Nanomaterial-Based Sensor Array Signal Processing and Classification Using Machine Learning

Chenxi Liu

Nanomaterial-Based Sensor Array Signal Processing and Classification Using Machine Learning

Research Thesis

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical Engineering

Chenxi Liu

Submitted to the Senate
of the Technion — Israel Institute of Technology
Iyyar 5783 Haifa May 2023

This research was carried out under the supervision of Prof. Israel Cohen, in the Faculty of Electrical and Computer Engineering.

The author and research collaborators have submitted some of the results presented in this thesis as an article in a journal during the author's master's research period. The latest version of the article is:

Liu Chenxi, Israel Cohen, Vishinkin Rotem, and Haick Hossam. Nanomaterial-based sensor array signal processing and tuberculosis classification using machine learning. *Journal of Low Power Electronics and Applications*, 29 May 2023.

Acknowledgements

This thesis was done under the supervision of Professor Israel Cohen in the Department of Electrical and Computer Engineering.

I would like to express my heartfelt gratitude to Professor Israel Cohen, my advisor, for his invaluable guidance, unwavering support, and patient supervision throughout this research. His mentorship has helped me develop new skills, broaden my knowledge, and foster independent thinking, for which I am greatly grateful.

I extend my sincere thanks to my family, especially my mother, for their love, unwavering support, and encouragement during this journey. Without their constant motivation, this achievement would not have been possible.

Finally, I wish to express my deepest appreciation to my faithful God for His unfailing love, guidance, and encouragement throughout my Master's study. His presence has enabled me to navigate through the many challenges and obstacles and to continue with unwavering faith in times of uncertainty and adversity.

The generous financial support of the Technion is gratefully acknowledged.

Contents

List of Figures

Abstract **1**

Abbreviations **3**

Notations **5**

1 Introduction **7**

2 Background **11**

2.1 Related Work 11

2.2 Time series classification 13

2.3 Encoding Time Series as Images 13

2.4 The Long Short Term Memory Network 14

2.5 The Graph Neural Network Model 15

2.5.1 Spectral Clustering and MinCutPool 16

2.6 Binary Cross Entropy 16

2.6.1 Dynamic Time Warping (DTW) 17

3 Nano-material based sensor signal processing and classification methods **19**

3.1 Introduction 19

3.2 Data Preprocessing 20

3.2.1 Dataset Description 20

3.2.2 Middle Part Signal Extraction 21

3.2.3 Data Normalization 21

3.2.4 Data Calibration 23

3.2.5 Sensor Selection by Using Pearson Correlation Coefficient Matrix 23

3.3 Long Short Term Memory (LSTM) Network 26

3.4 Convolution Neural Network (CNN) 27

3.5 GAF-CNN 28

3.6 MTSC with Graph Laplacian and MinCutPool 30

3.6.1 Graph Structure Learning Using Laplacian Matrix 31
3.6.2 Temporal Feature Extraction 32
3.6.3 Spatial-Temporal Modeling 33
3.6.4 Graph Coarsen by MinCutPool 33
3.6.5 Graph-Level Embedding Classification 34
3.7 Conclusions 34

4 Experiments and Result Analysis 37
4.1 Introduction 37
4.2 Experiments 37
4.2.1 Evaluation Metrics 37
4.2.2 Experimental Setup 38
4.3 Results and Discussion 39
4.3.1 Main Results 39
4.3.2 Discussion 40
4.4 Conclusions 43

5 Conclusions 47
5.1 Summary 47
5.2 Future Work 47

List of Figures

2.1 The structure of LSTM network. 14

3.1 The original 40 sensor signals corresponding to one sample before data preprocessing. 20
3.2 Nanomaterial-based sensor resistance signals, where the x-axis is the time cycles, and the y-axis is the resistance values. Useful signals are obtained under the exposure of the skin sample (within the red dotted line). 22
3.3 Wearable sensor devices on a volunteer's chest and anterior arm. 22
3.4 The change in the sensor resistivity (i.e. $\Delta R_{end}/R_b$) for different storage conditions at the starting point (M0) and after 9 months (M9). 22
3.5 The original 40 sensor signals correspond to one sample after middle part extraction, data normalization, and data calibration. 23
3.6 Pearson correlation coefficient matrix heat maps for both unstable sensors (a) and good sensors (b). 24
3.7 Selected 29 sensor signals of a sample after middle part extraction. . . . 25
3.8 The proposed LSTM model architecture. 26
3.9 CNN model architecture. 28
3.10 GAF-CNN model architecture. 30
3.11 MTMinCutPool model architecture. 31
3.12 Summary of main steps of building the Laplacian matrix to represent the graph structure. 32

4.1 The Receiver Operating Characteristic (ROC) curve of each model. . . . 42
4.2 The training and validation accuracy and loss curve for the best LSTM model. 43
4.3 The training and validation accuracy and loss curve for the best CNN model. 43
4.4 The training and validation accuracy and loss curve for the best GAF-CNN model. 44
4.5 The training and validation accuracy and loss curve for the best MT-MinCutPool model. 44

4.6 The average confusion matrices computed from 10 runs for each of the proposed models: CNN, LSTM, GAF-CNN, and MT-MinCutPool. . . . 45
4.7 The average confusion matrices computed from 10 runs for each of the baseline models: MLSTM-FCN, MTPool, and GAF-Attention. 46

Abstract

Tuberculosis (TB) has long been recognized as a significant health concern worldwide. Recent advancements in noninvasive wearable devices and deep learning (DL) techniques have enabled the development of rapid and cost-effective TB testing, which has the potential to facilitate real-time TB detection. However, DL models can face challenges in achieving high performance when working with small datasets standard in biomedical and chemical engineering domains. In particular, small datasets can lead to overfitting issues that can hinder the success of ML models. To address this challenge, we propose various data preprocessing methods and DL approaches, including Long Short Term Memory (LSTM), Convolutional Neural Network (CNN), Gramian Angular Field-CNN (GAF-CNN), and Multivariate Time Series with MinCutPool (MT-MinCutPool), for the classification of a small TB dataset consisting of multivariate time series (MTS) sensor signals.

To evaluate the efficacy of our proposed methods, we compared them with state-of-the-art models commonly used in multivariate time series classification (MTSC) tasks. Our experimental results demonstrate that lightweight models are more suitable for small-dataset problems. Moreover, the average performance of our proposed models surpassed that of the baseline methods in all aspects. Specifically, the GAF-CNN model achieved the highest accuracy of 0.639 and the highest specificity of 0.777, indicating its superior effectiveness for MTSC tasks. Additionally, our proposed MT-MinCutPool model outperformed the baseline MTPool model in all evaluation metrics, demonstrating its viability for MTSC tasks. The proposed approach can be applied to other domains that suffer from small datasets or involve MTS signals.

In conclusion, our proposed approaches can be applied to similar domains that suffer from small datasets or involve MTS signals. The results of our study suggest that the proposed models can help to improve the classification performance of MTS signals in small TB datasets and provide a promising avenue for the development of rapid and cost-effective TB testing methods.

Abbreviations

AUC	:	Area Under the Curve
BCE	:	Binary Cross Entropy
CI	:	Confidence Interval
CNN	:	Convolution Neural Network
CV	:	Computer Vision
DL	:	Deep Learning
DNNs	:	Deep Neural Networks
DTW	:	Dynamic Time Warping
FN	:	False Negative
FP	:	False Positive
FPR	:	False Positive Rate
GAF	:	Gramian Angular Field
GAF-CNN	:	Gramian Angular Field-CNN
GCN	:	Graph Convolution Network
GNN	:	Graph Neural Network
GNPs	:	Gold Nanoparticles
KL-Divergence	:	Kullback-Leibler Divergence
LSTM	:	Long Short Term Memory
ML	:	Machine Learning
MLP	:	Multilayer Perceptron
MLSTM-FCN	:	Multivariate LSTM fully convolutional network
MT-MinCutPool	:	Multivariate Time Series with MinCutPool
MTPool	:	Multivariate Time Series Classification with Variational Graph Pooling
MTS	:	Multivariate Time Series
MTSC	:	Multivariate Time Series Classification
NLP	:	Natural Language Processing
RNN	:	Recurrent Neural Network
RN-SWCNTs	:	Random Networks of Single-Walled Carbon Nanotubes

ROC	:	Receiver Operating Characteristic
SC	:	Spectral Clustering
TB	:	Tuberculosis
TL	:	Transfer Learning
TN	:	True Negative
TNR	:	True Negative Rate
TP	:	True Positive
TPR	:	True Positive Rate
TSC	:	Time Series Classification
VAE	:	Variational Auto Encoder
VOC	:	Volatile Organic Compound
WHO	:	World Health Organization
2D	:	Two-Dimensional

Notations

A	the adjacency matrix.
$\tilde{A}$	the symmetrically normalized adjacency matrix.
A_{pool}	the coarsened symmetrical adjacency matrix.
$arccos(\cdot)$	The inverse cosine function.
b	a bias term.
$cov(\cdot)$	the covariance between two variables.
C_{xy}	the Pearson Correlation Coefficient between sample x and y.
d	the number of the new feature dimension.
D	the degree matrix.
$\mathcal{D}$	distance matrix.
$diagonal(\cdot)$	the diagonalized function.
$D_{KL}(\cdot\|\|\cdot)$	the Kullback-Leibler divergence.
d(p, q)	distance between p and q.
$DTW(\cdot, \cdot)$	the dynamic time warping algorithm.
f	the output tensor of the convolution layer.
$f_h(\cdot)$	state update function.
g	activation vector.
$G(\cdot)$	the graph convolution function.
$H_p(q)$	the cross-entropy between distribution $p(\cdot)$ and $q(\cdot)$.
I	projection matrix.
in	the number of the input channels.
ks	the number of the kernel size.
L	the Laplacian matrix.
$\mathcal{L}_u$	the cut loss term.
max$(\cdot)$	maximum operation to find the biggest value in a set of numbers or a function.
$med(\cdot)$	the median function.
$MLP(\cdot)$	the multilayer perceptron operation.
min$(\cdot)$	minimum operation to find the smallest value in a set of numbers or a function.
n	the number of sensors.
N	a constant factor.
out	the number of output channels.
P, Q	two time-series sensor signals.

$Peak(\cdot)$	the peak point of the input signal.
p(y)	the predicted probability.
q(y)	the true distribution.
r	the radius of the transformed time stamp.
$ReLU(\cdot)$	the Rectified Linear Unit activation function.
s	stride step.
S	the cluster assignment matrix.
T	the number of total time steps.
$tanh(\cdot)$	hyperbolic tangent function.
t_i	time step i.
{u, f, o, c}	represent the input, forget, output, and the cell state gates.
$var(\cdot)$	the variance of a variable.
W	weight matrix.
$\mathcal{W}$	a warping path consists of a contiguous set of matrix indices.
X	A univariate or multivariate time series.
$\hat{X}$	the normalized X.
$\tilde{X}$	the output feature matrix after graph convolution.
X_{dense}	the flattened feature vector.
x_i	the i-th element of the time series X.
$\tilde{x_i}$	the rescaled i-th element of the time series X.
X_{pool}	the coarsened graph representation feature matrix.
X_{TC}	temporal feature matrix.
Y	one hot label vector.
$\oplus$	element-wise summation.
$\otimes$	element-wise multiplication.
$\sigma(\cdot)$	logistic sigmoid function.
ϕ	the angular cosine value.
$\sum$	the summation function.
$\|\|$	concatenation operation.
$*$	convolution operation.

