. In this case, the user is prompted to control the amount of knowledge to be transferred and the weight of the loss function by changing

Γ and C, respectively. The other three algorithms described in Section 5.2 are applied for comparison with our main algorithm.

5.5 Experimental Results

5.5.1 Experiments on the synthetic dataset

The main goal of this work was to determine the practicability of using transfer learning SVM for early illness recognition based on sensor data by using training data from another resident. First, we apply the transfer learning SVM algorithm for our synthetic data, which is imbalanced (*i.e.*, the source and target data have different distributions). The ROC obtained by classifying each target dataset is shown in Fig. 5.3. We see that the transfer learning SVM approach outscored the other three classifiers. The AUC was 0.998 for transfer learning SVM, while it was 0.915, 0.988, and 0.994 for regular SVM with an RBF kernel, SVDD, and KNNDD, respectively.

5.5.2 Experiments on the real dataset

For the real-world dataset, we run a set of tests to determine the best percentage of data that we need to transfer from the target to the source. Table 5.2 shows that the AUC increases as the amount of knowledge increases; it stops increasing after reaching 10% bad day data. The ROC obtained by classifying resident3 after training on resident1 using the transfer learning SVM algorithm is given in Fig. 5.4. For comparison, we also show the ROC curves obtained for the same data with regular SVM with an RBF kernel, SVDD, and KNNDD. Transfer learning outperforms the other three methods. The AUC for transfer learning SVM was 0.765, while it was 0.52, 0.44 and 0.55 for regular SVM with an RBF kernel, SVDD, and KNNDD, respectively. Table 5.3 illustrates the rest of the combinations by considering the data of specific

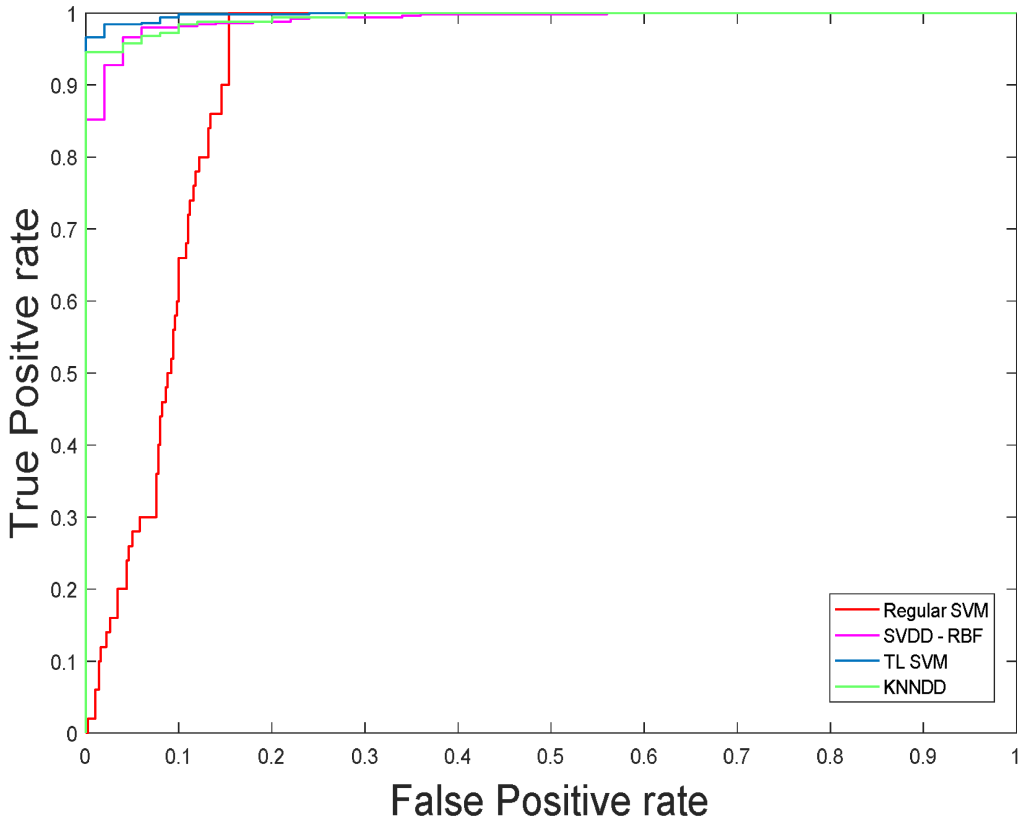


Figure 5.3: ROC curves for transfer learning SVM comparing regular SVM with an RBF kernel, SVDD, and KNNDD run on synthetic data.

residents as the training data and testing on the data of another resident. The average overall training/test combinations are shown for all methods in the last line of Table 5.3. The best standard was obtained for the transfer learning SVM algorithm.

