Gaussian Transformation Enhanced Semi-Supervised Learning for Sleep Stage Classification

by

Yifan Guo

Bachelor of Engineering, Jiangsu University, 2016

Submitted to the Graduate Faculty of

the Swanson School of Engineering in partial fulfillment

of the requirements for the degree of

Master of Science

University of Pittsburgh

2022

UNIVERSITY OF PITTSBURGH

SWANSON SCHOOL OF ENGINEERING

This thesis was presented

by

Yifan Guo

It was defended on

November 4, 2022

and approved by

Azime Can-Cimino, Ph.D., Assistant Professor, Department of Electrical and Computer Engineering

Ahmed Hassan Sayed Dallal, Ph.D., Assistant Professor, Department of Electrical and Computer Engineering

Thesis Advisor: Zhi-Hong Mao, Ph.D., Professor, Department of Electrical and Computer Engineering

Gaussian Transformation Enhanced Semi-Supervised Learning for Sleep Stage Classification

Yifan Guo, M.S.

University of Pittsburgh, 2022

Sleep disorders are torturing big populations in modern society. To provide efficient help to people with sleep difficulties, accurate sleep monitoring is essential. However, clinical examinations have problems of limited measure durations and biased observations and thus are often insufficient for diagnosis of sleep disorders. Electroencephalogram (EEG) based wearable devices are promising compliments for clinical examinations because of their ability for convenient and reliable long-term monitoring of sleep. This thesis develops both supervised and semi-supervised learning approaches for sleep stage classification, which can be applied on EEG wearable devices. Specifically, we design and implement various EEG based sleep stage classifiers with different feature extraction processes and then compare and analyze these classifiers through experiments. The classifiers are able to obtain satisfactory performance on the test data from a limited number of human subjects, but the learned classifiers usually cannot generalize well on previously unseen human subjects because the EEG signal characteristics vary significantly from person to person. To address this issue, we propose a semi-supervised learning algorithm to mitigate the performance deterioration when dealing with new subjects. We further study and evaluate several Gaussian transformations on EEG band power features to improve the robustness and accuracy of the algorithm.

Table of Contents

List of Tables

List of Figures

Preface

I feel truly grateful for Prof.Mao's wise and kind help to my research, my academic journey, and my whole life. I want to thank Prof.Yin, Prof.Dallal, and Prof.Can for their brilliant and warm suggestions during my work. I also would like to thank my parents for always been supportive and my cats for their meow.

1.0 Introduction

1.1 Motivation

Sleep is a crucial activity for all human being. Healthy sleep is important for both physical and mental health [45]. Unfortunately, sleep disorders have become very common, being observed in 30% to 50% of general population [40]. To provide proper treatments to patients suffering from sleep disorders, it is necessary collect information from patients' sleep rhythms. One of the most common ways to extract valuable information from the rhythms is sleep stage classification. According to the recent American Academy of Sleep Medicine (AASM) scoring manual [8], sleep can be divided into non rapid eye movement (NREM) and rapid eye movement (REM) sleep, and NREM sleep can be subdivided into N1, N2, and N3 stages. Commonly used clinical examinations includes overnight polysomnography (PSG), EEG and video-EEG monitoring, and multiple sleep latency tests (MSLT), etc [12].

Though these examinations are powerful, several major drawbacks should be noted. The first one is their limited measure time scale. Standard EEG test lasts for 20-30 minutes [13]. Overnight PSG test monitors patient's physiological indicators for one or two nights. Their short observing duration limits their ability to detect long-term sleep patterns. Long-term sleep monitoring sometimes relies on patient's self sleep estimation, which is subjective and noisy. The second disadvantage of current clinical sleep examinations is biased observation. Because most sleep monitoring procedures are conducted in sleep centers and/or require the patients to wear sensors on their fingers, chests, and wrists, the data are collected when the patients sleep in unusual environments. Given most participants are suffering from sleep disorders, strange environments or wearing alien devices can significantly influence their sleep process, which makes the collected data very biased. These drawbacks make long-term sleep measurement in the ambulatory setting a valuable complement to laboratory's results.

So far, only actigraphy has been widely accepted by the clinical community as a method for objective, long-term sleep monitoring. Actigraphy refers to measure sleep movements via FDA-approved, wrist-worn accelerometry [56]. This kind of devices are expensive and have

evident limits that it only measure movement data. As a result, it is difficult to be used for fine-grand sleep stage classification. On the other hand, even though consumer wearable devices have not formally accepted as an assistance for clinical diagnosis, they have been widely purchased and are showing promising usability [56].

The two biggest obstacles that prevent current consumer wearable devices from being formally accepted by the clinical community are 1) commercially used algorithms are confidential thus cannot be peer reviewed, and 2) they typically base on the less accurate technique—microelectromechanical systems (MEMS) accelerometers and photoplethysmography (PPG) [56, 23]. One promising alternative is EEG based technology because EEG signal contents richest information related to sleep [23]. Hence, with proper designs, EEG based devices could evidently outperform PPG based ones.

Deep learning is a rapidly developing technology which is powerful to dig out highly abstract features and patterns from rich data sources like EEG signals. The performance of deep models improves tremendously compared with that of traditional machine learning approaches and even surpasses human experts in some fields [6]. However, the state-of-the-art deep models usually require enormous amount of labeled data for training. For some specific fields, data labeling could be utterly time-consuming and expensive [58]. In sleep researches, for example, human experts often need to label patients' whole night electroencephalogram (EEG) signal data for every thirty seconds slice [8]. Public EEG data sets typically contain data from no more than a hundred subjects. Since EEG signal's distributions could vary significantly from one subject to another, it is challenging for classifiers to maintain high performances on previously unseen subjects. The need of additional training and validation data for new human subjects and the huge workload to label such data make automatic label generating strategies appealing.

1.2 Contributions

To address the above mentioned challenges, we propose a cluster-then-label strategy, which follows the spirit of semi-supervised learning and synergistically integrates a cluster-

ing procedure in a supervised learning pipeline. The classifier is then retrained with the generated pseudo-labels. Experiments show that this strategy can evidently improve the classifier's performance on data from out-of-distribution human subjects. The procedure is deployed to various combinations of clustering algorithms and classifiers. Furthermore, to overcome the intrinsic bottleneck of the proposed cluster-then-label algorithm, i.e., errors in pseudo-labels, we investigate Gaussian transformations for EEG band power features. With improved normality statistics (meaning that the probability distributions of the features after the Gaussian transformation are closer to Gaussian distributions), the strategy can obtain higher accuracy in both clustering and classification process.

1.3 Thesis Organization

Chapter II provides an introduction on prerequisite knowledge. In Chapter III we survey though related works. The proposed methods and their experimental results are contained in Chapter IV and Chapter V, respectively. Chapter VI concludes this thesis and talks about future works.

2.0 Prerequisites

2.1 Machine Learning

Machine learning is a general term which describes all the techniques that can enable a computer to learn from experience [38]. The word "learning" indicates that the designer of an ML algorithm do not need to explicitly specify the steps of problem solving. In the view of learning mechanism, machine learning can be divided into four classes: supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning.

Supervised learning entails learning a mapping between a set of of input variables $X = \{x_1, x_2, \ldots, x_n\}, X \in \mathcal{R}^{n \times d}$ and an output variable $Y \in \mathcal{R}^n$ and requires that such mapping is also feasible for unseen data. The learned models are evaluated on unseen data. When the scale of the labeled data set is too small to reflect the true data distribution, the generalization performance can be poor [42].

Unsupervised learning, on the other hand, exploits certain properties directly from unlabeled data $\{x_1, x_2, \ldots, x_n\} \in \mathcal{R}^{n \times d}$. Since unlabeled data are often easier to acquire, less data deficiency problems appear in this field. However, because the data come without labels, unsupervised learning can hardly satisfy some typical real world missions such as classification and regression. It is also harder to find a clear way to evaluate the performance of learned models [42].

Semi-supervised learning typically uses partially labeled data for training process. There exist some smart strategies in this field that combine the merits of both supervised and unsupervised learning so that the model can learn from small labeled data set and also take advantage of large unlabeled data set [67, 42].

Reinforcement learning aims to learn a map from situations to actions so that the cumulative reward is maximized. The learner is not supervised by the "correct" action in a certain situation but have to explore which action should be taken [53]. It is also different from unsupervised learning since it still want to learn a map. Reinforcement learning is wildly used in robotics, economy, human-computer interaction, etc.

2.2 Sleep Stages and EEG Band Power

According to the recent American Academy of Sleep Medicine (AASM) scoring manual [8], sleep can be divided into non rapid eye movement (NREM) and rapid eye movement (REM) sleep, and NREM sleep can be subdivided into N1, N2, and N3 stages. When used for diagnosis, physicians prefer using EEG than other physiological measurements such as electromyography (EMG) and electrooculography (EOG) because of EEG signal's richness of sleep information [14].

EEG signals are highly complex time serial signals and are usually organized into thirty second's slices. It requires demanding expertise to directly interpret EEG signals. Because EEG signals are often contaminated by device's noise and of high dimensionality, it usually requires a feature extraction step before EEG signals can be used for training ML models. EEG band power features are one of the most intuitive and natural feature sets for EEG signals. An EEG slice can be represented by a 4-D vector, where each dimension stands for the total power distributed into a certain frequency band range (delta: 1-4 Hz, theta: 4-8 Hz, alpha: 8-12 Hz, and beta: 12-30 Hz).

3.0 Related Works

3.1 Sleep Stage Classification

Sleep stage classification is an important research problem, and researchers have developed numerous approaches including supervised, unsupervised, and semi-supervised learning methods.

As classification is a traditional supervised learning problem, supervised learning becomes the most exploited field for sleep stage classification. Researchers in this field usually focus on improving the structure of classifiers and the performance of feature extractions. Support vector machines (SVM) [14, 66, 2] and artificial neural networks [25, 51, 52, 26] are two most commonly used classifiers. Other attempts include linear discriminate analysis (LDA) [16], bagged tree [61], etc. From feature usage point of view, wavelet is a popular method for feature extraction [16, 25, 2]. Arranging the input of a classifier in a form of graph is also an inspiring way to discover the patterns hidden in the signals [26, 66, 14]. Some deep models directly use raw signals to realize end-to-end approaches [26, 52].These approaches can attain accurate results, but often rely on large sets of labeled data for training and might not generalize well when dealing with new subjects.

Compared with supervised learning approaches, unsupervised learning methods are relatively infrequent to be used for sleep stage classification. This is mainly because unsupervised approaches do not take advantage of the information from labels and thus usually require extra steps before generating predictions. One representative work in this field is done by Rodríguez-Sotelo et al., who extract entropy features from multi-channel EEG signals, analyze feature relevance with the Q-α algorithm, and partition the data with the J-means clustering algorithm [47].

Semi-supervised learning is a promising technique for sleep stage classification, because it utilizes both labeled and unlabeled data. Munk et al. add unlabeled data part into their maximum likelihood estimation (MLE) cost function and propose their own form of conditional probability of unlabeled data [39]. Wuzheng et al. improve sparse concentration

index to evaluate pseudo-labels' confidence [59], and Bai and Lu use small fully labeled data to pre-train the classifier and feed the generated pseudo-labels back to the model for training [5]. Transfer learning has also been explored for sleep stage classification. For example, Zhao et al. add domain classifiers to basic convolutional neural networks (CNN) to learn domain information from different levels [62]. Jadhav et al. pre-train a CNN on ImageNet data set, extract time-frequency features from raw EEG data with continuous wavelet transform (CWT), and retrain the network on these features [24]. Other transfer learning researches include [1, 4, 44].

3.2 Domain Generalization

Traditional ML models are trained based on the identically and independently distributed assumption (i.i.d) that training data and test data are independently sampled from the same distribution. This assumption fails to hold in many scenarios especially in bio-medical fields like sleep studies, where data sets' quantity and/or diversity are usually limited. And because data collecting is very expensive and expertise intense in these fields, solving the out-of-distribution problem by simply gathering more data is prohibitively impossible. Various approaches are proposed in domain generalization (DG) to enhance the model's generalization ability when the test domain's distribution is different yet related to the training domain. These approaches can be categorized into three classes: data manipulation, representation learning, and learning strategy Please find detailed survey in [57] and [64].

3.2.1 Data Manipulation

Data manipulation methods enrich the training set by manipulating existing data points. Follow this track, there are two popular strategies: data augmentation and data generation.