Chapter 1

Introduction

Tuberculosis (TB) is an ancient, chronic disease caused by the bacillus Mycobacterium tuberculosis, which threatens an estimated 25% of the world's population, with a 5%-10% life-long risk of developing into TB disease. It usually affects the lungs and can spread from person to person through the air. Patients with pulmonary TB symptoms include a chronic cough, weight loss, chest pain, weakness, fatigue, night sweats, and fever [1]. TB has affected humanity for over 4,000 years, and over 10 million people get infected annually. Therefore, it remains one of the leading causes of morbidity and mortality worldwide, especially in developing countries. Early recognition of TB and prompt detection of drug resistance is critical to reducing its global burden. To overcome this problem, Vishinkin et al. [2] proposed a novel diagnostic pathway to detect TB in a noninvasive, reliable, and rapid manner. They developed a new biomedical apparatus containing a wearable and flexible polymer pouch for collecting and storing TB-specific volatile organic compounds (VOCs) that can be detected and quantified from the air above the skin (the skin's headspace). An abnormal pattern of VOC concentrations that deviates from the healthy pattern may indicate either TB infection or a high risk of infection with TB. The collected VOCs will then be fed into a set of specially designed nanomaterial-based sensors capable of detecting a variety of skin-based TB VOCs [3–6]. Finally, the sensors will translate these collected VOCs into time series resistance signals. Ultimately, the output multivariate time series (MTS) sensor signals will be used as feature inputs in DL models for the final discrimination between positive TB cases and healthy controls. Machine learning (ML) has gained much popularity in recent years, where deep neural networks (DNNs) have achieved considerable success in many tasks, such as computer vision (CV), speech, and natural language processing (NLP). The main characteristics that favored the rise of these algorithms are i) the use of large annotated datasets, and ii) networks with deep structures [7]. However, the first requirement cannot be fulfilled in some natural settings like medicine, biology, and chemical engineering for several reasons. First, the resource can be limited. Obtaining and labeling data can be costly and might take an extended period. Therefore, it is unrealistic to have large datasets under such conditions. Secondly, the standard deeper structure means a model with higher complexity and a more significant number of trainable parameters. This is highly prone to cause overfitting problems and poor results, especially when the trainable samples are limited. Finally, some standard DL networks are unsuitable for small data settings compared with big data scenarios since insufficient training samples can compromise the learning success [8]. Therefore, learning from a small dataset is highly challenging, and many unresolved problems still need to be solved in small dataset scenarios. Recent studies have shown that several sub-domains of DL are trying to solve the small dataset problem from different perspectives [8]. One is by trying to mitigate the necessity of big training data such as transfer learning, which aims to learn representations from one domain and then transfer the learned features to a similar and closely related domain [7]. Another approach is using surrogate data, which can be generated from random numbers to imitate the distribution of the original dataset [9]. The first approach is more prevalent in CV and NLP tasks since many large datasets can be used to train the models. The second approach is more common in time series analysis. The diagnosis of TB is an example of the application of DL to small dataset problems. In this paper, we present our work based on [2] and mainly focus on developing several DL-based networks to classify the input sensor signals and predict their corresponding labels.

Hypothesis Statement:

- This study proposes that employing lightweight DL models with reduced layers and parameters can be advantageous in scenarios where the available dataset is small.
- Additionally, alternative approaches such as graph neural networks (GNN) and transforming time series into images are expected to offer valuable contributions to addressing MTS problems.

Objective Statements:

- Utilize multiple data preprocessing techniques and provide a data preprocessing pipeline, such as sensor signal extraction, data normalization, data calibration, and sensor selection, with the aim of applying those techniques to similar tasks involving MTS sensor signals.
- Propose various DL-based models with fewer layers and parameters compared to common complex DL architectures to classify the small TB dataset. And try to tackle the problem from different perspectives, such as utilizing GNN and transforming the MTS signals into 2D images.
- Compare the performance of the proposed models with several state-of-the-art methods commonly used in multivariate time series classification (MTSC) tasks.
- Discuss and analysis the results based on the experiments and reveal the research findings.

- To encourage further research on MTSC with small-dataset problems, provide an open-source of our work accessible at `https://github.com/ChenxiLiu6/TB-Classification.git`.

The structure of this thesis is as follows. In Chapter 2, we introduce the related work on nanomaterial-based sensors for disease diagnosis by using disease-related VOCs, and describe the sensors employed in this study and their working mechanisms. Besides, we also provide some essential background information on MTSC tasks and describe some state-of-the-art approaches for solving MTSC tasks. Next, in Chapter 3, we present the dataset we use in our study and the data preprocessing methods we employed. Then, we describe the four proposed DL-based classification models in detail, namely LSTM, CNN, GAF-CNN, and MT-MinCutPool, which are appropriate for small TB dataset classification problems. In Chapter 4, we introduce the evaluation metrics used in this study and the experimental setup for each model. Then, we present the results and performance of each model in terms of accuracy, sensitivity, specificity, and the area under the curve (AUC), and compare our proposed methods with several baseline MTSC methods. Finally, in Chapter 5, we discuss the experiment findings, draw our conclusions, and point out some possible directions for future work based on our research.

Chapter 2

Background

2.1 Related Work

Traditional detection methods for TB, including sputum microscopy, culture test, radiology, drug susceptibility testing, whole genome sequencing, and clinical signs/symptoms, have proven effective in acid-fast bacilli detection, point-of-care diagnosis, and cost efficiency. However, these approaches exhibit shortcomings, such as low sensitivity, time consumption, and poor efficacy, which may produce false-negative results, lack of differentiation between various bacterial strains, inability to detect bacterial viability, and unsuitability for resource-limited settings. [10–12]. These limitations may delay TB diagnosis, which may further exacerbate infection severity, raise mortality risk, and enable bacilli transmission in the healthy population. Moreover, erroneous diagnosis can result in imprecise treatment, eventually leading to the development of drug-resistant in affected patients. [13] Therefore, the World Health Organization (WHO) has stated that there is an urgent need for a rapid, cost-effective, and sputum-free triage test to detect TB in real-time.

In addition, the importance of developing new diagnostic and detection technologies for the growing number of clinical challenges is rising each year. The analysis of disease-related VOCs represents a new frontier in medical diagnostics due to its noninvasive and inexpensive for illness detection. Specific VOC species and their concentration changes for each disease are unique and, thus, make them valuable biomarkers for disease detection [14] [15]. Spectrometry and spectroscopy techniques have demonstrated their efficacy in detecting VOCs directly from the headspace of the disease-related cells via urine, blood, skin, or exhaled breath. However, despite their effectiveness, these techniques are often hindered by their high cost, the level of expertise, and the time required to operate the sophisticated equipment necessary for sample analysis [16] [17]. To overcome these challenges, some researchers proposed a novel pathway that enables the use of sensor matrices based on nanomaterials as a clinical and point-of-care diagnostic tool. Nanomaterials have several advantages, including high sensitivity, fast response and recovery time, and synergetic properties when combined. Furthermore, nanomaterial-based sensors can be integrated into portable, low-cost devices through mass manufacturing, enabling noninvasive, easy-to-use, personalized disease diagnosis and follow-up. Existing studies [3] [14] [15] have shown the potential of nanomaterial-based sensors for VOC-based disease diagnosis.

Paper [14] reviewed two complementary approaches to profiling disease-related VOCs by nanomaterial-based sensors: selective and cross-reactive sensing. Our research is based on work [2], where they utilized the cross-reactive approach. This method broadly responds to various TB-specific VOCs emitted from the skin headspace. The VOC selectivity is gained through pattern recognition by obtaining information on the vapor's identity, properties, and concentration exposed to the sensor array.

In [3], they reported an artificially intelligent nanoarray for noninvasive diagnosis and classification of 17 diseases from the exhaled breath VOCs, work [2] employed a similar type of sensor, consisting of chemiresistive films containing spherical gold nanoparticles (GNPs; core diameter 3–4 nm) capped with different organic ligands, two dimensional (2D) random networks of single-walled carbon nanotubes (RN-SWCNTs) capped with different organic layers, and polymeric composites, The inorganic nanomaterials within the films are responsible for electric conductivity. In contrast, the organic component provides sites for VOCs adsorption. Upon VOC exposure, they are either absorbed onto the sensing surface or diffused into the sensing film, reacting with the organic phase or functional groups that cap the inorganic nanomaterials. This reaction/interaction results in volume expansion/shrinkage of the nanomaterial film, causing changes in the conductivity between the inorganic nanomaterial blocks. For sample collection, 40 sensors are employed, each with different functional groups capping the inorganic nanomaterial. This ensures that each sensor yields a distinct response to individual or patterned VOCs within the sample, generating a full metabolic profile of the tested state. Thus, resulting in a pattern of resistance changes detected by the sensor array to a given vapor.

Previous studies primarily concentrated on developing nanomaterial-based sensors for accurately detecting disease-related VOC patterns. However, they did not furnish comprehensive details on the applied classification procedures that procured the results. Furthermore, the dependability and progress of discriminant data classifiers cannot be ensured. Besides, existing ML approaches are more prevalent in large dataset scenarios. However, approaches such as transfer learning is widely used in small dataset settings as introduced in Chapter 1; there is currently a deficiency of similar and extensive datasets that can be utilized as the source domain in transfer learning for our task. Therefore, it is critical to developing dependable and suitable ML models pertinent to data-deficient problems that can be combined with other domains and foster their development.

2.2 Time series classification

The rapid expansion of data availability has led to the development of time series classification (TSC) in a wide range of fields ranging from human recognition [18], electronic health records [19] to acoustic scene classification [20] and stock market prediction [21]. Thus TSC has drawn the attention of a large number of researchers. The definition of a TSC task can be categorized into two types:

Definition 1. A univariate time series $X = \{x_1, x_2, \cdots, x_T\}$ is an ordered set of real values with timestamps. The length of X equals the number of real values T.

Definition 2. A dataset $D = \{(X_1, Y_1), (X_2, Y_2), \cdots, (X_N, Y_N)\}$ consists of a collection of N pairs of (X_i, Y_i), where (X_i, Y_i) is the ith sample, where X_i is either a univariate or multivariate time series accompanied by Y_i as its one-hot label vector.

The TSC task aims to train a classifier over dataset D to map from the time series inputs to a probability distribution over class labels. Our task can be categorized as an instance of the MTSC problem, where each sample comprises a set of MTS inputs denoted as X and a single corresponding label represented as Y.

2.3 Encoding Time Series as Images

The Gramian Angular Field (GAF) is one of the most widely used frameworks for encoding univariate time series as 2D images [22]. This approach has recently gained popularity due to its ability to capture cyclical patterns and correlations present in the original time series data, thus enabling researchers to take advantage of the success of deep learning architectures in CV and transfer it into the time series domain. The GAF transformation mainly involves two main steps: encoding the univariate time series into polar coordinates and then computing the Gramian matrix of the encoded data.