Table 5.2: Amount of transfer knowledge versus AUC.

Bad Day knowledge transfer	**AUC**
1 (3%)	0.7648
2 (6%)	0.7646
3 (10%)	0.7647
7 (20%)	0.7646
11 (30%)	0.7647
14 (40%)	0.7647
17 (50%)	0.7647
21 (60%)	0.7647

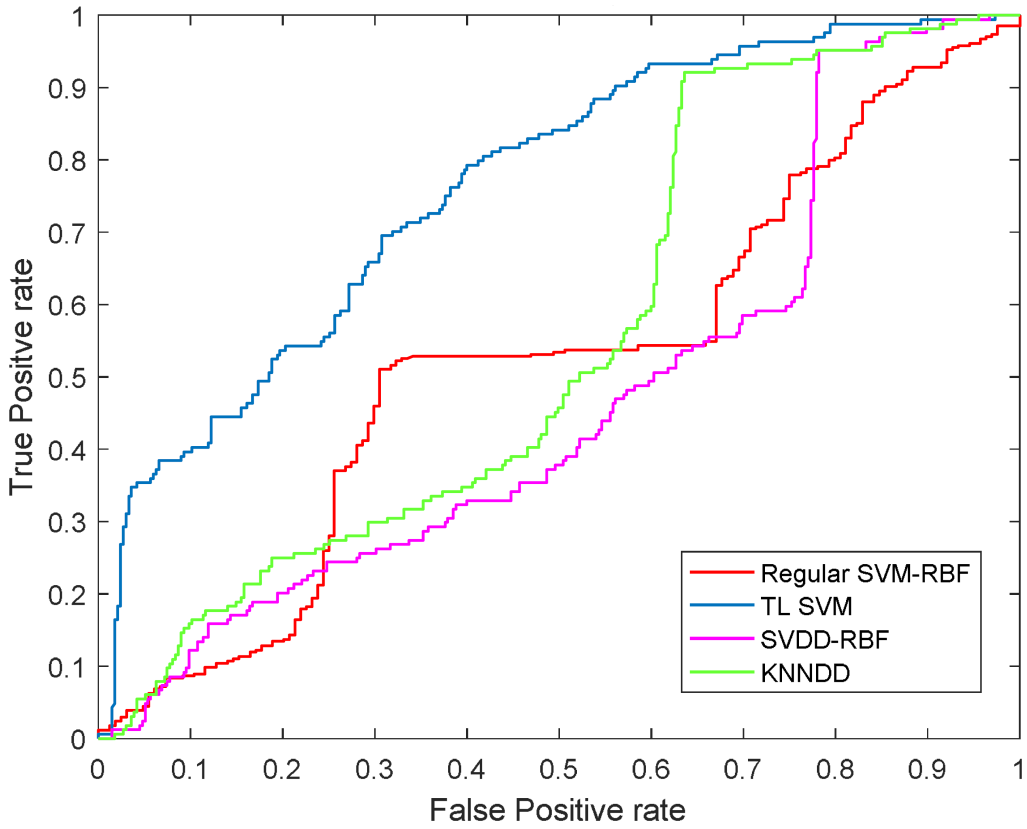


Figure 5.4: ROC curves for transfer learning SVM comparing regular SVM with an RBF kernel, SVDD, and KNNDD on the sensor data of Resident3 after training on the data of Resident1.

Table 5.3: AUC results for the all combination residents.