Data augmentation distorts initial data set with various operations including adding noise, flipping, rotation, etc. It is a general strategy for improving model's robustness and not limited to DG. Being required to handle distorted data, the model is forced to capture general

features of different domains. One special data augmentation method is called adversarial augmentation [49, 55, 65]. Specific noises are designed forcing misclassifications to appear for the current model. By explicitly overcoming its current weakness, the model generalize better.

Data generation based DG strengthens model's generalization capability by generating diverse *new* data points. Unlike data augmentation, which manipulates original data, data generation first trains a generative model using current data set then produces new data with the generative model. Popular generative techniques include variational auto-encoder (VAE) [30], generative adversarial networks (GAN) [19], and Mixup [60].

3.2.2 Representation Learning

Representation learning conceptually decomposes a prediction function into two part—feature extractor and executor (e.g., a classifier). Regularization is imposed implicitly or explicitly so that the output of feature extractor have certain properties. Two major subcategories of representation learning are domain-invariant representation learning and feature disentanglement.

Domain-invariant representation based DG is built on the theory that domain invariant features are general and transferable to different domains. Kernel based methods is one of the most popular representation learning methods. Kernel based methods project original data points into high dimensional features and avoid computational burden with kernel tricks [9, 20, 21]. Many methods are also proposed with the idea of domain adversarial learning [33, 17, 34] and explicit feature alignment [63, 27, 41]. The former uses adversarial learning to reduce domain discrepancy in manifold space and the latter uses explicit distribution alignments or feature normalization to align the feature distributions across domains.

3.2.3 Learning Strategy

There are plenty number of learning strategies can be used for DG directly or with minimum modification. They can be categorized into three classes: ensemble learning based DG, meta learning based DG and others.

Ensemble learning is built upon an assumption that any input is a weighted superposition of existing training domains. Thus, the final predictions can be obtained by assembling multiple models. Mancini et al. use a domain predictor to generate weights for results from domain-specific predictors then yield the final predication as a weighted sum [37]. Segu et al. propose a method that compute the weights according to the distance between the test sample's batch norm statistics and those of each training domain. The classifiers share parameters except for batch normalization parameters.

Meta learning technique is also referred as "learning to learn", which inducts a general model from multiple sources. Li et al. stimulate distribution variations by randomly divide source domains into meta-train and meta-test domains at each training iteration [31]. Balaji et al. parameterize the regularization term with a separate neural network. This regularizer is trained with meta learning so that it can enable generalization through domains [7]. Other studies in this category include [35, 15, 11].

There are also *other learning strategy* that can be adopted to DG and the proposed method in this thesis belongs to this category. Carlucci et al, propose a self-supervised method that learns general representations by solving jiasaw puzzles [10]. Li et al. train the feature extractor and the classifier using episodic training [32]. Self-challenging mechanism is used in [22] to iterativly abandon domain-specific features.

3.3 Gaussian Transformations for EEG Signal

Because we observe random failures in experiments using both real EEG data and artificial data, we want to use theoretically simplest algorithms in the our procedures so that we can get insights and intuitions of these failures. Gaussian distribution is an ubiquitous distribution which many real world data subject to. Many ML algorithms are also designed upon the Gaussian assumption. However, EEG band power features do not subject to Gaussian distributions by nature. In [18], the authors compare the performance of various fixed transformations like $\sqrt{x}$, $\log(x)$, and $\log(x/(1-x))$, which can symmetrize skew distributions. The x's are either absolute value of EEG band power or relative band power ratios.

Boyd and Lacher propose a two-step transformation procedure for clinical data. The first step removes the skewness and the second step handles kurtosis. All these works transform data in a complete open loop manner. In other words, their transformations are designed only with statistical a prior knowledge without any feedback from the transformed results. In this thesis, we design various data-driven Gaussian transformations (some of them are in a close loop manner) that are helpful to the proposed cluster-then-label strategy.

4.0 Proposed Methods

4.1 Problem Formulation

Denote the a raw EEG signal data set as $S_0 = \{(s_1, y_1), ..., (s_N, y_N)\}$ where s_i's are raw EEG signals and y_i's are according sleep stages. The set $S = \{(x_1, y_1), ..., (x_N, y_N)\}$ is the same set replacing the raw EEG signals with extracted features. The feature extraction process is represented as $F(.)$, serving as $S = F(S_0)$. The clustering algorithm is denoted as G, which takes S as input and generate a bunch of clusters (groups) $\{g_1, ..., g_K\}$. The classifier is denoted as C, which receives a feature vector x and predicts the according sleep stage. Importantly, the classifier's performance is evaluated on a independent data set S_{test} in which the features distribute differently compared with S.

4.2 Cluster-Then-Label Algorithm

The overall pipeline of our proposed cluster-then-label strategy is shown in fig. 1. Blue nodes stand for data, where X and Y are EEG signal features and labels, respectively. Note that no label is given except for the initial data set. X_0 and Y_0 are from S while X_i, $i > 0$ are from S_{test}. Yellow triangles denote classifiers, and green rectangles represent clustering and training processes. We start with a fully labeled yet relatively small data set and train a classifier C_0 on it. Instead of directly generating pseudo labels by the classifier, we use the clustering process to capture extra geometrical information thus correct the labels of the points which would have otherwise been mis-classified. Assume we can easily obtain large amount of unlabeled data, we feed these data into a clustering model—Gaussian mixture model (GMM) or k-means in our experiment—and get K clusters, where the number K is given. Using the classifier trained on previous data, each cluster is given a uniform label which corresponds to the dominating class in that cluster. Then these new data with pseudo labels can be used to train the classifier again. Such process can be repeated as long as new

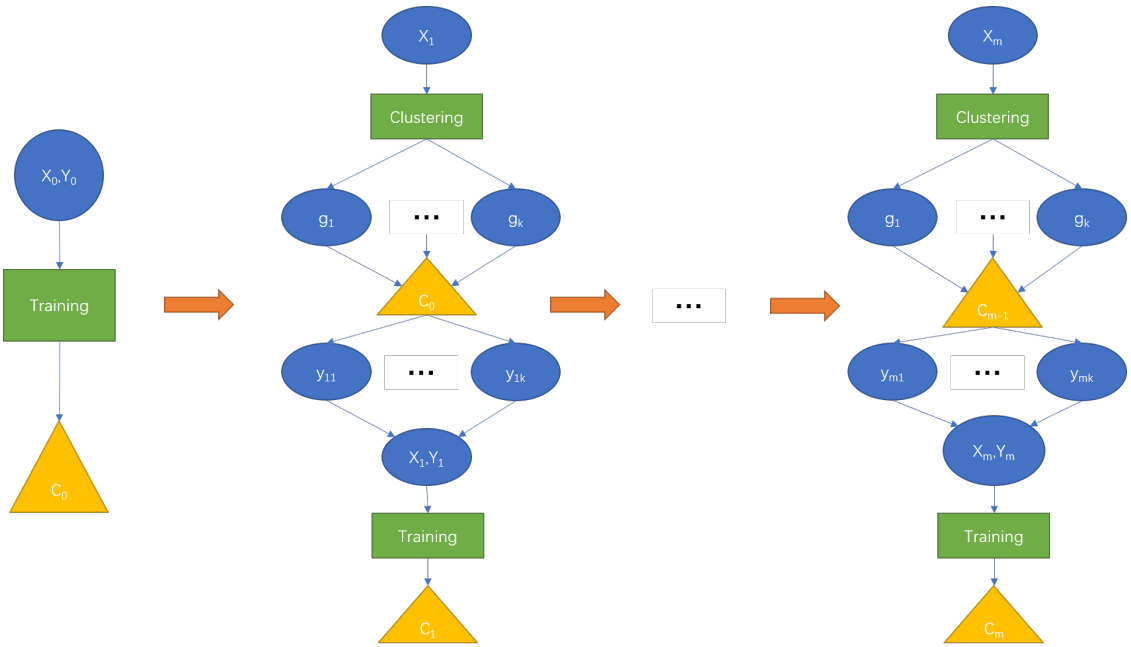

Figure 1: Cluster-then-label algorithm pipeline.

unlabeled data are available. The feature extraction process $F(.)$ is not shown explicitly in this figure because sometimes it is a separate process and in other cases is a part of the classifier. To sum up, the classifier C is initially trained on fully labeled S. Then it is retrained using pseudo-labeled samples from S_{test}.

4.3 Affects of Pre-Trained Classifier's Accuracy

In this section, we analyze the relationship between the classifier's accuracy and the probability that a cluster is assigned with correct label (i.e., the label of its major class).

In binary classification scenario, without lose of generality, assume that the majority class of a cluster has the label "1". Let n_0 and n_1 denote the number of negative and positive samples in the cluster, respectively. The probability of this cluster finally been given a pseudo-label that consistent with the ground truth label of the majority class in this cluster

can be expressed as $P(\hat{n}_1 > \frac{n}{2})$ where n is the size of this cluster and $\hat{n}_1$ is the number of data points that are classified to be positive. This is equivalent to the union probability of $P(\hat{n}_1 = k), k > \frac{n}{2}$. If we assume that the classifier follows $E_{S_{test}}(P(\hat{Y} = Y)) = \beta$, then $P(\hat{n}_1 = k)$ can be expressed as follows: when $0 \leq k \leq n_0$,

$$P(\hat{n}_1 = k) = \sum_{j=0}^{k} \binom{n_1}{k-j} a^{k-j}(1-a)^{(n_1-k+j)} \times \binom{n_0}{j} a^{n_0-j}(1-a)^j \tag{4-1}$$

when $n_0 \leq k \leq n_1$,

$$P(\hat{n}_1 = k) = \sum_{j=0}^{n_0} \binom{n_1}{k-j} a^{k-j}(1-a)^{(n_1-k+j)} \times \binom{n_0}{j} a^{n_0-j}(1-a)^j \tag{4-2}$$

when $n_1 \leq k \leq n$,

$$P(\hat{n}_1 = k) = \sum_{j=0}^{n_0} \binom{n_1}{j} a^{n_1-j}(1-a)^j \times \binom{n_0}{n-k-j} a^j(1-a)^{n_0-j}. \tag{4-3}$$

The curves of above equations are shown in fig 2. All curves rise up smoothly and different colors denote different classification accuracy.

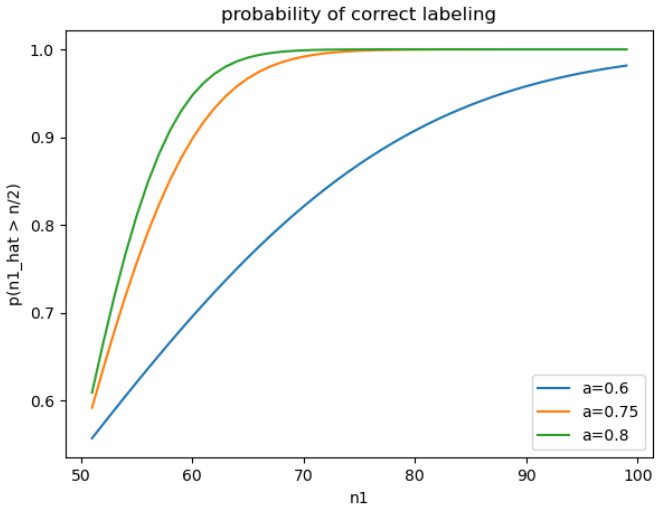

Figure 2: The curves of probabilities of correctly labeling a cluster.

4.4 k-Means Clustering and TinySleepNet Classifier

The idea of cluster-then-label is firstly instantiated by combining kmeans clustering and TinySleepNet classifier [52].

4.4.1 Clustering with k-Means

As the most commonly used clustering algorithm, k-means algorithm separates n data points of D dimensions into k non-overlapping groups so that the total within-cluster variation (TWCV) is minimized. Formally speaking, given a data set $X = \{x_1, x_2, ..., x_n\}^T$ and the number of desired clusters K, k-means algorithm forms a indicator matrix W whose element defined as follows

$$w_{ki} = \begin{cases} 1, & \text{if } x_i \text{ belongs to } k\text{th cluster} \\ 0, & \text{otherwise} \end{cases} \tag{4-4}$$

and

$$\sum_{k=1}^{K} w_{ki} = 1. \tag{4-5}$$

The centroid of the kth cluster c_k is defined as

$$c_k = \frac{\sum_{i=1}^{n} w_{ki} x_{i:}}{\sum_{i=1}^{n} w_{ki}}. \tag{4-6}$$

And the TWCV is defined as

$$\sum_{k=1}^{K} \sum_{i=1}^{n} w_{ki} ||x_i - c_k||_2. \tag{4-7}$$

To minimize TWCV, classic k-means algorithm starts with randomly chosen K centroids and assigns each data point to the closest centroid. Then the centroids are updated according to (4-6). Such process is repeated until no reassignment occur.