Before GAF transformation, the input time series $X = \{x_1, x_2, ..., x_n\}$ first needs to be normalized within the interval $[-1, 1]$ by

$$\tilde{x_i} = \frac{(x_i - max(X) + (x_i - min(X)))}{max(X) - min(X)}\,. \tag{2.1}$$

Then in the first step, the normalized time series $\tilde{X} = \{\tilde{x}_1, \tilde{x}_2, ..., \tilde{x}_n\}$ of n real-valued time steps can be represented in polar coordinates by encoding the value as the angular cosine and time stamp as the radius using

$$\begin{cases} \phi = arccos(\tilde{x_i}), -1 \leq \tilde{x_i} \leq 1, \tilde{x_i} \in \tilde{X}\,, \\ r = \dfrac{t_i}{N}, t_i \in \mathbb{N}\,. \end{cases} \tag{2.2}$$

The equation presented above defines the value of the time stamp t_i and incorporates a constant factor N to regulate the range of the polar coordinate system. In the second step, pairs of angular values from the polar coordinate representation are taken, and their outer product is calculated according to

$$G = \begin{pmatrix} \cos(\phi_1+\phi_1) & \cos(\phi_1+\phi_2) & \cdots & \cos(\phi_1+\phi_n) \\ \cos(\phi_2+\phi_1) & \cos(\phi_2+\phi_2) & \cdots & \cos(\phi_2+\phi_n) \\ \vdots & \vdots & \ddots & \vdots \\ \cos(\phi_n+\phi_1) & \cos(\phi_n+\phi_2) & \cdots & \cos(\phi_n+\phi_n) \end{pmatrix} \tag{2.3}$$

These outer products are then aggregated to form a Gramian matrix, which can be visualized as a 2D image. The resulting GAF image is a compact and information-rich representation of the original time series data. It captures the cyclical patterns and correlations present in the data and allows for the application of a wide range of image processing techniques for subsequent analysis.

2.4 The Long Short Term Memory Network

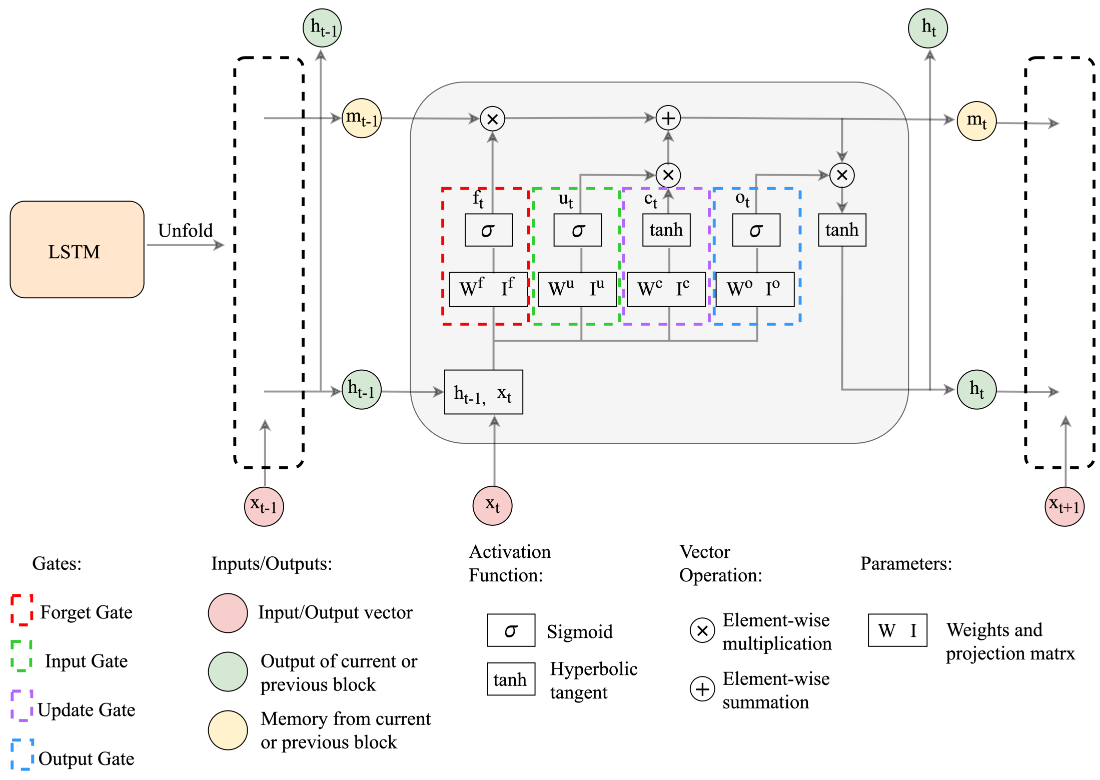


Figure 2.1: The structure of LSTM network.

A recurrent neural network (RNN) is a neural network that can simulate discrete-time dynamical systems with an input x_t, a hidden state h_t, and an output y_t [23]. The dynamical systems can be defined by:

$$h_t = f_h(x_t, h_{t-1}) = \tanh(Wh_{t-1} + Ix_t) \tag{2.4}$$

$$y_t = f_o(h_t) = \text{softmax}(Wh_t) \tag{2.5}$$

where the subscript t represents the time step. f_h, f_o are the state update function, and output function respectively. where f_h can use the hyperbolic tangent function $\tanh(\cdot)$, while the output function can usually use the $\text{softmax}(\cdot)$ function that can output a valid probability distribution as the model prediction. W, I represent the recurrent weight matrix and the projection matrix respectively, which serve as the parameters of the functions.

However, RNNs are often faced with vanishing gradient problems. Long short-term memory is an improved version of RNN [24] incorporating gating functions in the dynamical system to overcome this problem [25]. In a typical LSTM architecture, a memory vector $\mathbf{m}$, an LSTM hidden state vector $\mathbf{h}$, and the input $\mathbf{x}$ are employed to update the state and generate output at every time step, which can be expressed more precisely according to [26] by

$$\begin{aligned}
\mathbf{g}^u &= \sigma(\mathbf{W}^u\mathbf{h}_{t-1} + \mathbf{I}^u\mathbf{x}_t) \\
\mathbf{g}^f &= \sigma(\mathbf{W}^f\mathbf{h}_{t-1} + \mathbf{I}^f\mathbf{x}_t) \\
\mathbf{g}^o &= \sigma(\mathbf{W}^o\mathbf{h}_{t-1} + \mathbf{I}^o\mathbf{x}_t) \\
\mathbf{g}^c &= \tanh(\mathbf{W}^c\mathbf{h}_{t-1} + \mathbf{I}^c\mathbf{x}_t) \\
\mathbf{m}_t &= \mathbf{g}^f \otimes \mathbf{m}_{t-1} + \mathbf{g}^u \otimes \mathbf{g}^c \\
\mathbf{h}_t &= \tanh(\mathbf{g}^o \otimes \mathbf{m}_t)
\end{aligned} \tag{2.6}$$

where W, I, g denote the recurrent weight matrices, projection matrices, and activation vectors, respectively, and the superscripts $\{u, f, o, c\}$ represent the input, forget, output, and the cell state gates. The activation function includes the hyperbolic tangent denoted by $\tanh(\cdot)$, and the logistic sigmoid function $\sigma(\cdot)$. $\otimes$ represents the element-wise multiplication. The computation pipeline for the LSTM network is depicted in Figure 2.1.

2.5 The Graph Neural Network Model

Graph neural networks (GNNs) are a general framework for modeling deep neural networks using graph information, such as nodes, edges, and graph structures. The goal with GNNs is to generate representations of nodes based on the graph structure and any feature information of the graph. In recent years, GNNs have become increasingly popular due to their ability to capture complex relationships and dependencies among elements in the graph. It has been successfully applied to various problems, such as node classification, graph classification, etc. In Section 3.6, we first transformed the MTS signal into graph-structured nodes and then employed GNN to classify the node representation.

2.5.1 Spectral Clustering and MinCutPool

Spectral clustering (SC) is a widely-used technique to identify strongly-connected communities within a graph [27]. In the context of GNNs, it can be employed to perform pooling operations that aggregate nodes belonging to the same cluster, which can effectively reduce the dimensionality of the input graph by replacing groups of nodes with a smaller set of nodes, each of which represents a cluster of similar nodes. The new coarsened graph can help to improve computational efficiency while enabling more effective modeling of high-level graph features. However, most SC relies on the eigendecomposition of the graph Laplacian matrix to project the graph nodes into a lower-dimensional space [28], which can be very expensive, and the result is also graph-specific. Therefore, to overcome the limitations of SC, Bianchi et al. [27] propose a novel graph clustering method that enables rapid computation of cluster assignments without the need for spectral decomposition. Then they apply the generated cluster assignment matrix S as an input to the MinCutPool layer, which serves to coarsen the graph by aggregating nodes that belong to the same cluster while preserving the salient features of the original graph.

2.6 Binary Cross Entropy

The binary cross entropy (BCE)function is commonly used in binary classification tasks and it can be expressed as:

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot \log(p(y_i)) + (1-y_i) \cdot \log(1-p(y_i)) \tag{2.7}$$

where y is the label (0 for active TB samples and 1 for non-TB samples), and $p(y)$ is the predicted probability of the samples being non-TB samples for all N samples. We assume the samples follow the distribution $\mathbf{p(y)}$, and they are coming from the true (unknown) distribution $\mathbf{q(y)}$. To make the two distributions as close to each other as possible, a measure of dissimilarity between both distributions is needed, and the Kullback-Leibler divergence (KL Divergence) enables this measurement, which can be expressed as:

$$D_{KL}(p||q) = H_p(q) - H(q) \tag{2.8}$$

where the cross-entropy is

$$H_p(q) = -\sum_{c=1}^{C} q(y_c) \cdot \log(p(y_c))$$

and the entropy is

$$H(q) = -\sum_{c=1}^{C} q(y_c)\log(q(y_c))$$

Then the KL-Divergence can be expressed as

$$D_{KL}(p||q) = \sum_{c=1}^{C} q(y_c) \cdot [\log(q(y_c)) - \log(p(y_c))] \tag{2.9}$$

The closer between both distributions $\mathbf{p(y)}$ and $\mathbf{q(y)}$, the lower $D_{KL}(p||q)$ is, so as the cross-entropy $H_p(q)$. Therefore, minimizing $H_p(q)$ will give us a $\mathbf{p(y)}$ that is the closest distribution to the actual distribution $\mathbf{q(y)}$. So we will minimize the cross-entropy between both distributions $H_p(q)$. Since we have N total number of samples and the probability for each sample $q(y_i)$ is $\frac{1}{N}$, then the cross-entropy $H_p(q)$ becomes $H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} \log(p(y_i))$.

In this binary classification task, the labels are either 0 or 1, $y_i = \log(p(y_i))$ when the label is 1, and $y_i = \log(1 - p(y_i))$ when the label is 0. It means for each non-TB sample (y=1), it adds $\log(p(y))$ to the loss. Conversely, for active TB samples (y=0), it adds $\log(1 - p(y))$ to the loss. After plugging this manipulation into the original formula, the cross-entropy becomes:

$$\begin{aligned} H_p(q) &= -\frac{1}{N_{active} + N_{non}}[\sum_{i=1}^{N_{active}} \log(1 - p(y_i)) + \sum_{i=1}^{N_{non}} \log(p(y_i))] \\ &= -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot \log(p(y_i)) + (1 - y_i) \cdot \log(1 - p(y_i)) \end{aligned}$$

And we get the final binary cross-entropy expression.