Resident Training → Test	**AUC Transfer Learning SVM**	**AUC Regular SVM**	**AUC SVDD**	**AUC KNNDD**
Res1 → Res3	**0.765**	0.52	0.44	0.55
Res2 → Res3	**0.799**	0.52	0.43	0.48
Res1 → Res2	**0.708**	0.64	0.70	0.71
Res3 → Res1	**0.653**	0.39	0.71	0.72
Res3 → Res2	**0.838**	0.45	0.71	0.63
Res2 → Res1	**0.66**	0.61	0.64	0.65
Average	***0.737***	*0.52*	*0.60*	*0.62*

5.6 Conclusion

Using three resident case studies, we showed that the presented early illness recognition method based on transfer learning performs much better than individual resident learning, even when very few target instances are available. The performance of the transfer approach improves when additional target instances are added, but it will stop improving after a certain point. Since data labeling is difficult in real elder care settings such as TigerPlace, transfer learning could help to train illness recognition classifiers for various residents. Using transfer learning SVM, we could detect the illnesses of elderly residents of TigerPlace based on unobtrusive monitoring. The recognition of early signs of a disease might help nursing staff provide interventions that can prevent grave clinical events such as heart attacks or strokes.

Chapter 6

Conclusion and Future Directions

This final chapter summarizes the work in this dissertation by drawing together the hypotheses and discussing the critical outcomes of the research, and presents several possibilities for future work.

6.1 Summary

In this study, a novel clustering framework called the transfer-learning possibilistic c-means (TLPCM) clustering algorithm has been developed based on the original PCM algorithm and inspired by transfer learning. TLPCM works in applications where data are limited and not sufficient for useful clustering or are polluted by unknown noise or outliers. Hence, to evaluate the methodology presented, the first hypothesis is initially constructed where minimal unlabeled data is available in the target task, data in the form of a transfer learning process from contextually related, but differing source tasks can be used to learn predictive tasks. The hypothesis was tested through a series of experiments on both synthetic and real-world data. It

could achieve substantially better clustering results than PCM with even a small percentage of the dataset. Thus, the main principle of transfer learning is fulfilled since our algorithm does not need an abundance of data to cluster correctly. The proposed method surpassed the PCM algorithm for individual learning in terms of consistency and the volume of data needed for clustering.

Additionally, a new deep learning-based hierarchical classification method for automatic sleep stage classification based on BCG data with the help of transfer learning has been developed. The second hypothesis in this work is created to evaluate our model; where a very few labeled instances are assumed to be available in the target domain and sufficiently labeled as being similar, but not the same data which is available in the source task. Mainly, a transfer learning technique is used to train our proposed model with sleep posture data (source data) and use the extracted features to classify a few sleep stages subjects (target data). The CNNs and LSTMs are utilized to build the proposed architecture. While CNNs are trained to learn filters that extract time-invariant features from BCG signals, the LSTMs are trained to encode temporal information, such as sleep stages transition rules. The LOSO-CV strategy is used in this hypothesis, which showed potential in automatically classifying sleep stages epochs. The proposed architecture achieved a significantly improved performance in contrast to similar studies. Consequently, with the application of transfer learning, deep learning model tested was superior to traditional techniques. The results of our proposed model showed potential in classifying the sleep stages, and the features extracted from the BCG posture signals contain characteristics associated with sleep stages.

Moreover, the same hypothesis is deployed with a transfer learning support vector machine approach. First, I applied the transfer learning SVM algorithm for the synthetic data, which is imbalanced (*i.e.*, the source and target data have different distributions). Then, the method is validated using real-world data by conducting

a retrospective study on three residents from TigerPlace, a retirement community in Columbia, MO, where apartments are fitted with wireless networks of motion and bed sensors. The transfer learning SVM approach outperformed three other methods, *i.e.*, regular SVM, one-class SVM, and one-class kNN, and the average areas under the curve (AUCs) for the four methods based on the data of three residents.

6.2 Future work

We plan to continue our research on the following potential topics:

1. In our transfer learning algorithms, we assumed that the presence of a subset of relevant features distributed by all tasks. Nevertheless, in some practical applications, this assumption does not always hold. There might be clusters of tasks that share only a few futures among them. Finding these clusters, it might enable us to share knowledge more efficiently across the tasks.

2. Although the PCM and TLPCM have a unique feature compared to most clustering algorithms, which allows the algorithm to find only C dense regions out of K actual regions based on the typicalities while the rest of the regions have insignificant typicalities towards the existing C clusters. However, this mode of seeking property still faces parameter selection problems and initialization problems. Thus, for future work, we will revise our model to make it more robust and less sensitive to the initialization.

3. In this work, a deep learning-based classification method for automatic sleep stage classification based on BCG data is developed. The results of the leave-one-out cross-validation strategy showed potential in automatically classifying sleep stage epochs. However, some other designs of deep learning architecture can be used and compare the performance with the current one to produce

better results for the REM stage.