Though usually being simple and fast in practice, k-means algorithm relies on proper initialization and may suffer from local minimum.

4.4.2 TinySleepNet

TinySleepNet is a composed with a convolutional neural network (CNN) and a recurrent neural network (RNN) as shown in fig 3. The CNN part serves as a feature extractor and the RNN part is for capturing temporal dependencies. The outputs of feature extractor (CNN) can be view as non-normalized probabilities of five sleep stages. They do not subject to Gaussian distributions by nature and are not suitable to be modified into Gaussians. That is the reason for choosing k-means over GMM here.

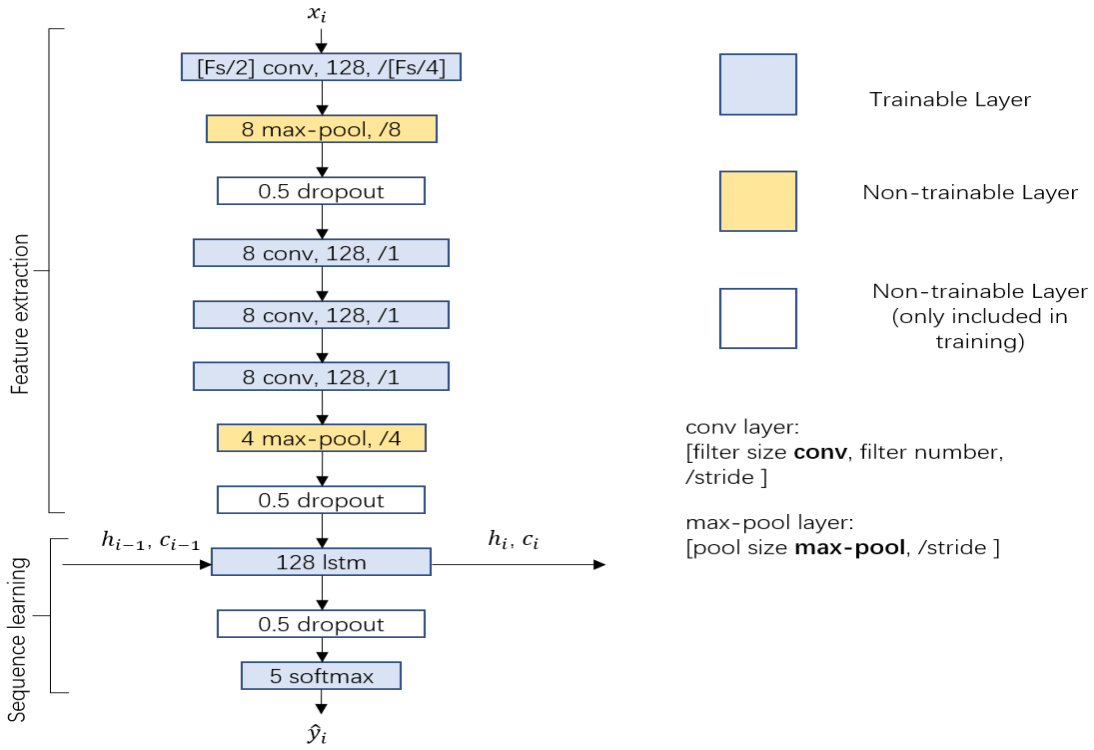

Figure 3: The structure of TinySleepNet.

4.5 GMM Clustering and LDA Classifier

In this section, we introduce required techniques for GMM+LDA cluster-then-label. Again, choosing these simplest models is not for highly accurate classification results but

for insights of cluster-then-label process. Very importantly, since GMM and LDA rely on prior probabilities of each class, the number of data points for each sleep stage in training set need to be carefully chosen. This is because sleep stages have sequential dependencies on each other which means their prior probabilities vary with time. As a default, we keep the number of samples of each sleep stage the same in our experiments.

4.5.1 Linear Discriminate Analysis

LDA is a kind of Bayes's classifier. It assumes the distribution of each class subjects to a multivariate Gaussian distribution. In binary classification scenario, assume $class0$ and $class1$have probability density functions $f_0 = N(\mu_0, \Sigma)$ and $f_1 = N(\mu_1, \Sigma)$, respectively. Note that both density functions have the same covariance matrix. The form where two density functions have different covariance matrices is referred as quadratic LDA. LDA decides the belonging of a test point $\mathbf{x}$ by calculating:

$$L(x) = \pi_0 \times f_0 - \pi_1 \times f_1 \tag{4-8}$$

where π_0 and π_1 are the prior probabilities of $class0$ and $class1$, respectively. Inserting the definition of multivariate Gaussian distribution density function with a logarithm trick, we get

$$L(x) = \mathbf{a}^T\mathbf{x} + b \tag{4-9}$$

where

$$\mathbf{a} = \Sigma_x^{-1}(\mu_0 - \mu_1) \tag{4-10}$$

$$b = -\frac{1}{2}(\mu_0^T\Sigma_x^{-1}\mu_0 - \mu_1^T\Sigma_x^{-1}\mu_1) + \log\pi_0 - \log\pi_1. \tag{4-11}$$

In practice, μ_0, μ_1, and Σ are unknown. We usually estimate these values with statistics of training samples:

$$\mu_i = \frac{1}{n_i}\sum_{j=1}^{n_i} x_{ij}, i = 0, 1 \tag{4-12}$$

$$\Sigma_x = \frac{1}{n}(\sum_{j=1}^{n_0}(x_{0j} - \mu_0)(x_{0j} - \mu_0)^T + \sum_{j=1}^{n_1}(x_{1j} - \mu_1)(x_{1j} - \mu_1)^T). \tag{4-13}$$

When $L(x) \geq 0$, x is assigned to $class0$, and to $class1$ otherwise. This discrimination rule can be easily adopted to multiply class discrimination scenario by treating such scenario as several one-to-one classification problems.

4.5.2 Clustering with Gaussian Mixture Model

A Gaussian mixture model, as shown below, is a parametric probability function representing weighted sum of multiple Gaussian distributions [46],

$$p(x|\lambda) = \sum_{i=1}^{M} w_i g(x|\mu_i, \Sigma_i) \tag{4-14}$$

$$\sum_{i} w_i = 1 \tag{4-15}$$

where x is a D-dimensional continuous data vector, M is the number of components included in GMM model, w_i refers to the mixture weight for the ith component, g stands for the Gaussian density function of the form:

$$g(x|\mu_i, \Sigma) = \frac{1}{(2\pi)^{D/2}|\Sigma_i|^{1/2}} e^{\frac{1}{2}(x-\mu_i)^T \Sigma_i^{-1}(x-\mu_i)} \tag{4-16}$$

where μ_i and Σ_i are the mean value and covariance matrix, respectively. λ is the total set of all the parameters of GMM, i.e.,

$$\lambda = \{\mu_i, \Sigma_i, w_i\}. \tag{4-17}$$

Such model is usually trained with expectation-maximization (EM) algorithm, an iterative algorithm that alternately updates the guess of generating distribution for each data point (E step) and the estimations of λ (M step)[48].

During test time, we assign a data point to the cluster which maximizes the posteriori probability. Formally, we assign a data point x to cluster c such that

$$\begin{aligned} c &= \arg\max_{c_i} p(c_i|x) \\ &= \arg\max_{c_i} \frac{p(x|c_i)p(c_i)}{\sum_j p(x|c_j)p(c_j)} \\ &= \arg\max_{c_i} \frac{g(x|c_i)w_i}{\sum_j g(x|c_j)w_j}. \end{aligned} \tag{4-18}$$

At this stage, we set the criteria to evaluate the performance of clustering process as

$$\alpha = \min \frac{|n_i'|}{|n_i|} \tag{4-19}$$

where $|n_i|$ stands for the size ith cluster produced by GMM model, $|n_i'|$ is the size of the majority class in ith cluster.

4.5.3 Multitaper Spectrogram

Because LDA and GMM cannot handle high dimensional temporal inputs like raw EEG signals, we need a separate feature extraction step. In this thesis, we use a technique called multitaper spectral analysis [43]. Spectral analysis is a classic tool of signal process. It extracts the frequency information of a signal. However, typical spectral analysis approaches, for example, fast Fourier transform (FFT), suffer from the side lobe leakages and the high variance of EEG signals, which result in very noisy and unclear spectra. Multitaper spectral analysis uses multiple specially designed tapes (or windows) to reduce the leakages and the variance by taking the average spectra. The tapes are called discrete prolate spheroidal sequences (DPSS). They are able to remove the false power from the side lobes and are orthogonal to each other. The feature extraction steps is formally described in algorithm 1. Note that N and M have different meaning from the notations in Section 4.1.

Algorithm 1 EEG Band Power Feature Extraction.

1: s is a raw EEG signal, T is the duration of the signal in seconds, df is the required frequency resolution, β, α, θ, and δ are band of interest as defined in Section 2.2.
2: The time-halfbandwidth product $TW \leftarrow \frac{T \times df}{2}$.
3: The number of tapers is $M \leftarrow \lfloor 2TW \rfloor - 1$.
4: Generate M DPSS tapers $\{t_1, ..., t_M\}$ according to TW, N, and M.
5: Separately multiply the EEG signal to each taper, get $S_t = \{s_1, ..., s_M\}$.
6: Apply FFT to each element in S_t, and calculate the average of the results.
7: Sum up S_t on β, α, θ, and δ get the four dimensional feature x of s.

4.6 Gaussian Transformations

In this chapter, we introduce two methods for EEG Gaussian transformation. Because a multivariate Gaussian is a combination of multiple 1-D Gaussian in each dimension, our research about Gaussian transformation focuses on scalar transformations, i.e., we transform one band at a time.

4.6.1 Assembled Fixed Transformation

In [18], the author reports the best result on resting EEG with the transformation $\log(\frac{x}{1-x})$, where x stands for the relative band power ratio. The curve of this transformation is shown in fig 4. This transformation works in a way that dilutes points in tails because the curve is steep in those areas. We follow this result in our research. The difference is that our data contain EEG signals from multiple stages. This means the band power features subject to a mixture distribution in our setting. So, we keep the basic shape of the transformation curve and apply a variational version of it to each stage. We take the combined curve as our final transformation. Algorithm 2 formally describes the transformation.

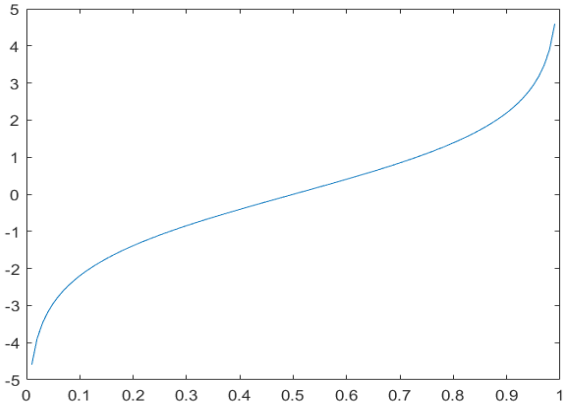

Figure 4: The curves of the basic transformation.

More intuitively speaking, we first select $100 \times r$ percent points in each sleep stage. Then the these data points are scaled into $(0, 1)$ and the original transformation is applied to them.

Algorithm 2 Manipulation of Basic Transformation.

Input: $S = \{(x_0, y_0), .., (x_N, y_N)\}$.

Define: R_{log} is the range of $\log(\frac{x}{1-x})$, r is the effective ratio.

Require: Sets should be treated as sequential data type in this algorithm.

for each sleep stage **do**

    $S_j = \{(x_i, y_i) | y_i =$ `current sleep stage`$\}$.

    Define L to be the length of S_j.

    $DownIdx \leftarrow L \times \frac{r}{2}$, $UpIdx \leftarrow L - DownIdx$

    $S_{sort} \leftarrow sort(S_j)$

    Define $Down \leftarrow$ the $DownIdx$'th element of S_{sort}, $Up \leftarrow$ the $UpIdx$'th element of S_{sort}

    $Range \leftarrow Up - Down$, $Mean \leftarrow$ the mean value of x_i's in S_j.

    Define $S_{changes} = \{x_i | Down < x_i < Up\}$.

    Element wise apply $S_{changes} \leftarrow \frac{S_{changes} - Down}{Range}$

    Element wise apply $S_{changes} \leftarrow (\log(\frac{S_{changes}}{1-S_{changes}})/R_{log}) \times Range + Mean$

end for

$S \leftarrow$ `element wise mean of all` S_j

Finally, they are re-scaled into $(Mean - 0.5Range, Mean + 0.5Range)$.