2.6.1 Dynamic Time Warping (DTW)

The Euclidean distance metric is a widely used approach for measuring distance. However, its susceptibility to sensitivity and distortion in time axis [29] has led us to explore alternative options. Among the existing distance measurement approaches, Dynamic Time Warping (DTW) is the best solution for time series tasks due to its ability to align two similar time series that are locally out of phase in a non-linear manner [30]. Therefore, we employed DTW to measure the similarities between each time series sensor signal as it provides great flexibility in handling such scenarios.

DTW is a versatile method that can handle time series of both equal and unequal lengths. However, in our task, we performed middle part extraction during the data preprocessing step, resulting in time series with equal lengths. Therefore, we will describe the computation of DTW similarities using equal-length time series.

When using DTW to compute the distance between two-time series sensor signals, denoted as $P = (p_1, p_2, \cdots, p_n)$ and $Q = (q_1, q_2, \cdots, q_n)$, the following steps are undertaken:

1. **Initialization:** We will first need to construct a distance matrix $\mathcal{D} \in \mathbb{R}^{n \times n}$ which

is shown as follows:

$$\mathcal{D} = \begin{pmatrix} d(p_1, q_1) & d(p_1, q_2) & \cdots & d(p_1, q_n) \\ d(p_2, q_1) & d(p_2, q_2) & \cdots & d(p_2, q_n) \\ \vdots & \vdots & \ddots & \vdots \\ d(p_n, q_1) & d(p_n, q_2) & \cdots & d(p_n, q_n) \end{pmatrix}$$

where each element $d(p_i, q_j)$ of distance matrix $\mathcal{D}$ is the distance between the i-th point of P and the j-th point of Q. Specifically, $d(p_i, q_j) = (p_i - q_j)^2$.

2. **Constraints:** A warping path $\mathcal{W}$ is a contiguous set of matrix indexes that defines a mapping between P and Q: $\mathcal{W} = \{w_1, w_2, \cdots, w_{2n-1}\}$, which is subject to the following constraints [31]:

   (a) $w_1 = d(p_1, q_1)$ and $w_{2n-1} = d(p_n, q_n)$

   (b) Given $w_{k+1} = d(p_i, q_j)$ and $w_k = d(p_{i'}, q_{j'})$:
   $0 \leq i - i' \leq 1$ for all $i < 2n - 1$
   $0 \leq j - j' \leq 1$ for all $j < 2n - 1$

3. **Computation:** Given that there may exist multiple warping paths, the objective of the DTW algorithm is to find the path $\mathcal{W}$ that can minimize the warping cost, which can be mathematically expressed as:

$$DTW(P, Q) = \min_{W=\{w_1, w_2, \cdots, w_{2n-1}\}} \sum_{k=1}^{2n-1} w_i \tag{2.10}$$

The optimal distance $\mathcal{W}$ is obtained by solving the following recurrence relation:

$$DTW(p_i, q_j) = d(p_i, q_j) + \min \begin{cases} DTW(p_{i-1}, q_j) \\ DTW(p_i, q_{j-1}) \\ DTW(p_{i-1}, q_{j-1}) \end{cases} \tag{2.11}$$

where the final distance is $DTW(p_{2n-1}, q_{2n-1})$. Thus, we finally obtain the DTW similarity distance between two equal-length time series following the steps above.

Chapter 3

Nano-material based sensor signal processing and classification methods

3.1 Introduction

In this chapter, we aim to improve the accuracy of DL models for TB classification. To achieve this goal, we introduce a comprehensive data preprocessing pipeline and propose four different DL models. The success of DL on any given task depends heavily on the quality and representation of the data. Thus, we begin by describing the data preprocessing pipeline in Section 3.2, which involves removing irrelevant and redundant information, normalizing the data, and calibrating the signal. We also explain how we select the best sensor signals while removing noisy and unreliable ones.

With the preprocessed data, we then explore four DL models for our TB classification task. We present each model in a dedicated section, highlighting its unique features and contributions to the classification task. The first two models directly feed the preprocessed data into proposed lightweight LSTM and CNN models, which we describe in Sections 3.3 and 3.4, respectively. In Section 3.5, we propose the third model, GAF-CNN, which first transforms the MTS sensor signals into 2D images using GAF and then uses the images as feature inputs for subsequent classification. Lastly, we introduce the fourth model MT-MinCutPool, in Section 3.6, which explores the MTS data from a graph perspective. In this section, we first obtain a global representation of the MTS and then cluster similar nodes together to coarsen the graph. Finally, we classify the coarsened graph-level embedding as the final output labels. Through these models, we hope to provide insights into the potential of the proposed DL models in enhancing the accuracy of the TB classification.

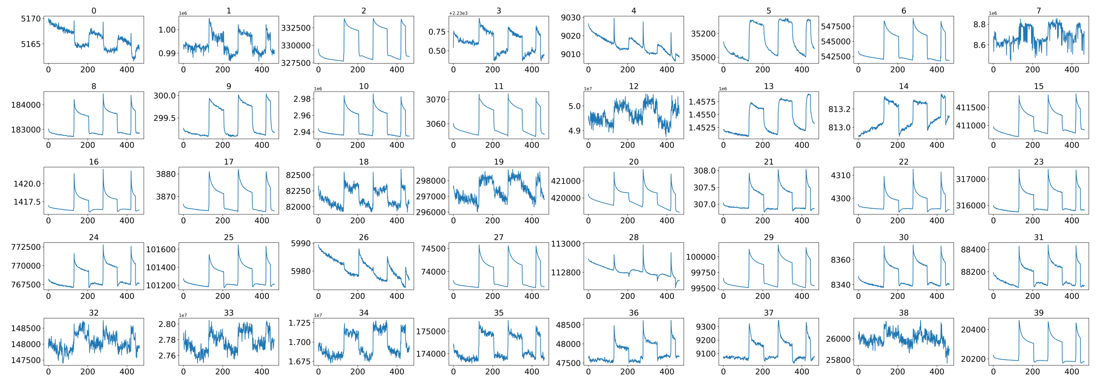


Figure 3.1: The original 40 sensor signals corresponding to one sample before data preprocessing.

3.2 Data Preprocessing

3.2.1 Dataset Description

In [2], during the sample collected phase, the study included 928 subjects between the ages of 22 and 60. To establish a robust method for TB detection and eliminate the influence of environmental factors on the samples, the samples and analysis were conducted in three different locations, including New Delhi in India, Cape Town in South Africa, and Riga in Latvia. The study population consisted of 461 healthy controls, including both healthy volunteers and confirmed non-TB samples, denoted by label 1, and 467 newly diagnosed and confirmed pulmonary-active TB patients, each represented by label 0.

As TB involves many bodily systems, it isn't easy to be diagnosed with only one unique biomarker when collecting VOCs from the skin headspace. To overcome the lack of specific biomarkers [32], a combination of 40 non-selective nanomaterial-based sensors is used simultaneously to detect a variety of TB-specific VOCs, thus providing a comprehensive metabolic assessment of each sample's tested state. Then, each sensor will translate the detected VOCs into resistance signals with a duration of T time steps. The raw sensor signals' time length T corresponding to each sample may differ. The translated 40 original sensor signals corresponding to one sample are shown in Figure 3.1, and each raw sensor signal is like the one shown in Figure 3.2. Figure 3.4 displays the change in the sensor resistivity (i.e., $\Delta R_{end}/R_b$) under different storage conditions, including vacuum, ambient air, and pure nitrogen respectively for nine months. As can be seen, the resistivity change in the room air is the largest (35%), followed by vacuum (19%), and pure nitrogen (17%) [2]. Therefore, it supports the observation of the sensor signal's characteristics in Section 3.2.2. During the sample measurement, the wearable device is applied directly to the skin on the sample's chest and anterior arm regions. (Figure 3.3).

The dataset can be represented by (X, Y), where $X \in \mathbb{R}^{N\times n\times T}$ represents the samples, and $Y \in \mathbb{R}^N$ is the corresponding label which is either 0 or 1. Note that N is the number of samples, n denotes the number of sensors (or nodes), and T represents the total time steps of each resistance signals.

3.2.2 Middle Part Signal Extraction

The multivariate time series sensor signals are recorded under three different conditions, including vacuum, pure N_2, and sample exposure [2] (see Figure 3.2). The sensor's baseline responses were recorded for 5 min in a vacuum, 5 min under pure nitrogen (99.999%), 5 min in a vacuum, and 5 min under the sample exposure, followed by a further 3 min under vacuum conditions [2]. Only the resistance signals obtained under the ambient air on skin samples are valuable signals that need to be extracted from the full signals before applying DL models, which corresponds to the middle peak part of the signal, where the signal curve presented the characteristics of a flat line, a rising peak, and then a return to a flat state.

To achieve this, we first computed the three peak points of the entire signal denoted by P1, P2, and P3, respectively. Since the length of the middle part signal of each sample varied and could fluctuate within a certain range, setting a fixed length in advance and moving P2 or P3 separately to intercept the signal was not feasible. To overcome this problem and obtain the start and end points of the rough middle part signal extracted from each sample, we moved P2 and P3 to the left by 40 and 10 time steps, respectively.

The reason for the different time steps used to move P2 and P3 was to account for the varying lengths observed in the roughly extracted middle part signal across samples. To ensure that the final middle part signal obtained from each sample was of equal length, we first needed to find the minimum length denoted by l_{min} among all extracted signals. We achieved this by shifting P3 to the left by a shorter distance 10, reserving space for the final signal interception to a consistent length. Accordingly, the final middle part signal of each sample was defined as spanning the starting index and the minimum length by $[\text{start}, \text{start} + l_{min}]$. This was a crucial preprocessing step that helped standardize the signal features and enhance the accuracy of the subsequent analysis.

3.2.3 Data Normalization

Data normalization is one of the essential preprocessing approaches [33]. However, during the signal generation and collection phase, due to different sampling times and the external environments (e.g., pressure, temperature, humidity, etc.) and the characteristics of each sensor itself [34], the resistance values measured by each sensor corresponds to each sample have various starting points and large ranges in $O(10^7)$. Therefore, to improve the data quality, we need to normalize the data to the same scale and give each feature a uniform contribution. Inspired by the normalization methods from [33],

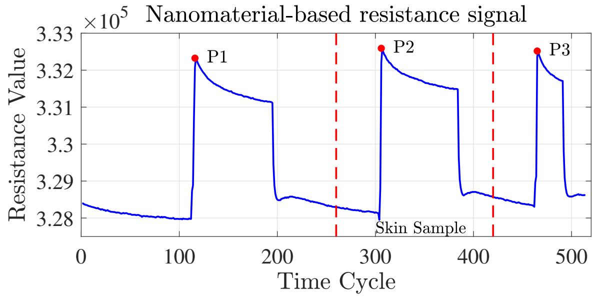


Figure 3.2: Nanomaterial-based sensor resistance signals, where the x-axis is the time cycles, and the y-axis is the resistance values. Useful signals are obtained under the exposure of the skin sample (within the red dotted line).

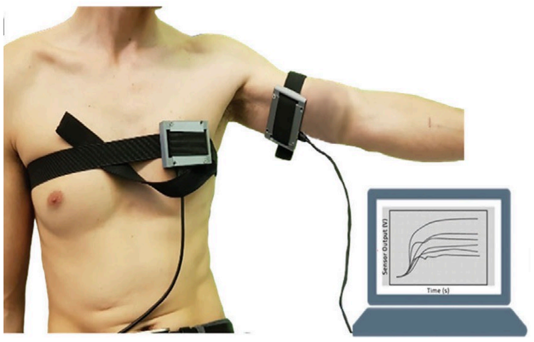


Figure 3.3: Wearable sensor devices on a volunteer's chest and anterior arm.