4. The bed sensor signal is particularly sensitive to movement. This caused much noise overlapping with the original signal. The BCG signals were not accurate and clean enough compared with ECG signal. A noise removal method needs to be applied before using BCG signals.

5. Another problem is that the subjects in this study were elderly; many had a history of sleep disorders and medical complications. Thus collecting more data for healthy people in the future will be useful for further investigation.

6. This dissertation investigates single cross-domain transfer learning, future direction to transfer the common knowledge among multiple domains and tasks.

Bibliography

[1] Hua Zuo, Guangquan Zhang, Witold Pedrycz, Vahid Behbood, and Jie Lu. Fuzzy regression transfer learning in takagi–sugeno fuzzy models. *IEEE Transactions on Fuzzy Systems*, 25(6):1795–1807, 2017.

[2] Christopher Olah. Understanding lstm networks. 2015.

[3] David N Perkins and Gavriel Salomon. Transfer of learning. international encyclopedia of education. *Oxford: Pergamon Press. Retrieved April*, 14:2019, 1992.

[4] Sarah Leberman and Lex McDonald. *The transfer of learning: Participants' perspectives of adult education and training*. Routledge, 2016.

[5] Anne McKeough, Judy Lee Lupart, and Anthony Marini. *Teaching for transfer: Fostering generalization in learning*. Routledge, 2013.

[6] Stephen M Cormier and Joseph D Hagman. *Transfer of learning: Contemporary research and applications*. Academic Press, 2014.

[7] Jethro Shell and Simon Coupland. Fuzzy transfer learning: methodology and application. *Information Sciences*, 293:59–79, 2015.

[8] Derek Hao Hu and Qiang Yang. Transfer learning for activity recognition via sensor mapping. In *Twenty-second international joint conference on artificial intelligence*, 2011.

[9] Jethro Shell, Stephen Vickers, Simon Coupland, and Howell Istance. Towards dynamic accessibility through soft gaze gesture recognition. In *2012 12th UK Workshop on Computational Intelligence (UKCI)*, pages 1–8. IEEE, 2012.

[10] Ariadna Quattoni, Michael Collins, and Trevor Darrell. Transfer learning for image classification with sparse prototype representations. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008.

[11] Manu Sharma, Michael P Holmes, Juan Carlos Santamaría, Arya Irani, Charles Lee Isbell Jr, and Ashwin Ram. Transfer learning in real-time strategy games using hybrid cbr/rl. In *IJCAI*, volume 7, pages 1041–1046, 2007.

[12] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[13] Lisa Torrey, Jude Shavlik, et al. Transfer learning. handbook of research on machine learning applications and trends: algorithms, methods, and techniques. *Information Science Reference*, page 22, 2009.

[14] Rayan Gargees, James M Keller, and Mihail Popescu. Tlpcm: Transfer learning possibilistic c-means. *IEEE Transactions on Fuzzy Systems*, 2020.

[15] Rayan Gargees, James M Keller, Mihail Popescu, and Marjorie Skubic. Non-invasive classification of sleep stages with a hydraulic bed sensor using deep learning. In *International Conference on Smart Homes and Health Telematics*, pages 73–82. Springer, 2019.

[16] Rayan Gargees, James Keller, and Mihail Popescu. Early illness recognition in older adults using transfer learning. In *2017 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 1012–1016. IEEE, 2017.

[17] Santosh Kumar. Transfer learning for dataset shift in classification and clustering problems. 2017.

[18] Ievgen Redko and Younès Bennani. Kernel alignment for unsupervised transfer learning. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 525–530. IEEE, 2016.

[19] Xiaojin Zhu and Andrew B Goldberg. Introduction to semi-supervised learning. *Synthesis lectures on artificial intelligence and machine learning*, 3(1):1–130, 2009.

[20] Richard Marcum. Application of deep convolutional neural networks to automatic feature detection in high resolution remote sensing imagery. 2017.

[21] Jacob M Williams. Deep learning and transfer learning in the classification of eeg signals. 2017.

[22] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[23] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[24] George E Dahl, Tara N Sainath, and Geoffrey E Hinton. Improving deep neural networks for lvcsr using rectified linear units and dropout. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 8609–8613. IEEE, 2013.

[25] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[26] Kevin Jarrett, Koray Kavukcuoglu, Marc'Aurelio Ranzato, and Yann LeCun. What is the best multi-stage architecture for object recognition? In *2009 IEEE 12th international conference on computer vision*, pages 2146–2153. IEEE, 2009.