Although the assembled transformation brings better flexibility and is useful for our cluster-then-label algorithm, it also has some drawbacks. Firstly, the curve's basic shape is still fixed. This limits the overall flexibility. Secondly, at test time, a new point need to go through the transformation for every stage (because we do not know which stage it belongs to). As a result, the steep tail of one stage may intrude into other's flatten area leading to a groove around other's center. Finally, the designing process of the assembled transformation is still in a open loop manner.

4.6.2 Transformations Using Neural Networks

To address the limitations of assembled fixed transformation, we try to parameterize each stage's transformation with a neural network. We proposed three loss functions for training. Two related normality tests are mentioned below. Note that lower normality test statistics indicate better obedience to Gaussian distributions.

4.6.2.1 Training With Jarque–Bera (JB) Normality Test

JB test [54] is a kind of goodness-of-fit normality test. The statistic of JB test is defined as:

$$JB = \frac{n}{6}(S^2 + \frac{(K-3)^2}{4}) \tag{4-20}$$

where $S = \hat{\mu}_3/\hat{\mu}_2^{3/2}$ is the sample skewness, and $K = \hat{\mu}_4/\hat{\mu}_2^2$ is the sample kurtosis. The notation $\hat{\mu}_i = \frac{1}{n}\sum_{j=1}^{n}(x_j - \bar{x})$ is the estimation of ith order central moment where $\bar{x}$ is the mean value of x's. In our practice, we use modified JB statistic as our loss function:

$$L = \frac{n}{6}(\lambda_1 S^2 + \frac{(K-3)^2}{4}) + \lambda_2(\bar{X}_{in} - \bar{X}_{out})^2. \tag{4-21}$$

The hyper parameter λ_1 is to scale the magnitude of skewness in loss function because we empirically find the outputs tend to keep large skewness. The hyper parameter λ_2 controls the trade-off between output's normality and its mean value shifting since the mean value of the output distribution can shift dramatically without this term resulting in the loss of

biological meaning. JB loss is easy to be calculated. However, it estimates the goodness-of-fit using only skewness and kurtosis which means it may ignore certain types of abnormality.

4.6.2.2 Training with Anderson-Darling (AD) Test

AD test [3] is in general a more powerful normality test method compared with JB test. Its statistic is in the form:

$$A^2 = -n - \frac{1}{n}\sum_{i=1}^{n}[(2i-1)\log\Phi(Y_i) + (2(n-i)+1)(\log(1-\Phi(Y_i)))] \tag{4-22}$$

where Φ stands for standard Gaussian cumulative probability function (CDF). We form the final loss function with the same additional term $\lambda_2(\bar{X}_{in} - \bar{X}_{out})^2$ as in (4-21). Because we are using the CDF of standard Gaussian distribution, X_{out} need to be standardized and sorted after $\bar{X}_{out}$ is calculated.

4.6.2.3 Training with Kullback–Leibler (KL) Divergence

KL divergence [36] is a wildly used measure of how one distribution P is different from another distribution Q. The formal definition is $KL = E_x(\log(\frac{P(x)}{Q(x)}))$. In a sampling case, it becomes $KL = \sum_{i=1}^{n} P(x_i)\log(\frac{P(x_i)}{Q(x_i)})$. In our setting, P is a Gaussian distribution which has the same mean value and variance with x_i's in S and Q is the distribution of the network's outputs. The probability densities of network's outputs are estimated by a kernel based estimation method [50]. For a set of inputs $\{x_1, ..., x_n\}$, the probability density at x_i is calculated as

$$\hat{f}_h(x_i) = \frac{1}{nh}\sum_{j=1}^{n} K(\frac{x_i - x_j}{h}) \tag{4-23}$$

where h is a hyper parameter known as bandwidth, and K is Gaussian kernel, i.e., $K(x) = \frac{1}{\sqrt{2\pi}}\exp^{-\frac{x^2}{2}}$.

4.6.3 One Neural Network for All Stages

All the neural network based transformations we mentioned above are scalar functions. This brings clarity when we train this transformations but become an issue when we apply them to clustering or classification process. The issue is that we have five different neural networks corresponding to five different sleep stages. But we do not know the label of a point during inference time.

To address this problem, we proposed two candidate solutions. The first one is using the same strategy as in assembled fixed transformation, i.e., feed the new data point into every neural network and take the average value as the final transformed feature. However, the transformation proposed in 4.6.1 is a combination of multiple *local* function. This means each child function only impacts data within its support set and have no impact for data out of the scope. Neural networks, on the other hand, are *global* functions. The data point must be impacted by all the transformations. This approach is potentially feasible but makes each network very difficult to tune because of the interference.

The second solution is using a single neural network as a transformation for all stages. We deploy this idea by alternately feed data from each stage. Data in the same batch are of the same label, and labels are different from batch to batch.

5.0 Experiments and Results

5.1 Cluster-Then-Label Using k-Means and TinySleepNet

In this section, we will show how cluster-then-label strategy improves classifier's performance on real EEG signal. We use open-source data set "Sleep EDF Expanded" [28]. All the EEG signals in this data set are of 30 seconds window size and 100Hz sampling rate. The codes are implemented with python 3.6 and TensorFlow 1.13.1.

First, we pre-train the TinySleepNet classifier on the first twenty subjects using "twenty folders" method. To be specific, we train twenty classifiers independently which are all start from random initialization and been trained on 17 of overall 20 subjects. One of the three left subjects is used for testing and two for validation. The model with best performance on test set is selected for further training.

In the second step, EEG data from number $80, 81$ and 82 subjects are mixed to form a new data source. At this moment, the annotations that come along from the data set are only used for evaluating the performance and are invisible in training process. The CNN part of pre-trained classifier is used as a feature extractor which compresses the raw data of 3000 dimensions into 5 dimensions. Then the data are clustered based on the 5 dimension feature. Both GMM and k-means algorithms are tried in our experiment, yet only k-means gives improvement. Each cluster is assigned with the pseudo-label of its major class using the pre-trained classifier. These data are evenly divided into 6 folders. We retrain the classifier 6 times independently. Each time a different folder is selected as test data and another one for validation. The results of experiment are shown in table 1 and 2 showing the improvements on test set.

Table 1: Improvements in accuracy

Number of folder	1	2	3	4	5	6
Before training	74.6	74.2	82.4	84.9	89.6	83.4
After training	77.2	78.1	84.7	87.3	90.4	83.8

Table 2: Improvements in F1 score

Number of folder	1	2	3	4	5	6
Before training	55.1	49.7	68.0	68.4	73.6	69.5
After training	58.4	54.8	69.5	70.7	73.6	69.5

5.2 Cluster-Then-Label Using LDA and GMM

In this section, we will demonstrate how cluster-then-label strategy can improve the performance of simpler the model which takes band power features as their input. We use the EEG data from first ten subjects from Sleep EDF Expanded [28] and extract band power features using multitaper spectral analysis. Note that we keep the number of points of each stage the same in the training set to avoid the bias from prior probability. After feature extraction, we choose one subject (No.4 subject in our experiment) as the target subject. For other subjects, twenty percent data are separated as validation set and the rest points compose the training set. The classification accuracy raised from 75.3% to 79.5%.

5.3 Experiments on Artificial Data

In this section, we display how the distributions' statistics impact the performance of the clustering and the classification. Because all the processes and data distributions can be precisely modeled in these experiments, we can derive related analytical results in the future

following these insights and intuitions.

A very natural way to evaluate the performance of cluster-then-label is by the proportion of data points whose pseudo-labels are consistent with ground truth labels. Formally, we evaluate the overall performance with

$$\alpha = \frac{\hat{N}}{N} \tag{5-1}$$

where $\hat{N}$ is the number of data points whose pseudo-labels are consistent with their ground truth labels and N is the size of total unlabeled data.

5.3.1 2-D Experiments of Clustering

Clustering algorithms can capture extra geometrical information, but it also means that the following training process of classifier heavily rely on the clustering process. Hence, it is important to get some insights about when should we expect satisfying performance from cluster-then-label algorithm. In our 2-D experiments, we generate data from two Gaussian distributions $N(\mu_1, \Sigma_1)$ and $N(\mu_2, \Sigma_2)$. First we analyze the relationship between the accuracy of a single cluster-then-label step and the difference between the mean values of two Gaussian distributions. The pseudo-code of the experiment is shown in algorithm 3.

The second 2-D experiment is designed to explore the effect of covariance. The angle between the two eigenvectors that correspond to the largest eigenvalue of each covariance matrix is varied from 0 to $\frac{\pi}{2}$. The detailed steps are explained in algorithm 4 The results of these two experiment are shown in fig 5.3.1. The overall accuracy increases as the difference between the means and also as the angle increases. In fig 5(a), the curve has a long linear-like region except for the saturation at the end. In fig 5(b), the oscillation evidently decreases as the angle get closer to $\frac{\pi}{2}$, which implies the confidence of the classifier. Also there is no obvious linear-like region in this curve. Follow these curves, when the data are distributed similar to Gaussian distribution, one can evaluate how the clustering process will perform given the estimated means and covariance matrices.

Algorithm 3 2-D mean changing cluster-then-label

$\mu_1 \leftarrow (0,0)$, $\Sigma_1 \leftarrow I$, $\Sigma_2 \leftarrow I$

$max\_mean\_diff \leftarrow 5$, $step\_size \leftarrow 0.1, train\_size \leftarrow 50$, $increment \leftarrow 50$, $k \leftarrow 0$

while $k \leq max\_mean\_diff$ **do**

    Generate train data

    $train_1 \sim N(\mu_1, \Sigma_1)$, $|train_1| = train\_size$

    $train_2 \sim N((k,0), \Sigma_2)$, $|train_2| = train\_size$

    $train_data \leftarrow$shuffled$(train_1 \cup train_2)$

    Train GMM and LDA classifier on $train\_data$

    Generate new data

    $new_1 \sim N(\mu_1, \Sigma_1)$, $|new_1| = increment$

    $new_2 \sim N((k,0), \Sigma_2)$, $|new_2| = increment$

    $new_data \leftarrow$shuffled$(new_1 \cup new_2)$

    Cluster new_data, get clusters C_1, C_2

    Give labels to each cluster with classifier

    calculate α as in (5-1)

    $k \leftarrow k + step\_size$

end while

Algorithm 4 2-D eigenvectors angle changing cluster-then-label

htbp

$\mu_1 \leftarrow (2,5), \mu_2 \leftarrow (3.5,5), \Sigma_1 \leftarrow \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}, \Sigma_2 \leftarrow \begin{bmatrix} 1 & 0 \\ 0 & 0.1 \end{bmatrix}$,

$angle \leftarrow 0$, $iters \leftarrow 20$, $train\_size \leftarrow 50$, $increment \leftarrow 50$

while $angle \leq \frac{\pi}{2}$ **do**

    Generate train data

    $train_1 \sim N(\mu_1, \Sigma_1)$, $|train_1| = train\_size$

    $train_2 \sim N(\mu_2, \Sigma_2)$, $|train_2| = train\_size$

    $train\_data \leftarrow$ shuffled$(train_1 \cup train_2)$

    Train GMM and LDA classifier on $train\_data$

    Generate new data

    $new_1 \sim N(\mu_1, \Sigma_1)$, $|new_1| = increment$

    $new_2 \sim N(\mu_2, \Sigma_2), |new_2| = increment$

    $new\_data \leftarrow shuffled(new_1 \cup new_2)$

    Cluster new_data, get clusters C_1, C_2

    Give labels to each cluster with classifier

    calculate α as in (5-1)