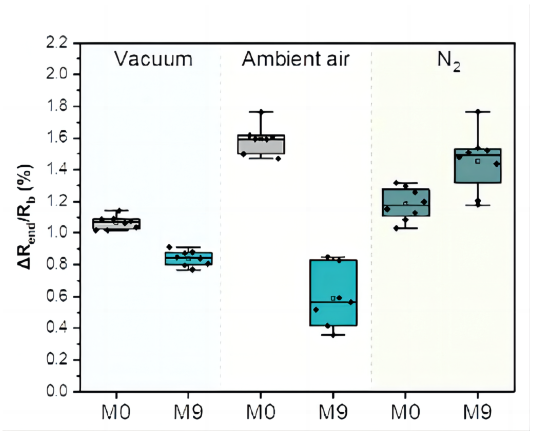


Figure 3.4: The change in the sensor resistivity (i.e. $\Delta R_{end}/R_b$) for different storage conditions at the starting point (M0) and after 9 months (M9).

we use a transformed median normalization method to preserve better relationships of the resistance values, which is defined as follows:

$$X' = \frac{X}{\text{med}(X_{[0,30]})} \tag{3.1}$$

$$\hat{X} = X' - \min(X') \tag{3.2}$$

where X is the input signal, min(·) denotes the minimum resistance value from all timestamps, and med(·) denotes the median value of the input. We first divide each X by the median of its first 30 resistance values as shown in equation 3.1 and get X', then we obtain the normalized signal by shifting each X' by its minimum value as shown in equation 3.2. After the data normalization, all signals will have the same minimum point 0, and a smaller same scale between $[0, 1]$.

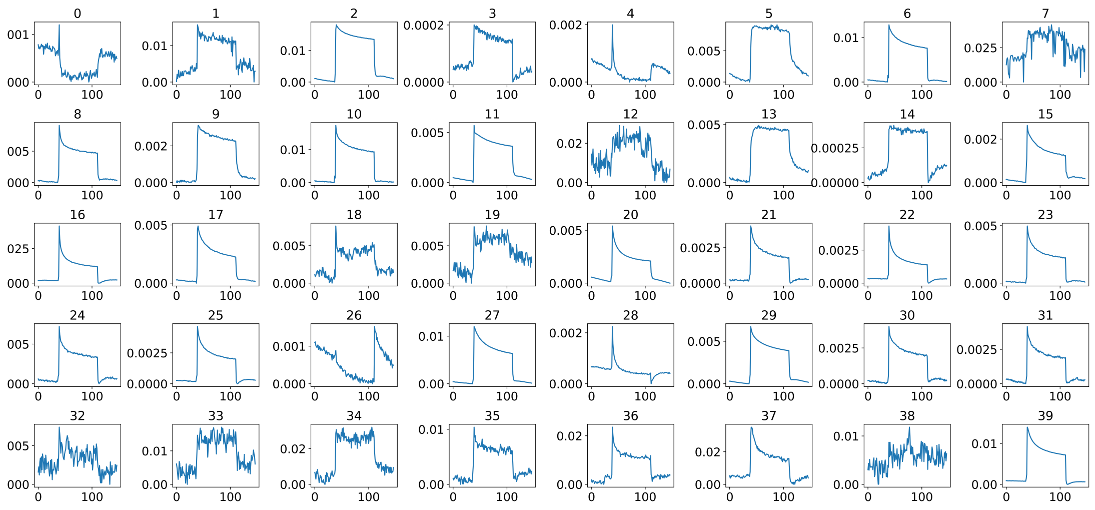


Figure 3.5: The original 40 sensor signals correspond to one sample after middle part extraction, data normalization, and data calibration.

3.2.4 Data Calibration

To supervise the sensor's functionality during the experiment and to overcome possible sensor response drift, therefore making the character of the sample resistance curve align with the original curve of the corresponding sensor, a baseline resistance is measured as a calibration signal before measuring the sample signals. The way we calibrate each sample signal is as follows:

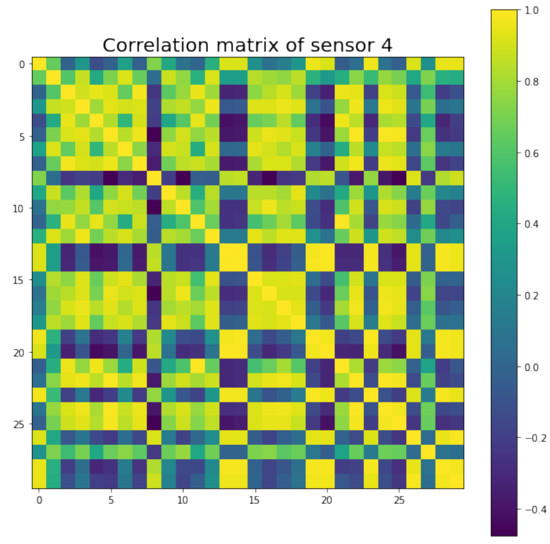
$$\hat{X} = \hat{X} \cdot \frac{\text{Peak}(\hat{C})}{\text{Peak}(\hat{X})} \tag{3.3}$$

where $\hat{X}$ and $\hat{C}$ are the normalized sample signal and its corresponding normalized calibration signal. We first compute the range ratio between $\hat{X}$ and $\hat{C}$. Since the minimum value for both normalized signals is 0, we only need to consider the peak point of each signal denoted by Peak$(\cdot)$ in this setting. Then each $\hat{X}$ is calibrated by multiplying its corresponding range ratio.

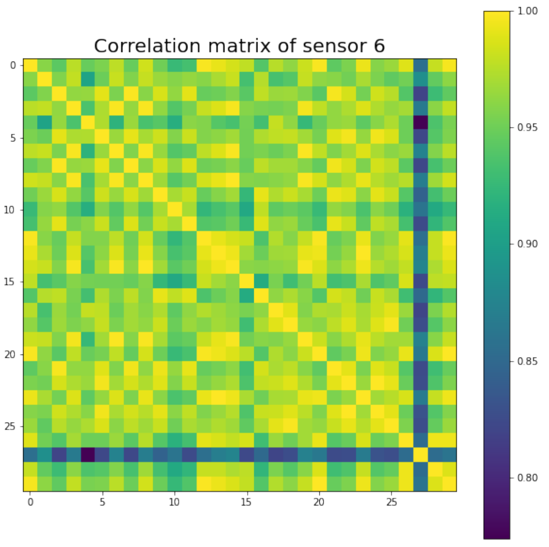
3.2.5 Sensor Selection by Using Pearson Correlation Coefficient Matrix

As wearable sensors have become more prevalent in settings that require reliability and accuracy, such as in healthcare and clinical diagnosis. Several sensors are usually combined together in order to overcome relative weaknesses of other sensors, such as sensor uncertainty, limited spatial coverage, imprecision, and malfunction [35], and also with the aim of improving the overall accuracy, robustness, and reliability of a decision-making process and finally enhance the overall performance of a system [36].

In our case, during the actual sample collection phase, each sample was measured by 40 sensors simultaneously, each useful sensor signal after middle part extraction, normalization, and calibration is shown in Figure 3.5, where some sensors were placed



(a) unstable sensor matrix heat map



(b) good sensor matrix heat map

Figure 3.6: Pearson correlation coefficient matrix heat maps for both unstable sensors (a) and good sensors (b).

on the subject's anterior arm, while others were placed on the subject's chest, as shown in Figure 3.3. However, not all of these 40 sensors have effective and stable signal outputs, as shown in Figure 3.5. Jagged signal fluctuations can be seen in some sensors that produce noisy, and unstable outputs. As a matter of fact, we cannot rely on all of the sensors that were used during measurements, and it is very important to select sensors that can produce a stable and clear signal output so that we can have a better representation of the entire system, thus improving the model performance in the subsequent decision-making process.

An indicator of a bad sensor is that the output signal is highly unstable, with large fluctuations and sawtooth patterns. Therefore, we use the Pearson correlation coefficient [37] to measure the signal similarities of each sensor in different samples. For bad sensors, the similarity coefficient is expected to be small, and for good stable sensors, the similarity coefficient will be large. To reduce the possible contingency of selection and obtain a robust selection result, while not excluding sensors that are critical to determining whether a sample is an active TB patient or not during the decision-making phase.

In the first step, the samples are divided into two groups, one group containing active TB patients, and the other group containing healthy controls. Then 10 lists of indices are generated from each group respectively, where each list contains 15 sample indices. Next, we iterate over from the 40 sensors to compute the Pearson correlation coefficient between different samples randomly selected corresponding to each sensor

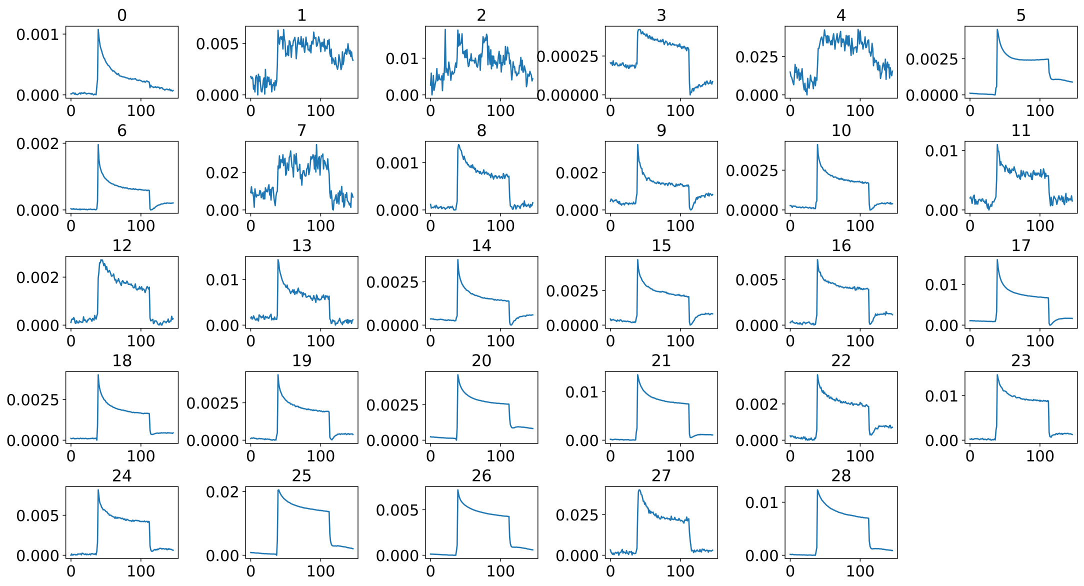


Figure 3.7: Selected 29 sensor signals of a sample after middle part extraction.

according to:

$$C_{xy} = \frac{\sum_{i=1}^{T}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{T}(x_i - \bar{x})^2 \sum_{i=1}^{T}(y_i - \bar{y})^2}} = \frac{\mathrm{cov}(x, y)}{\sqrt{\mathrm{var}(x) \cdot \mathrm{var}(y)}} \tag{3.4}$$

where C_{xy} is the Pearson correlation coefficient between signal x and signal y. Then the mean coefficient values from the randomly chosen samples are computed to represent each sensor's stableness, where sensors with a mean coefficient less than 0.65 are considered non-stable sensors, while the rest are considered good sensors to retain. The Pearson correlation coefficient matrices for both the bad sensors and the good sensors can be seen in Figure 3.6a 3.6b. In the end, both the active TB group and the healthy control group screened out the same 11 unstable sensors and kept the same 29 good sensors, which is shown in Figure 3.7. Note that in the initial stage, the dataset was divided into two distinct sets: the training-validation set and the test set. Subsequently, the sensor selection process was conducted independently on both the training-validation set and the test set. Notably, the outcome of this process revealed that both sets yielded identical sets of unstable sensors.