[27] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[28] Shouwei Sun, Yizhang Jiang, and Pengjiang Qian. Transfer learning based maximum entropy clustering. In *2014 4th IEEE International Conference on Information Science and Technology*, pages 829–832. IEEE, 2014.

[29] Wei Sun and Qian Xu. A transfer learning algorithm for document categorization based on clustering. In *2012 International Conference on Computer Science and Electronics Engineering*, volume 2, pages 528–531. IEEE, 2012.

[30] Quanquan Gu and Jie Zhou. Learning the shared subspace for multi-task clustering and transductive transfer classification. In *2009 Ninth IEEE International Conference on Data Mining*, pages 159–168. IEEE, 2009.

[31] Pengjiang Qian, Yizhang Jiang, Zhaohong Deng, Lingzhi Hu, Shouwei Sun, Shitong Wang, and Raymond F Muzic. Cluster prototypes and fuzzy memberships jointly leveraged cross-domain maximum entropy clustering. *IEEE transactions on cybernetics*, 46(1):181–193, 2015.

[32] S Ephina Thendral and C Valliyammai. Clustering based transfer learning in cross domain recommender system. In *2016 Eighth International Conference on Advanced Computing (ICoAC)*, pages 51–54. IEEE, 2017.

[33] Hua Zuo, Jie Lu, Guangquan Zhang, and Feng Liu. Fuzzy transfer learning using an infinite gaussian mixture model and active learning. *IEEE Transactions on Fuzzy Systems*, 27(2):291–303, 2019.

[34] Wenyuan Dai, Gui-Rong Xue, Qiang Yang, and Yong Yu. Transferring naive bayes classifiers for text classification. In *AAAI*, volume 7, pages 540–545, 2007.

[35] Sinno Jialin Pan, James T Kwok, Qiang Yang, and Jeffrey Junfeng Pan. Adaptive localization in a dynamic wifi environment through multi-view learning. In *AAAI*, volume 7, pages 1108–1113, 2007.

[36] Eduardo Pinheiro, Octavian Postolache, and Pedro Girão. Theory and developments in an unobtrusive cardiovascular system representation: ballistocardiography. *The open biomedical engineering journal*, 4:201, 2010.

[37] Omer T Inan, Pierre-Francois Migeotte, Kwang-Suk Park, Mozziyar Etemadi, Kouhyar Tavakolian, Ramon Casanella, John Zanetti, Jens Tank, Irina Funtova, G Kim Prisk, et al. Ballistocardiography and seismocardiography: A review of recent advances. *IEEE journal of biomedical and health informatics*, 19(4):1414–1427, 2015.

[38] Octavian A Postolache, Pedro MB Silva Girao, Joaquim Mendes, Eduardo C Pinheiro, and Gabriela Postolache. Physiological parameters measurement based on wheelchair embedded sensors and advanced signal processing. *IEEE Transactions on instrumentation and measurement*, 59(10):2564–2574, 2010.

[39] Licet Rosales, Marjorie Skubic, David Heise, Michael J Devaney, and Mark Schaumburg. Heartbeat detection from a hydraulic bed sensor using a clustering approach. In *2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 2383–2387. IEEE, 2012.

[40] Chang-Sei Kim, Stephanie L Ober, M Sean McMurtry, Barry A Finegan, Omer T Inan, Ramakrishna Mukkamala, and Jin-Oh Hahn. Ballistocardiogram: Mechanism and potential for unobtrusive cardiovascular health monitoring. *Scientific reports*, 6:31297, 2016.

[41] Feng Wang, Yanhui Zou, Mami Tanaka, Tadashi Matsuda, and Seiji Chonan. Unconstrained cardiorespiratory monitor for premature infants. *International Journal of Applied Electromagnetics and Mechanics*, 25(1-4):469–475, 2007.

[42] Xu Wang, Fangfang Jiang, Dan Yang, and Yuan Liao. Estimation of the respiratory component from ballistocardiography signal using adaptive interference cancellation. In *2011 Chinese Control and Decision Conference (CCDC)*, pages 571–574. IEEE, 2011.

[43] Arie Oksenberg, Aida Dynia, Khitam Nasser, and Natan Gadoth. Obstructive sleep apnoea in adults: body postures and weight changes interactions. *Journal of sleep research*, 21(4):402–409, 2012.