    Rotate the eigenvectors of Σ_2 by $\frac{\pi}{2 \times iters}$

    Let $angle \leftarrow angle + \frac{\pi}{2 \times iters}$

end while

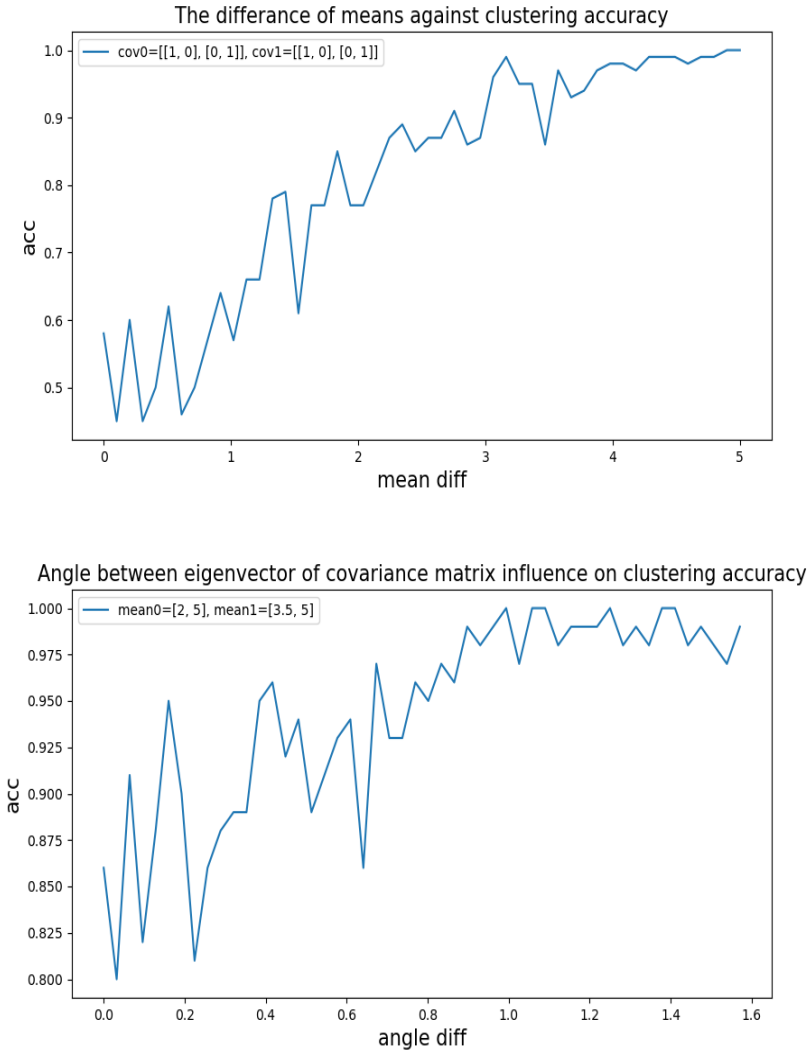

Figure 5: The cluster-then-label accuracy with respect to the mean difference and covariance.

5.3.2 2-D Experiment on the Effect of Mislabeled Data

In this section, we use experiments in 2-D to verify our analysis in 4.3. Let $\mu_0 = (0, 0)$, $\mu_1 = (0, 1)$ and the covariance matrix be $0.1 \times I$. Sample 200 points from each Gaussian distribution as the test set. In the ith run, let a be $0.01i$ and sample 50 points from each Gaussian distribution and use these data to estimate μ_0 and μ_1. Then calculate the risk on test set. Fig 5.3.2 visualizes how the true data, the mislabeled data and the classifier look like when a is 0 and 0.5, respectively. Fig 7 shows the overall trending of risk as a varies from 0 to 1. Since the classifier only depends on estimated means, it is insensitive to outliers thus the curve has two flat regions at the beginning and the end. However, the confidence is clearly low when a is close to 0.5.

5.4 Assembled Fixed Gaussian Transformation

According to [18], applying the transformation $\log \frac{x}{(1-x)}$ to relative band power ratio features can effciently convert current feature distributions to ones more like Gaussian distributions. We reproduce their experiment on our data and the result is shown in table 5.4. Note that data from multiple human subjects are content in this experiment. Though this fixed transformation can already reduce the statistics of AD test, it cannot handle the scenario where data from different stages are mixed together. As designed in algorithm 2, we conduct experiments of the assembled fixed Gaussian transformation. We use the same data set as in Section 5.2. One of the representative results is shown in fig 8 and table 4. The transformation is applied on the δ band of the data from the third subject in this figure. We choose δ band because it gives best separability. We can observe that the transformation clearly fixes the skewness of the awake stage and makes the peak of every stage more evident. The grooves (most clear at the middle of the awake stage) appear just as expected.

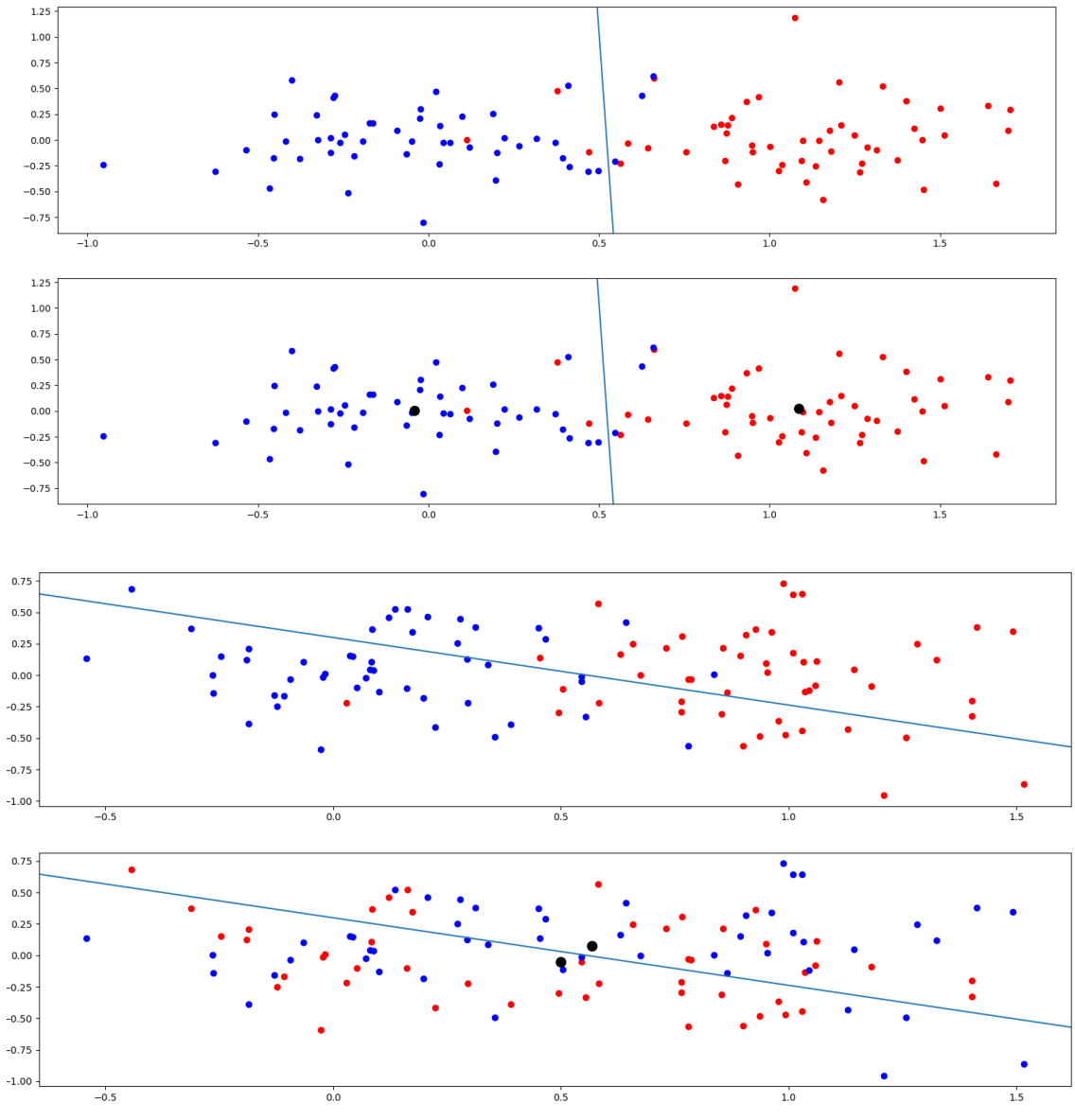

Figure 6: Visualization of classifiers.

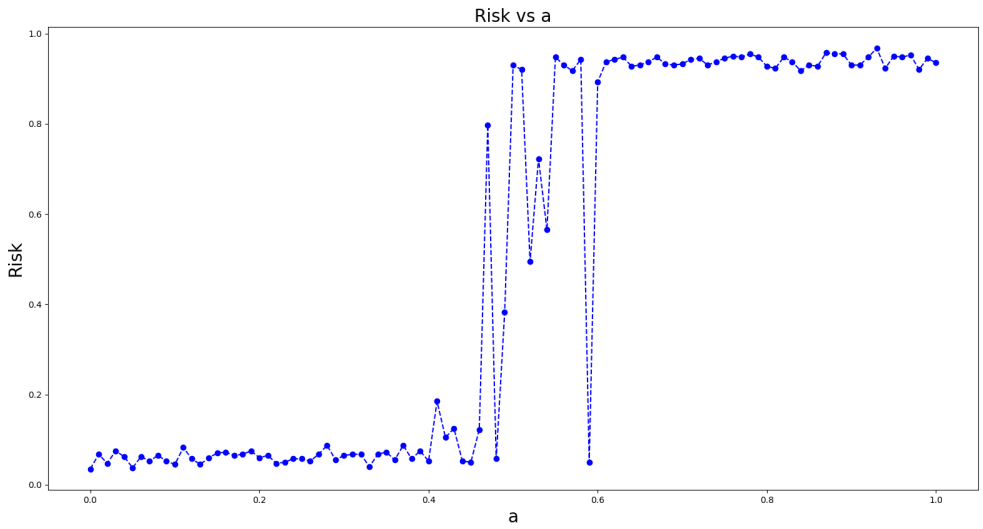

Figure 7: Overall trending of risk.

Table 3: Performance (AD test statistics) of fixed transformation

Before transformation				
Band	beta	alpha	theta	delta
Awake	137.09	252.17	93.45	178.01
N1	4.15	6.39	2.14	4.86
N2	1.81	30.19	0.48	10.01
N3	2.49	2.70	1.16	0.22
REM	5.92	4.09	2.07	1.47
After transformation				
Band	beta	alpha	theta	delta
Awake	156.99	223.13	52.09	49.78
N1	2.26	3.96	2.87	2.40
N2	0.73	18.64	1.26	4.09
N3	1.19	2.44	1.54	0.38
REM	3.67	7.44	1.27	0.55

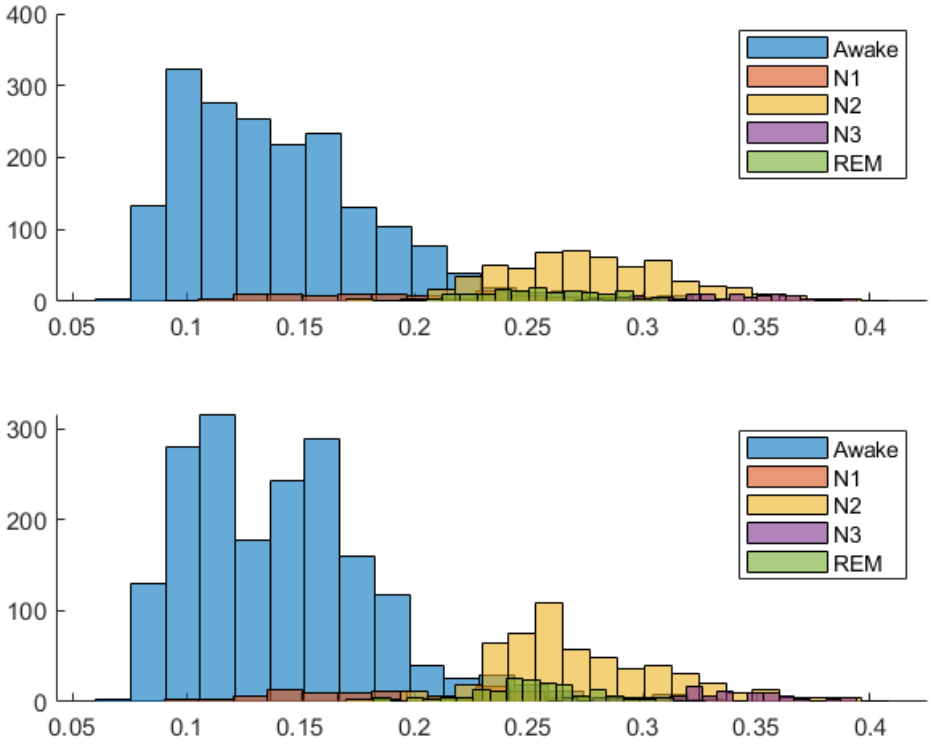

Figure 8: Data distribution before and after transformation.

Table 4: Improvements in AD statistics

Sleep stage	Awake	N1	N2	N3	REM
Before transformation	20.13	1.16	6.91	0.42	0.75
After transformation	26.70	0.82	1.44	0.28	0.21

5.5 Neural Network Based Transformations

In this section, we show the results of neural network based Gaussian transformation.