The stability of a signal is mainly attributed to the inherent characteristics of the sensor itself. However, it is essential to note that we cannot rule out the case that when dealing with a different dataset denoted as X_b, a sensor set S_a that is considered stable when used on a particular dataset X_a may not exhibit the same stability when employed on X_b. Therefore, to ensure a stable sensor selection process that generalizes to new data and guarantees accurate subsequent classification, it is advisable to calculate new sensor stabilization values and choose an alternative set of stable sensors denoted as S_b when using only X_b.

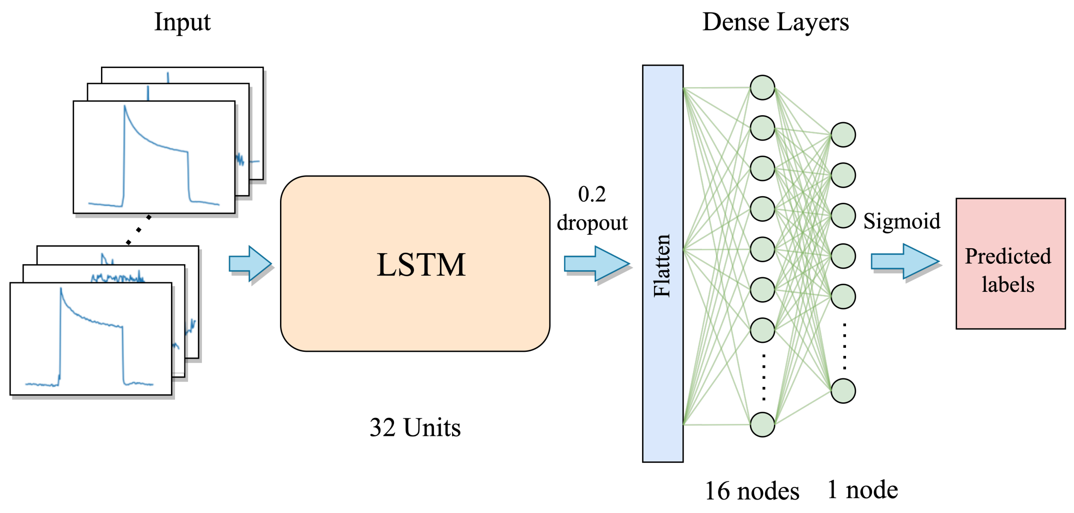


Figure 3.8: The proposed LSTM model architecture.

In cases when using both X_a and X_b as input, it is feasible to use an intersection of the stable sensors to achieve reliable classification outcomes. Specifically, $S_c = S_a \cap S_b$ can be used to select the common set of sensors shared between the two datasets.

3.3 Long Short Term Memory (LSTM) Network

The TB samples consist of multivariate time series sensor signals, and LSTMs are good at learning temporal dependencies [38]. Thus we first employ the LSTM network to learn the temporal features from the multivariate time series inputs. The first layer of the LSTM model architecture consists of an LSTM layer that includes 32 LSTM node units. The unfolded internal LSTM network structure is shown in Figure 2.1. Subsequently, a dropout layer with a dropout rate of 0.2 is applied. Following this, the resulting features are flattened into a vector, which is then forwarded into a dense layer containing 16 units, and activated by the ReLU activation function. The final layer is a single-node output dense layer, where the predicted labels are generated via the sigmoid activation function. The proposed LSTM model can be seen in Figure 3.8.

Due to the fact that large complex architectures with a big size of parameters are very prone to have overfitting problems when training with a small size dataset. Therefore, we only use a small number of units in each layer together with a dropout layer in our model, aiming to avoid or minimize the overfitting problem during the training phase. Finally, a total of 8481 parameters are included in the whole LSTM model. The detailed layer and parameter information is shown in Table 3.1.

During the training phase, we use the Adam optimizer and the BCE function as the loss function, which is a widely used loss function for binary classification problems,

Table 3.1: LSTM model layers and parameters.

Layer(Type)	**Output Shape**	**Activation**	**Parameter Number**
LSTM	(bs, 32)	–	7936
Dropout	(bs, 32)	–	0
Dense	(bs, 16)	ReLU	528
Dense	(bs, 1)	Sigmoid	17

the detailed description of BCE loss can refer to 2.6, and it can be expressed as:

$$BCE = -\frac{1}{N}\sum_{i=1}^{N}[y_i \cdot \log(p(y_i)) + (1 - y_i) \cdot \log(1 - p(y_i))] \tag{3.5}$$

where N is the total number of the input samples, y_i is the actual label for sample i, which is either 0 for active TB patients or 1 for non-TB samples in our case, and $p(y_i)$ is the corresponding predicted probability for the positive class. The BCE loss function provides a useful measure of the discrepancy between the true labels and the predicted probabilities. A lower value of the BCE loss indicates a better fit between the predicted probabilities and the true labels. Conversely, the loss will be large if the predicted probability is far from the true label.

The LSTM model input is (X, Y), where $X \in \mathbb{R}^{N\times n\times T}$ and $Y \in \mathbb{R}^{N}$, the detailed description of the input dataset can refer to 3.2.1, where T equals 147 after middle part extraction, and n equals 29 which represents the number of stable sensors after the sensor selection.

3.4 Convolution Neural Network (CNN)

In recent years, deep learning (DL) has been successfully applied in various domains, including image recognition problems, natural language processing tasks, etc. In light of the tremendous success of DL architectures in these different domains, researchers have begun adopting them for time series analysis as well [39].

Most deep learning-based TSC methods can be divided into two types: generative and discriminative [40]. Generative methods, characterized as model-based methods in the TSC community, are designed to find a suitable time series representation before training a classifier. In contrast, discriminative methods directly learn the mapping between the raw time series and the class probability distributions. The implementation of generative models is more complex than that of discriminative models, while the performance could be better. Thus the researchers focus primarily on discriminative models, notably on end-to-end approaches for TSC classification tasks [41].

According to the recent comprehensive review of DL-based TSC methods in [39],

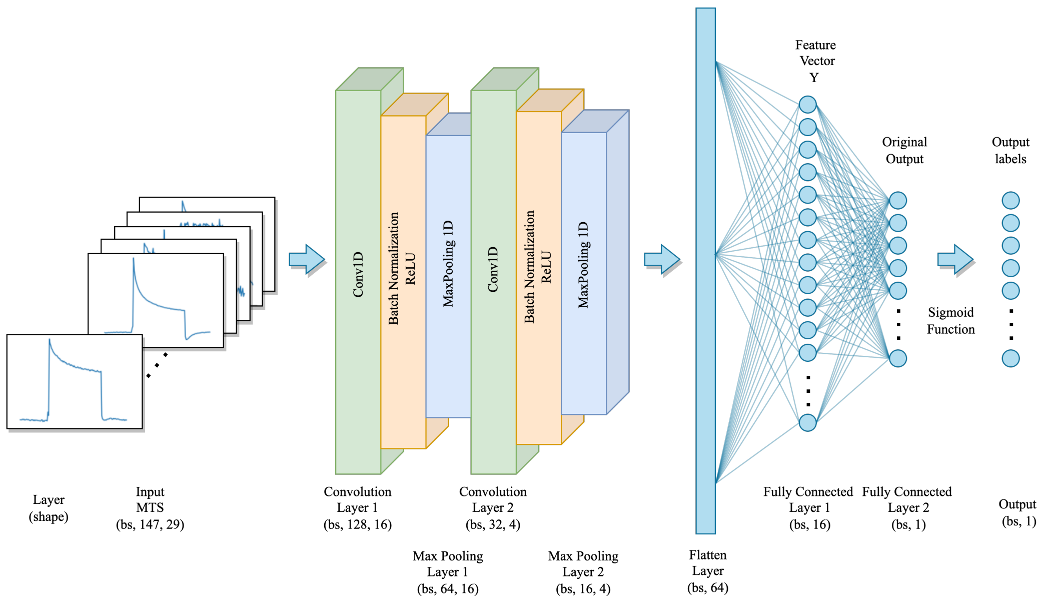


Figure 3.9: CNN model architecture.

CNN is found to be the most commonly used structure for TSC tasks due to its robustness and less training time than other DL architectures. Therefore, we propose a CNN architecture to classify the MTS sensor signals. The overall architecture of the proposed CNN is depicted in Figure 3.9.

The proposed CNN model takes a set of MTS sensor signals as input with the shape of (bs, T, n), where bs is the batch size, T equals 147 is the time steps of each sensor, and n equals 29 is the number of sensors used corresponds to each sample. The proposed model comprises two one-dimensional convolutional layers with different kernel sizes and output channels. Specifically, the first convolutional layer has a kernel size of 20 with 16 output channels, and the second convolutional layer has a kernel size of 2 with four output channels. Each CNN layer is followed by batch normalization and a ReLU activation function to model the interactive relationships between multivariate dimensions and the sequential information of time series. Furthermore, a one-dimensional max-pooling layer with a pooling kernel of size 2 is employed to reduce the spatial dimension of the output from the convolution layer while retaining features with stronger identification. The outputs of the convolutional layers are then flattened into a dense vector and then processed with two fully connected layers with 16 and 1 unit respectively. Finally, the classification results are obtained by computing the probability of each class by using the sigmoid activation function after the final fully-connected layer. The detailed parameter setting of the CNN model is shown in Table 3.2.

3.5 GAF-CNN

Inspired by [42], we first encoded each univariate time series sensor signals into polar coordinates using the amplitude and phase of the time series. This encoding captures

Table 3.2: CNN model parameter setting.

Layer	**Stride**	**Activation**	**Kernel Size**	**Input Shape**	**Output Shape**	**Parameter Number**
Conv1D	1	ReLU	20	(bs, 147, 29)	(bs, 128, 16)	9296
MaxPooling1D	2	-	2	(bs, 128, 16)	(bs, 64, 16)	0
Conv1D	2	ReLU	2	(bs, 64, 16)	(bs, 32, 4)	2116
MaxPooling1D	2	-	2	(bs, 32, 4)	(bs, 16, 4)	0
Flatten	-	-	-	(bs, 16, 4)	(bs, 64)	0
Dense	-	ReLU	-	(bs, 64)	(bs, 16)	1040
Dense	-	Sigmoid	-	(bs, 16)	(bs, 1)	17

Table 3.3: GAF-CNN model parameter setting.