[44] R Gardner. Normal motor patterns in sleep in man. *Advances in sleep research*, 2:67–107, 1975.

[45] Mak Adam Daulatzai, Neela Khan, Chandan Karmakar, Ahsan Khandoker, and Marimuthu Palaniswami. Lateral decubitus posture during sleep: Sub-groups of obstructive sleep apnea patients—therapeutic value of vertical position in osa. In *Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2009 5th International Conference on*, pages 181–184. IEEE, 2009.

[46] Meng-Hsi Wu, Emily J Chang, and Tzu-Hsuan Chu. Personalizing a generic ecg heartbeat classification for arrhythmia detection: A deep learning approach. In *2018 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*. IEEE, 2018.

[47] Alex Graves. Supervised sequence labelling. In *Supervised sequence labelling with recurrent neural networks*, pages 5–13. Springer, 2012.

[48] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[49] Jen Hong Tan, Yuki Hagiwara, Winnie Pang, Ivy Lim, Shu Lih Oh, Muhammad Adam, Ru San Tan, Ming Chen, and U Rajendra Acharya. Application of stacked convolutional and long short-term memory network for accurate identification of cad ecg signals. *Computers in biology and medicine*, 94:19–26, 2018.

[50] James C Bezdek, Robert Ehrlich, and William Full. Fcm: The fuzzy c-means clustering algorithm. *Computers & Geosciences*, 10(2-3):191–203, 1984.

[51] Raghuram Krishnapuram and James M Keller. A possibilistic approach to clustering. *IEEE transactions on fuzzy systems*, 1(2):98–110, 1993.

[52] Pengjiang Qian, Yizhang Jiang, Zhaohong Deng, Lingzhi Hu, Shouwei Sun, Shitong Wang, and Raymond F Muzic. Cluster prototypes and fuzzy memberships jointly leveraged cross-domain maximum entropy clustering. *IEEE transactions on cybernetics*, 46(1):181–193, 2016.

[53] Inderjit S Dhillon, Subramanyam Mallela, and Dharmendra S Modha. Information-theoretic co-clustering. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 89–98. ACM, 2003.

[54] Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

[55] Xiaojin Zhu, Zoubin Ghahramani, and John D Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th International conference on Machine learning (ICML-03)*, pages 912–919, 2003.

[56] Sinno Jialin Pan, Qiang Yang, et al. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

[57] Zhaohong Deng, Yizhang Jiang, Fu-Lai Chung, Hisao Ishibuchi, Kup-Sze Choi, and Shitong Wang. Transfer prototype-based fuzzy clustering. *IEEE transactions on fuzzy systems*, 24(5):1210–1232, 2015.

[58] Paul A Heidenreich, Justin G Trogdon, Olga A Khavjou, Javed Butler, Kathleen Dracup, Michael D Ezekowitz, Eric Andrew Finkelstein, Yuling Hong, S Claiborne Johnston, Amit Khera, et al. Forecasting the future of cardiovascular disease in the united states: a policy statement from the american heart association. *Circulation*, 123(8):933–944, 2011.

[59] Bo Yu Su, Moein Enayati, KC Ho, Marjorie Skubic, Laurel Despins, James Keller, Mihail Popescu, Giovanna Guidoboni, and Marilyn Rantz. Monitoring the relative blood pressure using a hydraulic bed sensor system. *IEEE Transactions on Biomedical Engineering*, 66(3):740–748, 2019.

[60] David Heise, Licet Rosales, Mary Sheahen, Bo-Yu Su, and Marjorie Skubic. Non-invasive measurement of heartbeat with a hydraulic bed sensor progress, challenges, and opportunities. In *2013 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, pages 397–402. IEEE, 2013.

[61] Licet Rosales, Bo Yu Su, Marjorie Skubic, and KC Ho. Heart rate monitoring using hydraulic bed sensor ballistocardiogram 1. *Journal of Ambient Intelligence and Smart Environments*, 9(2):193–207, 2017.

[62] Katy Lydon, Bo Yu Su, Licet Rosales, Moein Enayati, KC Ho, Marilyn Rantz, and Marjorie Skubic. Robust heartbeat detection from in-home ballistocardiogram signals of older adults using a bed sensor. In *2015 37th Annual Inter-*

national Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), pages 7175–7179. IEEE, 2015.