For all three set of experiments in this section, we use three layers shallow, fully connected network structure. The hidden layer contents 150 nodes with rectified linear unit (ReLU) activation function. Adam algorithm [29] is used for optimization. The hyper parameters of JB loss function are set as $\lambda_1 = 1$, $\lambda_2 = 5$ unless otherwise specified.

Gaussian transformations should be monotonically increasing functions and this cannot be easily promised when we are treating a neural network. To induce such property, we first initialize all the networks by imitating the fixed transformation $\log \frac{x}{(1-x)}$. The training data for initialization is uniformly sampled in the range of $1 - r$ percent EEG data points, where r has the same meaning as in algorithm 2. These EEG data are used for fine-tuning later. We try initialization training set of size 10000 and 100000 and the resulting curves are shown in 5.5. We conduct the same fine-tuning process on both initializations to compare their impacts on final results.

Data usage in our experiments are very special because of a major dilemma to use neural networks for Gaussian transformations. The target transformation is a scalar function that transform a single distribution to a more Gaussian one. The dilemma is that, on one hand, the feature distributions of different subjects (or different sleep stage) are very different. This means that the overall distribution of data from multiple subjects (or multiple sleep stages) is a mixture distribution which is against our target. On the other hand, if we only use data from one subject, the size of data set may not sufficient for training, and more importantly, the network may loss generalization ability. We report our experiments training the network on *one subject's one sleep stage (N1)* data along with our discussions and reflections on them.

5.5.1 Impact of Initialization

In fig 10 we present the resulting transformation function and the according distributions. The networks are trained with the same process from different initialization. The networks

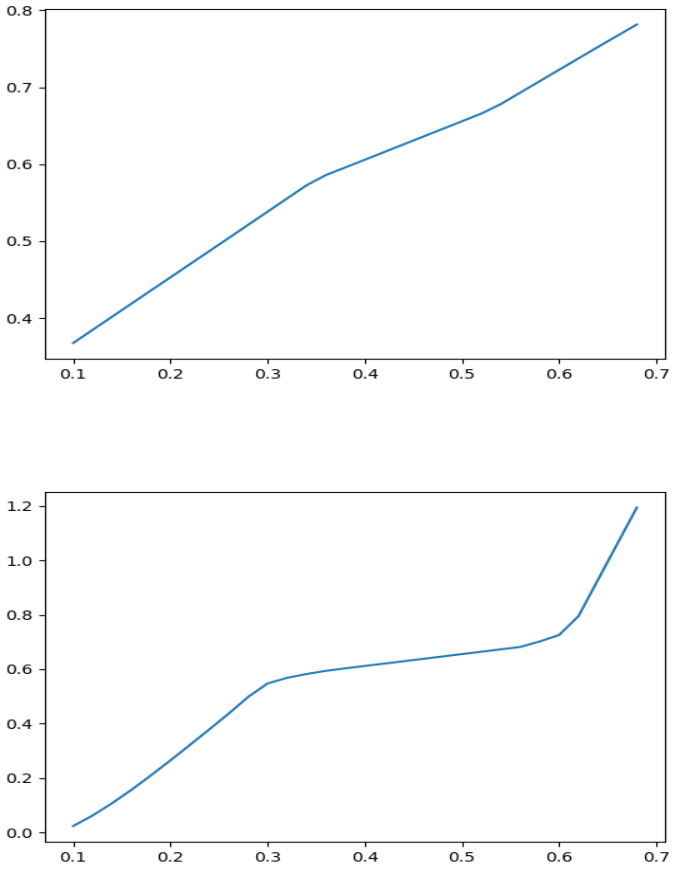

Figure 9: Initialization for Gaussian transformation networks.

are trained using KL divergence loss function in this figure. On the left of the first row is the transformation initialized with ten thousands samples, in the middle is the one initialized with a hundred thousands samples, and on the right is the one without any initialization. The second row, from left to right, are the distribution of input features, the distribution of the output of first transformation, the distribution of the second transformation, and that of the last transformation. The AD normality statistics of these distributions are 1.19, 2.06, 10.73, and 1.09. In fig 11, we show the results of the same experiment only now we train the network using JB loss. As before, the first row as the resulting transformations and two results of no initialization are included. The right-most two figures in the second row are the resulting distributions of them. The AD statistics of the resulting distributions in the second row are 1.19, 0.85, 0.60, 0.62, and 1.19. Here are some conclusions we can get from these two figures and the statistics:

1. Initialization by imitating the fixed transformation can actually induce a monotonically increasing property. In the absence of initialization, the resulting transformation may become monotonically decreasing or not monotonic at all (not shown in the figures).
2. Initialization strongly impacts the final shape of the transformations. Transformation with same initial parameters tend to end up with similar shapes though they usually have different ranges. The resulting shape of no initialization training are more of random.
3. Initialization does not always benefits the AD statistics of resulting distributions. In fact, initialization is always poisonous to the statistics when we use KL loss for training. Using the initialization from a hundred thousands samples, the network squeezes the distribution into a sharp peak. This does not have to be a bad news because we finally want to use the transformed data for clustering and classification. A sharp peak may indicates better separability. When training with JB loss function, initialization makes the transformation more steady (AD statistics always decrease) and better (compared with no initialization situations).

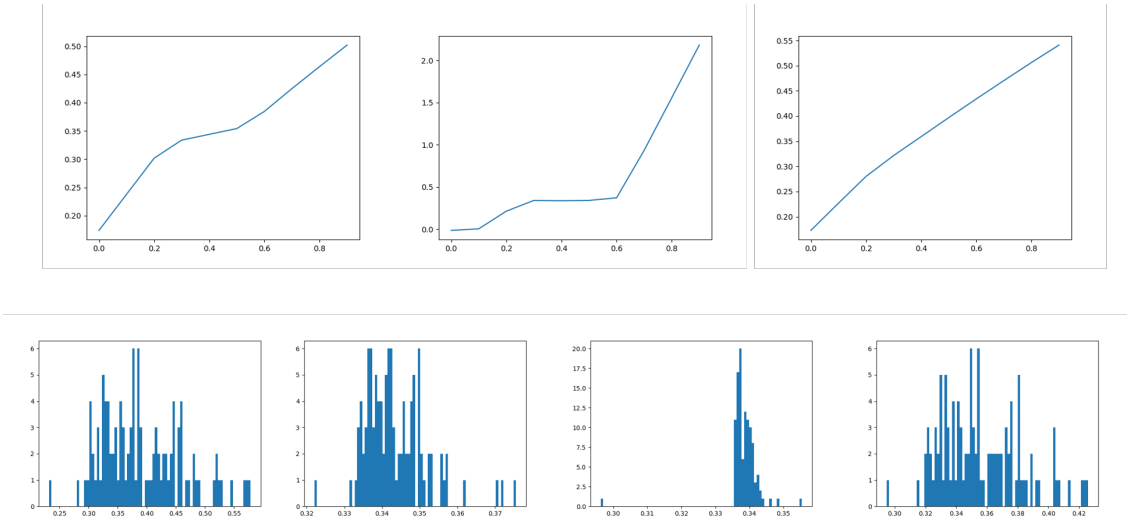

Figure 10: The impact of initialization using KL loss.

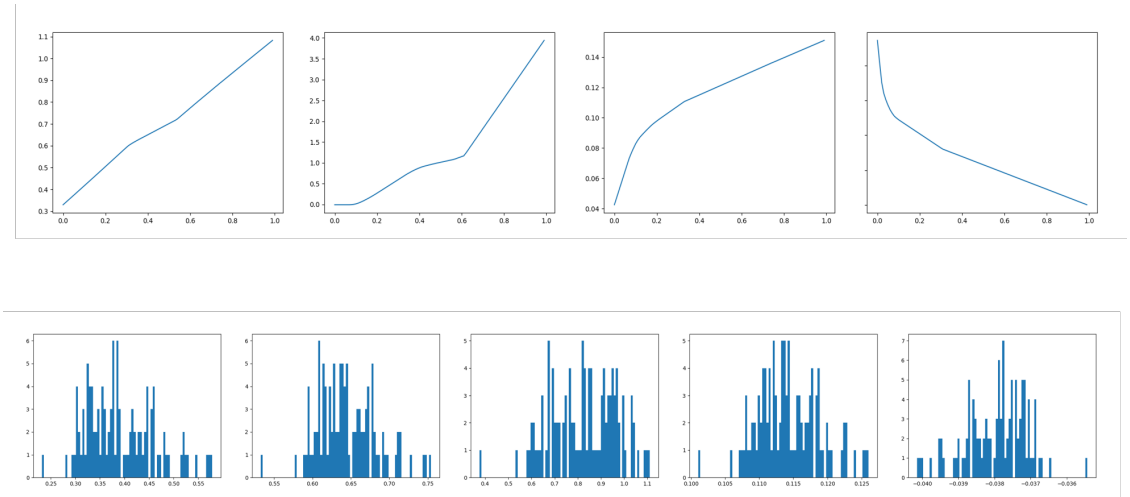

Figure 11: The impact of initialization using JB loss.

5.5.2 Impact of Different Training Loss

Now we compare the results from experiments using different training loss. Unless otherwise specified, all the experiments discussed in this section use initial parameters trained on ten thousands sample because it is in general beneficial. Fig 12 shows most representative behaviours of these three experiments. The first row shows the resulting transformations, and the second row presents the transformed distributions. When we train with KL loss, we **never** observe any resulting transformations violate the monotonically increasing requirement even if they start from random initialization. Training process using JB loss tends to generate smoother and linear like transformations, and training with AD loss prefer to the transformations that squeezes the outputs into a narrow range.

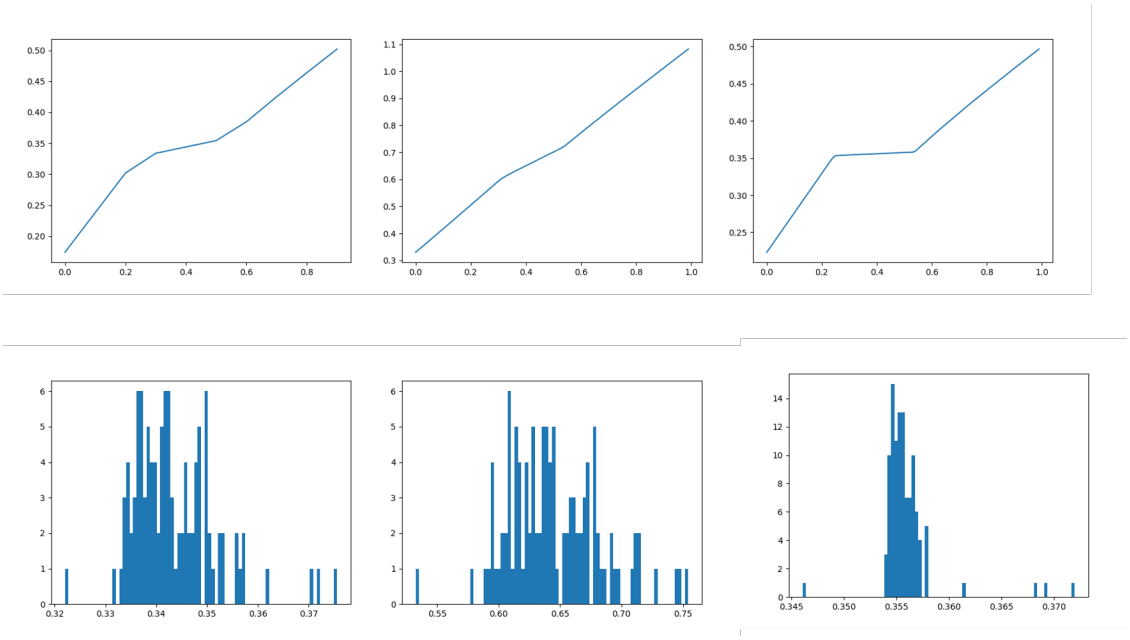

Figure 12: The impact of different training loss.