Layer	**Stride**	**Activation**	**Kernel Size**	**Input Shape**	**Output Shape**	**Params**
Conv2D	1	ReLU	(5, 5)	(bs, 147, 147, 29)	(bs, 143, 143, 12)	8712
MaxPooling2D	2	-	(2, 2)	(bs, 143, 143, 12)	(bs, 71, 71, 12)	0
Conv2D	1	ReLU	(5, 5)	(bs, 71, 71, 12)	(bs, 67, 67,6)	1806
MaxPooling2D	2	-	(2, 2)	(bs, 67, 67, 6)	(bs, 33, 33, 6)	0
Flatten	-	-	-	(bs, 33, 33, 6)	(bs, 6534)	0
Dense	-	Sigmoid	-	(bs, 6534)	(bs, 1)	6535

the temporal structure of the time series and allows us to apply the GAF transformation to create the image representation, which has been previously introduced in Section 2.3. The transformed images have a fixed shape of (batch_size, height, width, channels), where the height and width are equal to the time steps 147, and the input channels are similar to the number of sensors (29 in this case).

The resulting images are then input to the proposed GAF-CNN model, which consists of two 2D convolutional layers, each followed by a ReLU activation function and a 2D MaxPooling layer. The first and the second convolutional layer has 12 and 6 filters, respectively, where the kernel size for both layers is (5×5). The 2D max pooling operation uses 2×2 windows with a stride of 2. After the second max pooling layer, the output is flattened and passed through a fully connected layer with a single neuron and a Sigmoid activation function. The resulting output value is between 0 and 1, representing the probability of the corresponding sample belonging to the positive class. Table 3.3 shows the model's detailed parameter setting.

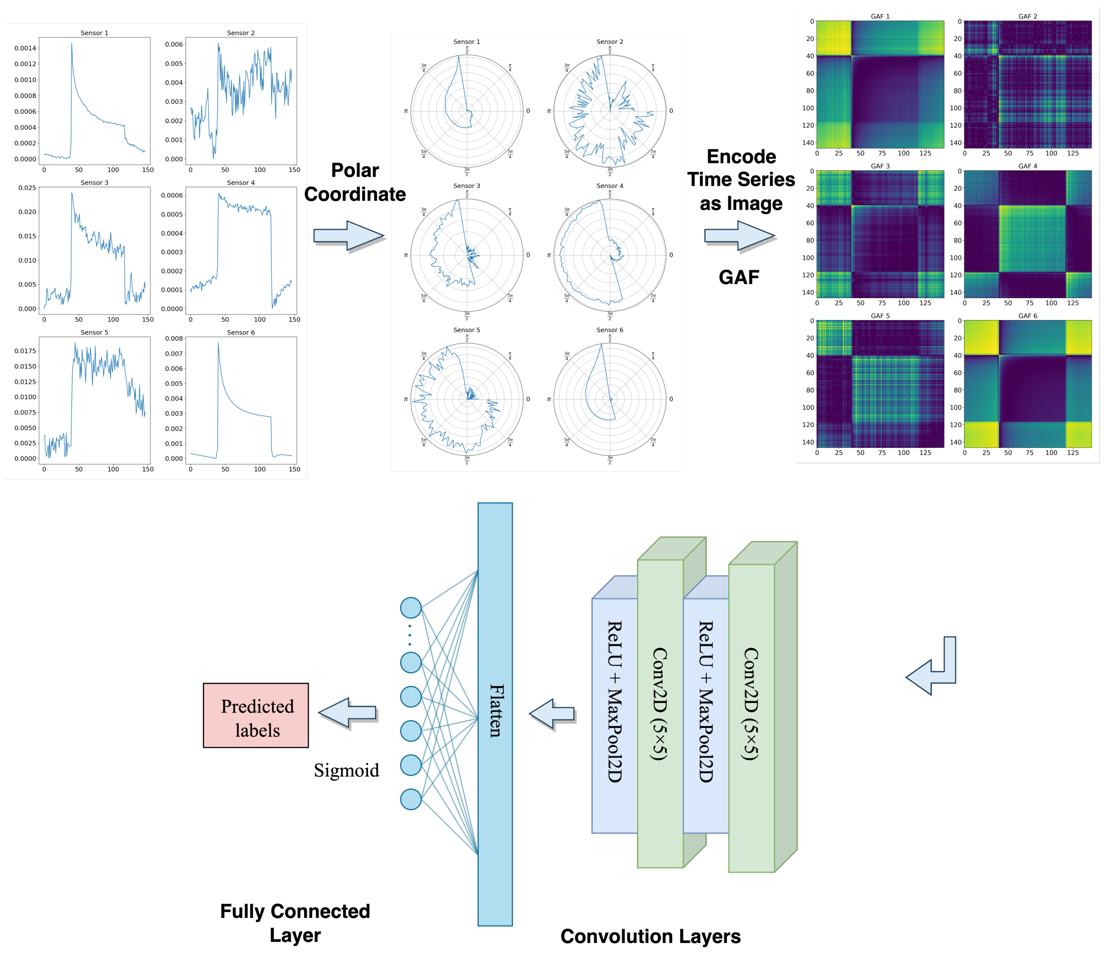


Figure 3.10: GAF-CNN model architecture.

3.6 MTSC with Graph Laplacian and MinCutPool

In [43], a novel graph pooling-based framework Multivariate Time Series Classification with Variational Graph Pooling (MTPool), is proposed to obtain an expressive global representation of MTS. This study is one of the few approaches that adopt GNN to solve the MTSC problem from a graph perspective. In contrast, others solve the MTSC problems using transformer-based frameworks (e.g., [44]), ensemble methods that ensemble over several univariate classifiers independently and then aggregates the predictions from each classifier to generate a single probability distribution for each TSC task (e.g., [45]), and variants of recurrent neural networks (e.g., Multivariate LSTM fully convolutional network (MLSTM-FCN) [46]).

This section proposes the MT-MinCutPool framework, a modified MTPool to solve the MTSC task. The main differences are that we employ the graph Laplacian matrix to construct the graph and use the MinCutPool to cluster similar nodes within a graph into one cluster to coarsen the graph. This section mainly contains five parts: Graph Structure Learning Using Laplacian Matrix, Temporal Feature Extraction, Spatial-Temporal

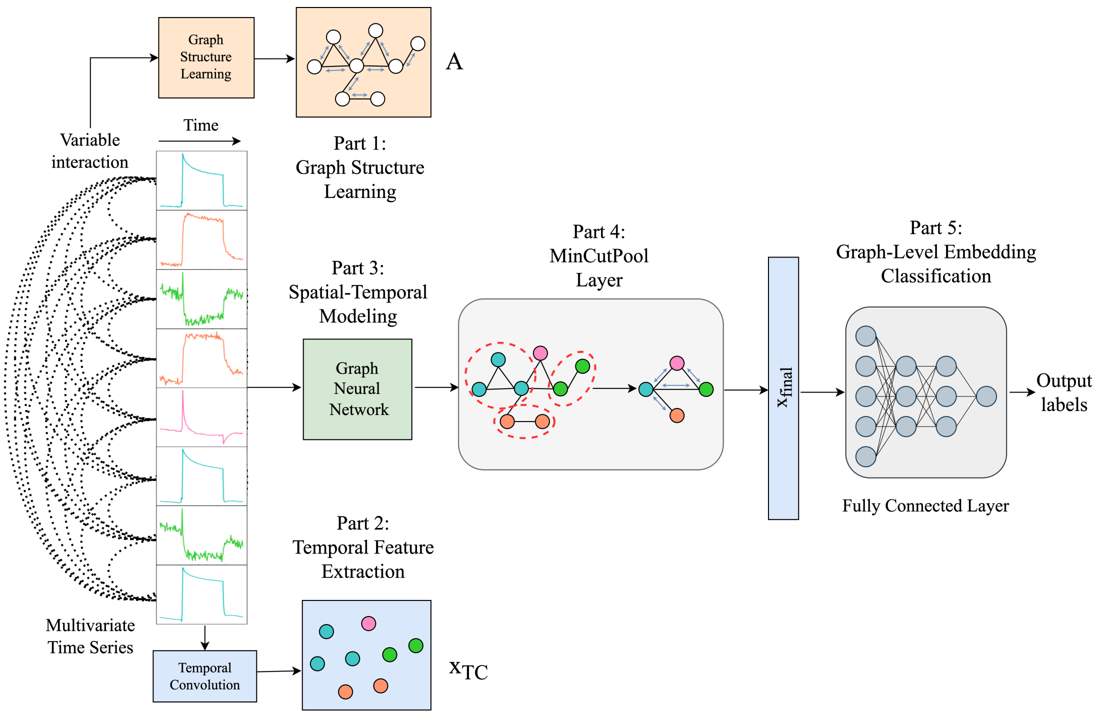


Figure 3.11: MTMinCutPool model architecture.

Modeling, Graph Coarsen by MinCutPool, and Graph-Level Embedding Classification.

3.6.1 Graph Structure Learning Using Laplacian Matrix

Graph-based representations and algorithms for handling structured data rely heavily on constructing meaningful graphs. Dong et al. [47] addressed the problem of learning the graph Laplacians, which is equivalent to learning the graph topologies, such that the input data will be transformed into graph signals with smooth variations in the resulting topology. Therefore, we also use the graph Laplacian matrix to represent the graph structure in the first part of the framework. The main steps of building the Laplacian matrix to represent the graph structure are shown in Figure 3.12

The dynamic time warping method (DTW) will likely produce a more reliable similarity assessment between two time series than other distance measurement methods, such as Euclidean distance, which matches timestamps regardless of feature values. For the input samples $X = \{x_1, x_2, \cdots, x_N\} \in \mathbb{R}^{n \times T}$, where N, n, T represent the number of samples, sensors, and total timestamps respectively. We first use the DTW method to calculate the distance matrix $DTW \in \mathbb{R}^{N \times n \times n}$ between each sensor within MTS samples. Then we'll construct an adjacency matrix A that can be used to calculate the following degree matrix D and the Laplacian matrix L. In the meantime, to improve the model's overall training efficiency, enhance the model's robustness, and reduce the impact of the noise introduced by the sensors, a threshold θ is set to make the distance

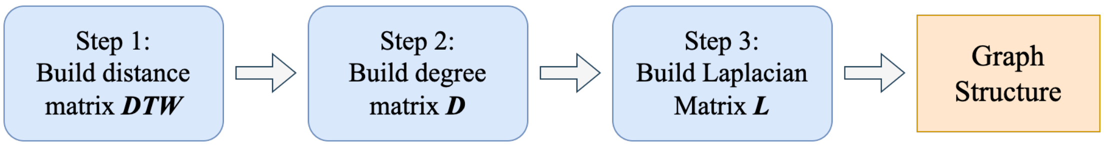


Figure 3.12: Summary of main steps of building the Laplacian matrix to represent the graph structure.

matrix DTW sparse:

$$A_{ij} = \begin{cases} 1, & \text{if DTW[i, j]} < \theta\,, \\ 0, & \text{if DTW[i, j]} > \theta\,. \end{cases} \tag{3.6}$$

After having adjacency matrix $A \in \mathbb{R}^{N \times n \times n}$, the elements of each row in matrix A are first added up to obtain vector $a \in \mathbb{R}^{N \times n}$, which is then diagonalized to generate degree matrix D. Finally, after having the adjacency matrix A and the degree matrix D. By definition, the graph Laplacian matrix L can be obtained through $D - A$. The detailed process of constructing the graph Laplacian matrix is shown in Algorithm 3.1.