[63] Ayan Acharya, Eduardo R Hruschka, Joydeep Ghosh, and Sreangsu Acharyya. Transfer learning with cluster ensembles. In *Proceedings of the 2011 International Conference on Unsupervised and Transfer Learning workshop-Volume 27*, pages 123–133. JMLR. org, 2011.

[64] Lilyana Mihalkova, Tuyen Huynh, and Raymond J Mooney. Mapping and revising markov logic networks for transfer learning. In *Aaai*, volume 7, pages 608–614, 2007.

[65] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

[66] JP Marques De Sa. *Pattern recognition: concepts, methods and applications*. Springer Science & Business Media, 2012.

[67] Raghuram Krishnapuram and James M Keller. The possibilistic c-means algorithm: insights and recommendations. *IEEE transactions on Fuzzy Systems*, 4(3):385–393, 1996.

[68] Patrick M Murphy. Uci repository of machine learning databases. *ftp:/pub/machine-learning-databaseonics. uci. edu*, 1994.

[69] Thierry Denoeux. A neural network classifier based on dempster-shafer theory. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 30(2):131–150, 2000.

[70] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[71] Fatih Kayaalp, Muhammet Sinan Basarslan, and Kemal Polat. A hybrid classification example in describing chronic kidney disease. In *2018 Electric Electronics, Computer Science, Biomedical Engineerings' Meeting (EBBT)*, pages 1–4. IEEE, 2018.

[72] Yedilkhan Amirgaliyev, Shahriar Shamiluulu, and Azamat Serek. Analysis of chronic kidney disease dataset by applying machine learning methods. In *2018 IEEE 12th International Conference on Application of Information and Communication Technologies (AICT)*, pages 1–4. IEEE, 2018.

[73] Abdullah Al Imran, Md Nur Amin, and Fatema Tuj Johora. Classification of chronic kidney disease using logistic regression, feedforward neural network and wide & deep learning. In *2018 International Conference on Innovation in Engineering and Technology (ICIET)*, pages 1–6. IEEE, 2018.

[74] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.

[75] Endang Purnama Giri, Aniati Murni Arymurthy, Mohammad Ivan Fanany, and Sastra Kusuma Wijaya. Sleep stages classification using shallow classifiers. In *2015 International Conference on Advanced Computer Science and Information Systems (ICACSIS)*, pages 297–301. IEEE, 2015.

[76] Ibrahim Sadek, Jit Biswas, and Bessam Abdulrazak. Ballistocardiogram signal processing: a review. *Health information science and systems*, 7(1):10, 2019.

[77] M Dursun, S Gunes, S Ozsen, and S Yosunkaya. Comparison of artificial immune clustering with fuzzy c-means clustering in the sleep stage classification problem. In *2012 International Symposium on Innovations in Intelligent Systems and Applications*, pages 1–4. IEEE, 2012.

[78] Jarno Tuominen, Karoliina Peltola, Tarja Saaresranta, and Katja Valli. Sleep parameter assessment accuracy of a consumer home sleep monitoring ballistocardiograph beddit sleep tracker: A validation study. *Journal of Clinical Sleep Medicine*, 15(03):483–487, 2019.

[79] Albert Vilamala, Kristoffer H Madsen, and Lars K Hansen. Deep convolutional neural networks for interpretable analysis of eeg sleep stage scoring. In *2017 IEEE 27th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE, 2017.

[80] Tara N Sainath, Oriol Vinyals, Andrew Senior, and Haşim Sak. Convolutional, long short-term memory, fully connected deep neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 4580–4584. IEEE, 2015.

[81] Haşim Sak, Andrew Senior, and Françoise Beaufays. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *Fifteenth annual conference of the international speech communication association*, 2014.

[82] Jasmin Kevric and Abdulhamit Subasi. Comparison of signal decomposition methods in classification of eeg signals for motor-imagery bci system. *Biomedical Signal Processing and Control*, 31:398–406, 2017.

[83] Sabine Schmidt, Georg Eich, Sylviane Hanquinet, Heinz Tschäppeler, Peter Waibel, and François Gudinchet. Extra-osseous involvement of langerhans' cell histiocytosis in children. *Pediatric radiology*, 34(4):313–321, 2004.