5.5.3 Experiments of One-for-All Neural Network

In this experiment, we use the same training set and test set as before. But now the absolute values of band power (instead of relative band power ratio) are used as input features. This is because the network cannot be effectively trained using relative band power

ratios. Training set takes 90 percent data from training *subject* and validation set takes the rest. Testing process still uses data from another subject. We train the network for 300 epochs using JB loss as the objective. The hyper parameter are set as $\lambda_1 = 0.1$, $\lambda_2 = 1$. Fig 13 shows the tracks of training and validation losses. The oscillation raises from the switch of sleep stage, and the overall decreasing trend indicates the training progress. Fig 14 and

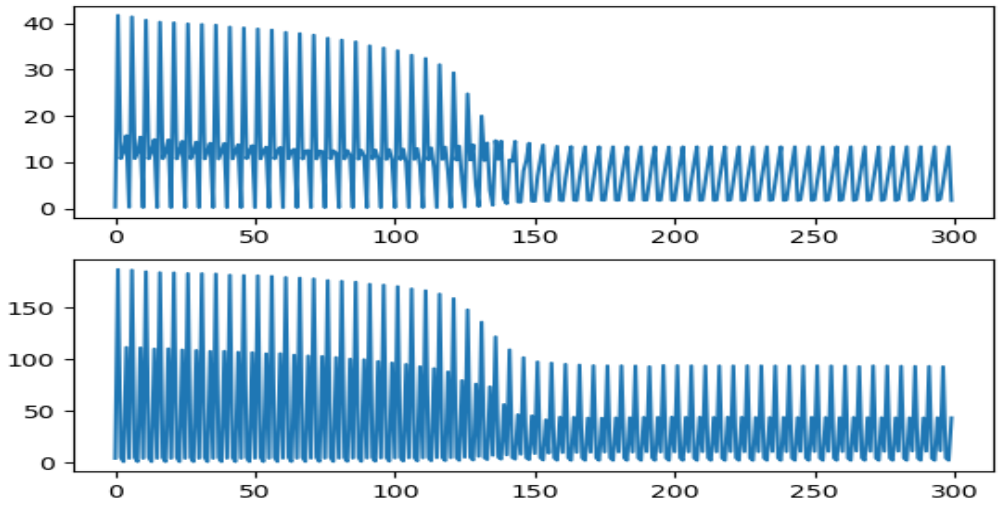

Figure 13: Loss history of one-for-all network.

fig 15 are the distributions of test data from each stage, before and after the transformation, respectively. The "s" above each mini-figure stands for the according AD statistic and the "mean" is the mean value for that distribution. The transformation reduced the AD statistics of most feature distributions, implying that it turned the original distributions closer to Gaussian in general. However, it is seen that when the original distributions are already close to normal distributions, the transformation might not work as expected.

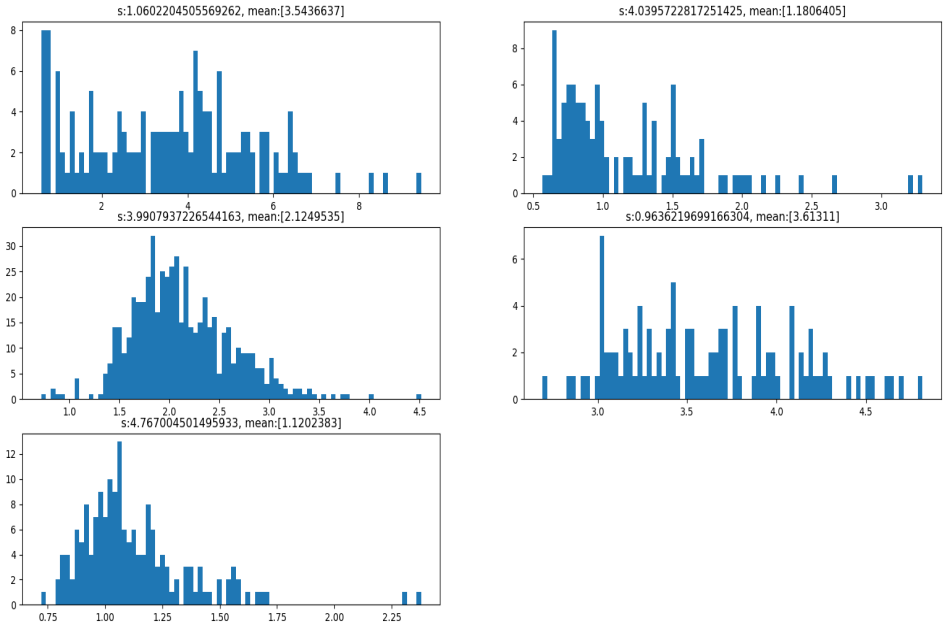

Figure 14: Distributions before transformation.

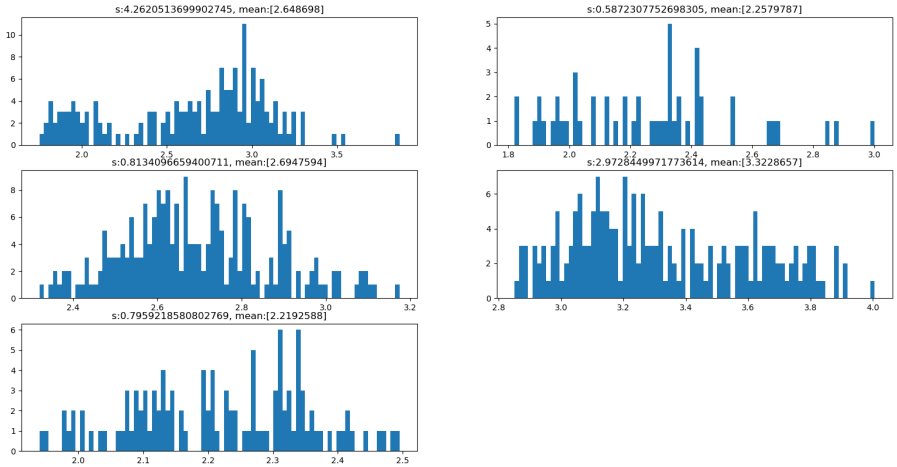

Figure 15: Distributions after transformation.

5.6 Transformation Networks with Dimensionality Reduction

This section describes an unsuccessful trail. At the beginning stage, we also consider a kind of neural network which compress the four dimensional feature into a scalar feature. This is because we observe networks with this structure are much easier to optimized. The network is trained with JB loss function, and we use the same data set as in Section 5.2. Because the original features and the outputs are of different dimensionalities, we compare the outputs with the results of principle component analysis (PCA). The results are shown in 5.6. Both the first row in fig 16(a) and 16(b) are the results from PCA and the second rows are the results of neural networks. Each column stands for a distinct sleep stages, namely awake, N1, N2, N3, and REM. Indeed, this kind of neural networks can perform Gaussian transformations, and even have good generalization ability (subject wise). But there are two major problems of this approach. Firstly, we empirically find the output's mean values shift dramatically even with the second term in (4-20). This shift completely ruins the biological meaning in original features. Thus, taking the mean value of all outputs (as mentioned in Section 4.6.3) will be totally meaningless. We first assume that these networks can obtain a implicit selectivity. For example, if one "awake" data point is fed into all five neural networks, the output of the "awake" network will be closer to the mean value of its outputs on training set. Unfortunately, we fail to endow the networks such property. The second major drawback of this kind of network is the dimensionality reduction itself. Band power features from each band are necessary (at least from a human's point of view) for correct sleep stage classification. Compressing the dimensionality of this kind of feature set loses critical information.

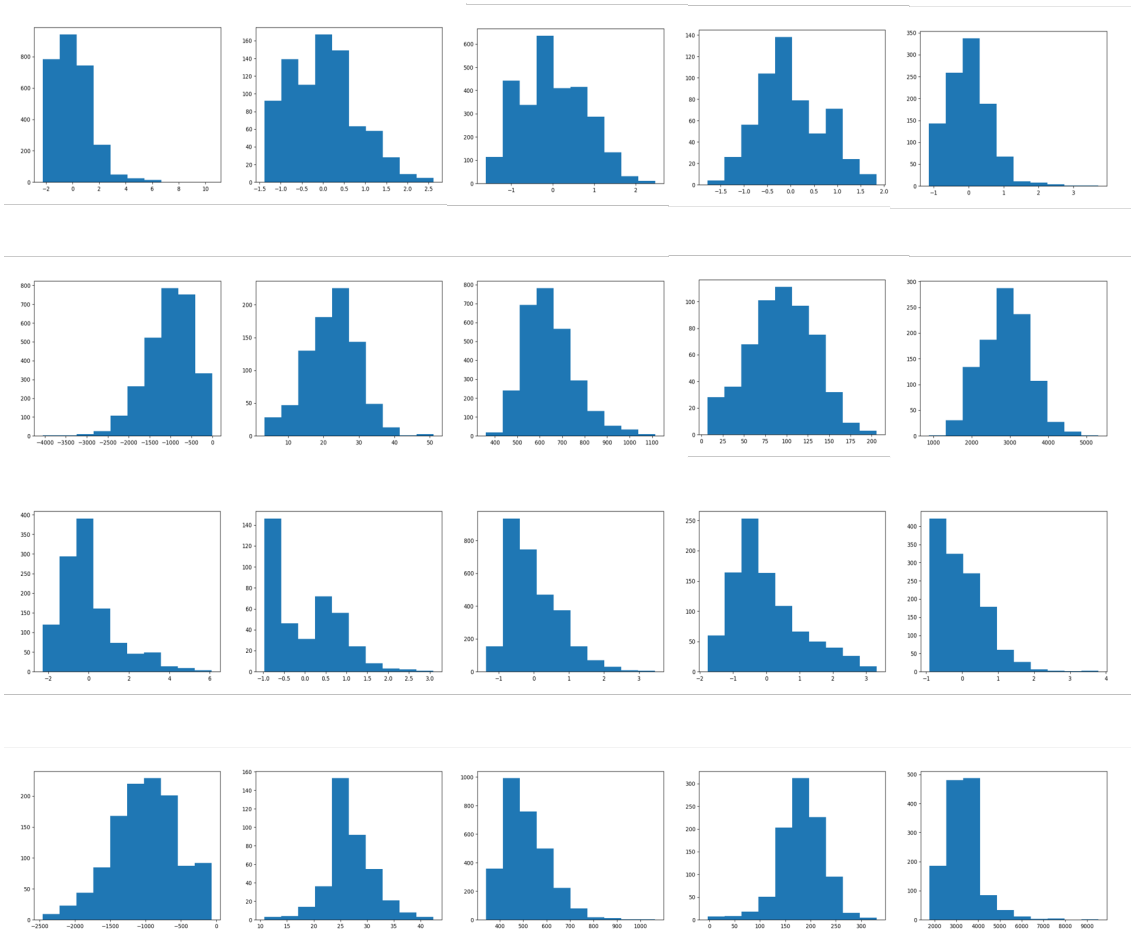

Figure 16: Comparison between the resulting distributions of PCA and neural networks.

6.0 Conclusions

In this thesis, we introduced how wearable sleep monitoring devices can help people who are suffering from sleep disorders. A cluster-then-label strategy is proposed to help the classifier to generalize on out-of-distribution subjects and prove the algorithm's effectiveness through experiments. To make the cluster-then-label process more accurate and robust, various Gaussian transformations approaches are designed and compared.

In the future, we will introduce additional components into the neural network process to make the output features more separable. The transformation's effectiveness need to be further proved by integrating them into cluster-then-label procedure. Theoretical analysis regarding the effect of pseudo label's errors still need to be accomplished.

Bibliography

[1] Mehdi Abdollahpour, Tohid Yousefi Rezaii, Ali Farzamnia, and Ismail Saad. Transfer learning convolutional neural network for sleep stage classification using two-stage data fusion framework. *IEEE Access*, 8:180618–180632, 2020.

[2] Emina Alickovic and Abdulhamit Subasi. Ensemble SVM method for automatic sleep stage classification. *IEEE Transactions on Instrumentation and Measurement*, 67(6):1258–1265, 2018.

[3] Theodore W Anderson and Donald A Darling. A test of goodness of fit. *Journal of the American Statistical Association*, 49(268):765–769, 1954.

[4] Fernando Andreotti, Huy Phan, Navin Cooray, Christine Lo, Michele TM Hu, and Maarten De Vos. Multichannel sleep stage classification and transfer learning using convolutional neural networks. In *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 171–174, 2018.

[5] Haoran Bai and Guanze Lu. Semi-supervised end-to-end automatic sleep stage classification based on pseudo-label. In *2021 IEEE International Conference on Power Electronics, Computer Applications*, pages 83–87, 2021.

[6] Stephen Balaban. Deep learning and face recognition: The state of the art. In *Biometric and Surveillance Technology for Human and Activity Identification XII*, volume 9457, 2015.

[7] Yogesh Balaji, Swami Sankaranarayanan, and Rama Chellappa. Metareg: Towards domain generalization using meta-regularization. *Advances in Neural Information Processing Systems*, 31, 2018.

[8] Richard B Berry, Rita Brooks, Charlene E Gamaldo, Susan M Harding, C Marcus, Bradley V Vaughn, et al. The AASM manual for the scoring of sleep and associated events: Rules, terminology and technical specifications. *American Academy of Sleep Medicine*, 176, 2012.

[9] Gilles Blanchard, Gyemin Lee, and Clayton Scott. Generalizing from several related classification tasks to a new unlabeled sample. *Advances in Neural Information Processing Systems*, 24, 2011.

[10] Fabio M Carlucci, Antonio D'Innocente, Silvia Bucci, Barbara Caputo, and Tatiana Tommasi. Domain generalization by solving jigsaw puzzles. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2229–2238, 2019.

[11] Keyu Chen, Di Zhuang, and J Morris Chang. Discriminative adversarial domain generalization with meta-learning based cross-domain validation. *Neurocomputing*, 467:418–426, 2022.

[12] Sudhansu Chokroverty. Overview of sleep and sleep disorders. *Indian J Med Res*, 131(2):126–140, 2010.

[13] Mayo Clinic. EEG (electroencephalogram). `https://www.mayoclinic.org/tests-procedures/eeg/about/pac-20393875`.

[14] Mohammed Diykh, Yan Li, and Peng Wen. EEG sleep stages classification based on time domain features and structural graph similarity. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 24(11):1159–1168, 2016.

[15] Yingjun Du, Jun Xu, Huan Xiong, Qiang Qiu, Xiantong Zhen, Cees GM Snoek, and Ling Shao. Learning to learn with variational information bottleneck for domain generalization. In *European Conference on Computer Vision*, pages 200–216. Springer, 2020.

[16] Luay Fraiwan, Khaldon Lweesy, Natheer Khasawneh, Mohammad Fraiwan, H Wenz, and H Dickhaus. Classification of sleep stages using multi-wavelet time frequency entropy and LDA. *Methods of Information in Medicine*, 49(3):230–237, 2010.

[17] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *International Conference on Machine Learning*, pages 1180–1189. PMLR, 2015.

[18] Theo Gasser, Petra Bächer, and Joachim Möcks. Transformations towards the normal distribution of broad band spectral parameters of the EEG. *Electroencephalography and Clinical Neurophysiology*, 53(1):119–124, 1982.

[19] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[20] Thomas Grubinger, Adriana Birlutiu, Holger Schöner, Thomas Natschläger, and Tom Heskes. Domain generalization based on transfer component analysis. In *International Work-Conference on Artificial Neural Networks*, pages 325–334. Springer, 2015.

[21] Shoubo Hu, Kun Zhang, Zhitang Chen, and Laiwan Chan. Domain generalization via multidomain discriminant analysis. In *Uncertainty in Artificial Intelligence*, pages 292–302. PMLR, 2020.

[22] Zeyi Huang, Haohan Wang, Eric P Xing, and Dong Huang. Self-challenging improves cross-domain generalization. In *European Conference on Computer Vision*, pages 124–140. Springer, 2020.

[23] Syed Anas Imtiaz. A systematic review of sensing technologies for wearable sleep staging. *Sensors*, 21(5):1562, 2021.

[24] Pankaj Jadhav, Gaurav Rajguru, Debabrata Datta, and Siddhartha Mukhopadhyay. Automatic sleep stage classification using time–frequency images of CWT and transfer learning using convolution neural network. *Biocybernetics and Biomedical Engineering*, 40(1):494–504, 2020.

[25] Vijaylaxmi P Jain, VD Mytri, VV Shete, and BK Shiragapur. Sleep stages classification using wavelettransform and neural network. In *Proceedings of 2012 IEEE-EMBS International Conference on Biomedical and Health Informatics*, pages 71–74, 2012.

[26] Ziyu Jia, Youfang Lin, Jing Wang, Xiaojun Ning, Yuanlai He, Ronghao Zhou, Yuhan Zhou, and L.-W. H Lehman. Multi-view spatial-temporal graph convolutional networks with domain generalization for sleep stage classification. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 29:1977–1986, 2021.

[27] Xin Jin, Cuiling Lan, Wenjun Zeng, Zhibo Chen, and Li Zhang. Style normalization and restitution for generalizable person re-identification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3143–3152, 2020.

[28] B. Kemp, A.H. Zwinderman, B. Tuk, H.A.C. Kamphuisen, and J.J.L. Oberye. Analysis of a sleep-dependent neuronal feedback loop: the slow-wave microcontinuity of the EEG. *IEEE Transactions on Biomedical Engineering*, 47(9):1185–1194, 2000.

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

[30] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[31] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy Hospedales. Learning to generalize: Meta-learning for domain generalization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[32] Da Li, Jianshu Zhang, Yongxin Yang, Cong Liu, Yi-Zhe Song, and Timothy M Hospedales. Episodic training for domain generalization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1446–1455, 2019.

[33] Haoliang Li, Sinno Jialin Pan, Shiqi Wang, and Alex C Kot. Domain generalization with adversarial feature learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5400–5409, 2018.

[34] Ya Li, Xinmei Tian, Mingming Gong, Yajing Liu, Tongliang Liu, Kun Zhang, and Dacheng Tao. Deep domain generalization via conditional invariant adversarial networks. In *Proceedings of the European Conference on Computer Vision*, pages 624–639, 2018.

[35] Yiying Li, Yongxin Yang, Wei Zhou, and Timothy Hospedales. Feature-critic networks for heterogeneous domain generalization. In *International Conference on Machine Learning*, pages 3915–3924. PMLR, 2019.

[36] David JC MacKay, David JC Mac Kay, et al. *Information theory, inference and learning algorithms*. Cambridge University Press, 2003.

[37] Massimiliano Mancini, Samuel Rota Bulo, Barbara Caputo, and Elisa Ricci. Best sources forward: Domain generalization through source-specific nets. In *2018 25th IEEE International Conference on Image Processing*, pages 1353–1357. IEEE, 2018.

[38] Tom M Mitchell and Tom M Mitchell. *Machine learning*, volume 1. McGraw-Hill New York, 1997.

[39] Andreas Muff Munk, Kristoffer Vinther Olesen, Sirin Wilhelmsen Gangstad, and Lars Kai Hansen. Semi-supervised sleep-stage scoring based on single channel EEG. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 2551–2555, 2018.

[40] W Chance Nicholson and Kate Pfeiffer. Sleep disorders and mood, anxiety, and post-traumatic stress disorders: Overview of clinical treatments in the context of sleep disturbances. *Nursing Clinics*, 56(2):229–247, 2021.

[41] Sinno Jialin Pan, Ivor W Tsang, James T Kwok, and Qiang Yang. Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks*, 22(2):199–210, 2010.

[42] Mohammad Peikari, Sherine Salama, Sharon Nofech-Mozes, and Anne L Martel. A cluster-then-label semi-supervised learning approach for pathology image classification. *Scientific Reports*, 8(1):1–13, 2018.

[43] Michael J Prerau, Ritchie E Brown, Matt T Bianchi, Jeffrey M Ellenbogen, and Patrick L Purdon. Sleep neurophysiological dynamics through the lens of multitaper spectral analysis. *Physiology*, 32(1):60–92, 2017.

[44] Mustafa Radha, Pedro Fonseca, Arnaud Moreau, Marco Ross, Andreas Cerny, Peter Anderer, Xi Long, and Ronald M Aarts. A deep transfer learning approach for wearable sleep stage classification with photoplethysmography. *NPJ Digital Medicine*, 4(1):1–11, 2021.

[45] Kannan Ramar, Raman K Malhotra, Kelly A Carden, Jennifer L Martin, Fariha Abbasi-Feinberg, R Nisha Aurora, Vishesh K Kapur, Eric J Olson, Carol L Rosen, James A Rowley, and Anita V Shelgikar. Sleep is essential to health: An American Academy of Sleep Medicine position statement. *Journal of Clinical Sleep Medicine*, 17(10):2115–2119, 2021.

[46] Douglas A Reynolds. Gaussian mixture models. *Encyclopedia of Biometrics*, 741(659-663), 2009.

[47] Jose Luis Rodríguez-Sotelo, Alejandro Osorio-Forero, Alejandro Jiménez-Rodríguez, David Cuesta-Frau, Eva Cirugeda-Roldán, and Diego Peluffo. Automatic sleep stages classification using eeg entropy features and unsupervised pattern analysis techniques. *Entropy*, 16(12):6573–6589, 2014.

[48] Donald B Rubin and Dorothy T Thayer. EM algorithms for ML factor analysis. *Psychometrika*, 47(1):69–76, 1982.

[49] Shiv Shankar, Vihari Piratla, Soumen Chakrabarti, Siddhartha Chaudhuri, Preethi Jyothi, and Sunita Sarawagi. Generalizing across domains via cross-gradient training. *International Conference on Learning Representations*, 2018.

[50] Simon J Sheather. Density estimation. *Statistical Science*, pages 588–597, 2004.

[51] Michael Sokolovsky, Francisco Guerrero, Sarun Paisarnsrisomsuk, Carolina Ruiz, and Sergio A Alvarez. Deep learning for automated feature discovery and classification of sleep stages. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 17(6):1835–1845, 2019.

[52] Akara Supratak and Yike Guo. TinySleepNet: An efficient deep learning model for sleep stage scoring based on raw single-channel EEG. In *2020 42nd Annual International Conference of the IEEE Engineering in Medicine & Biology Society*, pages 641–644, 2020.

[53] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT Press, 2018.

[54] Thorsten Thadewald and Herbert Büning. Jarque–bera test and its competitors for testing normality–a power comparison. *Journal of Applied Statistics*, 34(1):87–105, 2007.

[55] Riccardo Volpi, Hongseok Namkoong, Ozan Sener, John C Duchi, Vittorio Murino, and Silvio Savarese. Generalizing to unseen domains via adversarial data augmentation. *Advances in Neural Information Processing Systems*, 31, 2018.

[56] Olivia Walch, Yitong Huang, Daniel Forger, and Cathy Goldstein. Sleep stage prediction with raw acceleration and photoplethysmography heart rate data derived from a consumer wearable device. *Sleep Research Society*, 42(12):zsz180, 2019.

[57] Jindong Wang, Cuiling Lan, Chang Liu, Yidong Ouyang, Tao Qin, Wang Lu, Yiqiang Chen, Wenjun Zeng, and Philip Yu. Generalizing to unseen domains: A survey on domain generalization. *IEEE Transactions on Knowledge and Data Engineering*, 2022.

[58] Hao Wu and Saurabh Prasad. Semi-supervised deep learning using pseudo labels for hyperspectral image classification. *IEEE Transactions on Image Processing*, 27(3):1259–1270, 2017.

[59] Xiaolei Wuzheng, Shigang Zuo, Li Yao, and Xiaojie Zhao. Semi-supervised sparse representation classification for sleep EEG recognition with imbalanced sample sets. *Journal of Mechanics in Medicine and Biology*, 21(5):2140006–1–2140006–13, 2021.

[60] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. Mixup: Beyond empirical risk minimization. *International Conference on Learning Representations*, 2018.

[61] Li Zhang, Junjun Xiong, Heng Zhao, Hong Hong, Xiaohua Zhu, and Changzhi Li. Sleep stages classification by CW Doppler radar using bagged trees algorithm. In *2017 IEEE Radar Conference*, pages 788–791, 2017.

[62] Ranqi Zhao, Yi Xia, and Yongliang Zhang. Unsupervised sleep staging system based on domain adaptation. *Biomedical Signal Processing and Control*, 69:1–9, 2021.

[63] Fan Zhou, Zhuqing Jiang, Changjian Shui, Boyu Wang, and Brahim Chaib-draa. Domain generalization with optimal transport and metric learning. *arXiv preprint arXiv:2007.10573*, 2020.

[64] Kaiyang Zhou, Ziwei Liu, Yu Qiao, Tao Xiang, and Chen Change Loy. Domain generalization: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[65] Kaiyang Zhou, Yongxin Yang, Timothy Hospedales, and Tao Xiang. Deep domain-adversarial image generation for domain generalisation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 13025–13032, 2020.

[66] Guohun Zhu, Yan Li, and Peng Wen. Analysis and classification of sleep stages based on difference visibility graphs from a single-channel EEG signal. *IEEE Journal of Biomedical and Health Informatics*, 18(6):1813–1821, 2014.

[67] Xiaojin Zhu and Andrew B Goldberg. Introduction to semi-supervised learning. *Synthesis Lectures on Artificial Intelligence and Machine learning*, 3(1):1–130, 2009.