[84] Richard B Berry, Rita Brooks, Charlene E Gamaldo, Susan M Harding, Carole L Marcus, Bradley V Vaughn, et al. The aasm manual for the scoring of sleep

and associated events. *Rules, Terminology and Technical Specifications, Darien, Illinois, American Academy of Sleep Medicine*, 176, 2012.

[85] Jialei Yang, James M Keller, Mihail Popescu, and Marjorie Skubic. Sleep stage recognition using respiration signal. In *2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 2843–2846. IEEE, 2016.

[86] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37, pages 448–456, 2015.

[87] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[88] Moein Enayati, Marjorie Skubic, James M Keller, Mihail Popescu, and Nasibeh Zanjirani Farahani. Sleep posture classification using bed sensor data and neural networks. In *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 461–465. IEEE, 2018.

[89] Ruhan Yi, Moein Enayati, James Keller, Mihail Popescu, and Marjorie Skubic. Non-invasive in-home sleep stage classification using a ballistocardiography bed sensor. In *2019 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. IEEE, 2019.

[90] Rifat Shahriyar, Md Faizul Bari, Gourab Kundu, Sheikh Iqbal Ahamed, and Md Mostofa Akbar. Intelligent mobile health monitoring system (imhms). In *International Conference on Electronic Healthcare*, pages 5–12. Springer, 2009.

[91] Jim Rowan and Elizabeth D Mynatt. Digital family portrait field trial: Support for aging in place. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 521–530, 2005.

[92] Marilyn J Rantz, Karen Dorman Marek, Myra A Aud, Rebecca A Johnson, Donna Otto, and Rose Porter. Tigerplace: A new future for older adults. *Journal of nursing care quality*, 20(1):1–4, 2005.

[93] Marjorie Skubic, Gregory Alexander, Mihail Popescu, Marilyn Rantz, and James Keller. A smart home application to eldercare: Current status and lessons learned. *Technology and Health Care*, 17(3):183–201, 2009.

[94] M Popescu and A Mahnot. Early illness recognition using in-home monitoring sensors and multiple instance learning. *Methods of information in medicine*, 51(04):359–367, 2012.

[95] Vladimir Vapnik. *The nature of statistical learning theory*. Springer science & business media, 2013.

[96] David Martinus Johannes Tax. One-class classification: Concept learning in the absence of counter-examples. 2002.

[97] Kyle D Feuz and Diane J Cook. Transfer learning across feature-rich heterogeneous feature spaces via feature-space remapping (fsr). *ACM Transactions on Intelligent Systems and Technology (TIST)*, 6(1):1–27, 2015.

[98] Akshay Jain, James M Keller, and James C Bezdek. Quantitative and qualitative comparison of periodic sensor data. In *2016 IEEE-embs international conference on biomedical and health informatics (bhi)*, pages 37–40. IEEE, 2016.

[99] Botond Bocsi, Lehel Csató, and Jan Peters. Alignment-based transfer learning for robot models. In *The 2013 international joint conference on neural networks (IJCNN)*, pages 1–7. IEEE, 2013.

[100] Jun Yang, Rong Yan, and Alexander G Hauptmann. Adapting svm classifiers to data with shifted distributions. In *Seventh IEEE International Conference on Data Mining Workshops (ICDMW 2007)*, pages 69–76. IEEE, 2007.

[101] Tatiana Tommasi and Barbara Caputo. The more you know, the less you learn: from knowledge transfer to one-shot learning of object categories. In *BMVC*, number CONF, 2009.

[102] Yusuf Aytar and Andrew Zisserman. Tabula rasa: Model transfer for object category detection. In *2011 international conference on computer vision*, pages 2252–2259. IEEE, 2011.

[103] David MJ Tax and Robert PW Duin. Support vector domain description. *Pattern recognition letters*, 20(11-13):1191–1199, 1999.

VITA

Rayan Gargees received a master's degree in Electrical and Computer Engineering from the University of Missouri, USA, in 2018. Additionally, he has served as a graduate teaching assistant at the University of Missouri. The results of his works are published in top journals such as IEEE Transaction on Fuzzy Systems. Additionally, he has attended and presented his papers at scientific conferences, such as the IEEE BIBM, 2017 and ICOST 2019. Besides, he authored various papers that have been published in other avenues. He also serves as a reviewer in leading journals and conferences such as the springer cluster computing journal. His current research interests include machine learning, computational intelligence, transfer learning, fuzzy systems, image processing, computer vision, bioinformatics, sensor networks, and deep learning.