3.6.2 Temporal Feature Extraction

The purpose of temporal convolution is to extract features along the time axis, as well as to design a temporal feature matrix $X_{TC} \in \mathbb{R}^{N \times n \times d}$, where d is the dimension of the new extracted features, that can serve as a strong reference for the subsequent classification step. When analyzing time series data, it is essential to consider both numerical values and long-term patterns. Therefore, to extract features from the time dimension and keep as much origin pattern as possible, we adopt the method employed in the prior work as presented in [43], which employs multiple convolutional neural network channels with varying kernel sizes:

$$X_{TC} = ||_{i=1}^{m} f_i = ||_{i=1}^{m} \sigma(W_i * X + b) \tag{3.7}$$

where $||_{i=1}^{m}$ denotes the concatenation operation that merges the feature maps generated by the first to the m-th CNN filters, where the subscript $i = \{1, 2, \cdots, m\}$ represent the specific CNN filter number. Each $f_i \in \mathbb{R}^{N \times n \times d_i}$ is the output tensor of each convolution layer containing the extracted temporal feature. And it is given by the convolution of the input tensor X with the learnable filter $W_i \in \mathbb{R}^{out \times in \times ks}$, where out is the number of output channels, in is the number of input channels, and ks is the kernel size of the filter. Then followed by an elementwise bias addition $b \in \mathbb{R}^{out}$, and an activation function $\sigma(\cdot)$ such as ReLU$(\cdot)$, to introduce non-linearity into the model. The operator $*$ denotes the convolution operation. After applying the convolutional layer with a given kernel size ks, the new sequence length can be computed according to $d_i = (T - ks)/s + 1$, where ks and s are the kernel size and stride step from the learnable filter W respectively. Finally, we get the temporal feature matrix $X_{TC} \in \mathbb{R}^{N\times n\times d_{TC}}$ by concatenating each convolved tensor f_i, where $d_{TC} = \sum_{i=1}^{m} d_i$ is the new sequence dimension. The concatenation of the extracted temporal features from various time scales provides a reliable reference for the subsequent classification task.

3.6.3 Spatial-Temporal Modeling

Spatial-temporal modeling is an essential task in many applications such as skeleton-based action recognition [48], traffic forecasting [49], etc. GNNs have demonstrated promising results on spatial-temporal modeling tasks. Their ability to directly apply filters on the graph nodes and their neighbors enables the model to learn representations that capture both the spatial dependencies and the temporal patterns of the data.

Graph convolution networks (GCNs) are a specific type of GNN designed to deal with graph-structured data. Therefore, in this part, we adopt GCN to operate on the input graph-structured data, typically represented as an adjacency matrix A that indicates whether an edge connects two nodes and a feature matrix X that contains the features of each node in the graph. The Graph Convolution operation can be defined as:

$$\tilde{X} = G(A, X_{TC}, W, b) = \sigma(A * X_{TC} * W + b) \tag{3.8}$$

where $\tilde{X}$ is the output feature matrix. $G(\cdot)$ is the graph convolution function, which takes the adjacency matrix A, the input feature matrix X_{TC}, a learnable weight matrix W of the convolutional filter, and a bias term b as input.

In the graph convolutional layer, the feature matrix $X_{TC} \in \mathbb{R}^{N\times n\times d_{TC}}$ is first multiplied by the adjacency matrix $A \in \mathbb{R}^{N\times n\times n}$, which in our case is the graph Laplacian matrix, it allows the information to be propagated from every single node to its neighbors in the graph. Then the output is linearly transformed by a weight matrix $W \in \mathbb{R}^{N\times n\times d_{out}}$ through another multiplication. After adding the bias term b, the new feature matrix is transformed using a nonlinear activation function such as ReLU or tanh, where $\sigma(\cdot)$ is the activation function. And finally get the resulting output $\tilde{X} \in \mathbb{R}^{N\times n\times d_{out}}$. The graph convolution process updates the graph's node representation, and the output node features $\tilde{X}$ can be used for the subsequent graph classification task.

3.6.4 Graph Coarsen by MinCutPool

To reduce the computational complexity of the GNN while at the same time preserving its expressive power, we combine the clustering method with the MinCutPool pooling method proposed in [27] in this part of our framework to coarsen the graph.

Let $\tilde{A} = D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \in \mathbb{R}^{N\times n\times n}$ be the symmetrically normalized adjacency matrix, where D is the degree matrix. The cluster assignment matrix S is computed using a multilayer perceptron (MLP) activated by a softmax function to map each node feature

x_i into the i-th row of matrix S. Specifically, it can be expressed as:

$$\bar{X} = GNN(\tilde{X}, \tilde{A}, W_{GNN}) \tag{3.9}$$

$$S = MLP(\bar{X}, W_{MLP}) \tag{3.10}$$

where $\tilde{X}$ is the matrix of the representation generated from the previous graph convolution layer, and $\bar{X}$ is the new feature matrix yielded by one or more subsequent graph convolution layers. W_{GNN} and W_{MLP} are the trainable parameters. Then the cluster assignment matrix S and the normalized adjacency matrix $\tilde{A}$ are fed into the MinCutPool layer to get the pooling node representations of the coarsened graph. The pooling process is computed according to:

$$A_{pool} = S^T \tilde{A} S; \quad X_{pool} = S^T X \tag{3.11}$$

where the entry $x^{pool}_{j,k}$ from $X^{pool}_i \in \mathbb{R}^{C \times d}$ is the sum of feature k among the items in cluster j from sample i, which is weighted by the cluster assignment scores from S, where $i = \{1, \cdots, N\}$ is the sample index, C is the number of clusters, and d is the new sequence feature length after graph convolution. $A^{pool}_i \in \mathbb{R}^{C \times C}$ is the coarsened symmetrical adjacency matrix, where the matrix element $a_{j,k}$ is the weighted sum of the edges between cluster j and k. The entries $a_{j,j}$ is the weighted sum of the node edges within a cluster. Each MinCutPool layer will generate a cut loss term $\mathcal{L}_u$ [27], and the weight parameters W_{GNN} and W_{MLP} are optimized by minimizing $\mathcal{L}_u$ during training, thereby increasing the likelihood of clustering similar nodes together.

3.6.5 Graph-Level Embedding Classification

After obtaining the graph-level embedding from the MinCutPool layer, the resulting output graph representation $X_{pool} \in \mathbb{R}^{N \times C \times d}$ is flattened into a vector form, referred to as $X^{N \times (C*d)}_{dense}$. Subsequently, X_{dense} is forwarded to the succeeding fully connected layers for the final classification.

3.7 Conclusions

This chapter presented a comprehensive data preprocessing pipeline and four different DL models for TB classification. The preprocessing data pipeline includes steps of middle part extraction, normalizing data, calibrating the signal, and selecting the best sensor signals while removing noisy and unreliable ones. The four proposed DL models are lightweight LSTM, CNN, GAF-CNN, and MT-MinCutPool. Each model has its unique features and contributions to the TB classification task. By exploring different DL models and optimizing the preprocessing data pipeline, we expect to achieve better performance in TB classification, which is crucial for the early detection and effective

Algorithm 3.1 Build Laplacian Adjacency matrix.

```
Input: $X \in \mathbb{R}^{N \times n \times T}$, $\theta$
Output: $L \in \mathbb{R}^{N \times n \times n}$
1: (1) Build Distance Matrix DTW
2: DTW ← empty matrix with shape $\mathbb{R}^{N \times n \times T}$
3: for $i = 1$ to $N$ do
4:     x ← X[i]
5:     distance ← empty matrix with shape $\mathbb{R}^{n \times n}$
6:     for $j = 1$ to $n$ do
7:         for $k = 1$ to $n$ do distance[j, k] ← dtw(x[j], x[k]) {dtw(·) is the dynamic time
   warping distance function}
8:         end for
9:     end forDTW[i] ← distance
10: end for
11: (2) Build Degree Matrix D
12: D ← empty array with shape $\mathbb{R}^{N \times n \times T}$
13: A ← int(DTW $< \theta$)
14: for $i = 1$ to $N$ do
15:     adj ← A[i]
16:     D[i] ← diagonal($\sum_{k=1}^{n}$ adj[k, :]) {diagonal(·) is the diagonalized function}
17: end for
18: (3) Build Laplacian Matrix L
19: L = D − A
```

treatment of TB. Overall, this chapter provides a foundation for future studies on TB classification using DL techniques.

Chapter 4

Experiments and Result Analysis

4.1 Introduction

This chapter presents the results of our proposed DL models from the previous chapter and compares them with various state-of-the-art baseline approaches commonly used in MTSC tasks.

In Section 4.2, we introduce the evaluation metrics that we use to assess model performance. We also outline the experimental evaluation process employed in our study, which includes dataset splitting, stratified cross-validation, and multiple 10 runs to ensure the reliability of our results. In Section 4.3, we provide a comprehensive display of the results obtained from both the proposed models and the baseline models. We present a result table, receiver operating characteristic (ROC) curves, and confusion matrices for each model. Finally, we discuss the findings and draw conclusions based on the results obtained. The structure of this chapter is intended to provide a logical and formal progression of our research methodology, experimental design, and empirical results, culminating in a clear and insightful analysis of our findings.

4.2 Experiments

4.2.1 Evaluation Metrics

To evaluate the performance of our proposed models with some state-of-the-art methods for the MTSC problem, we utilize several widely used binary classification metrics, including:

- Accuracy: It measures the proportion of the correct predictions among all the predictions made by the models.
- Sensitivity (True Positive Rate (TPR)): It measures the proportion of true positive models made by the model out of all actual positive samples. It indicates the ability of the model to correctly identify the positive samples, which is very

important for clinical settings. It indicates the ability of a model to identify negative samples correctly.

- Specificity (True Negative Rate (TNR)): It measures the proportion of true negative predictions made by the model out of all actual negative samples.
- AUC: It considers the performance of a classifier over all possible threshold values, taking into account both the TPR (sensitivity) and the false positive rate (FPR: 1−specificity).

The above metrics can be expressed by the following formulas:

$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{4.1}$$

$$\text{sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{4.2}$$

$$\text{specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}} \tag{4.3}$$

where TP, TN, FP, and FN represent True Positive, True Negative, False Positive, and False Negative, respectively. A higher value indicates better performance for all the evaluation metrics, while a low value indicates poor performance.

4.2.2 Experimental Setup

During the experiment phase, the dataset (X, Y) was partitioned into training, validation, and test sets to ensure a reliable evaluation of the model's performance. Specifically, 80% of the data is used for training and validation, while the remaining 20% is reserved for testing. This enabled the model to be trained and tuned on a subset of the data, with the held-out test set serving as an unbiased measure of its performance on unseen data. Since we only have a total of 928 samples, which is a relatively small dataset for DL classification tasks. To efficiently use the existing data to train the model and better represent each model's performance, we adopt the widely used technique of stratified cross-validation for model training and evaluation. Cross-validation presents numerous advantages over conventional methods, including mitigating overfitting issues, particularly prevalent in scenarios involving small datasets, improved data utilization, a robust performance evaluation, and the capability for hyperparameter tuning.

In our experiment, we employed a rigorous evaluation process to ensure the reliability of our results. Firstly, we set the number of folds to 5. Then during each training iteration, the model is trained on four-folds, i.e., 80% of the train-validation dataset. At the same time, the remaining held-out fold, i.e., 20% of the train-validation dataset, is reserved for validation. This process is repeated for each fold, with a different fold held out as the validation set each time. After training each fold with a single model, we saved and loaded the models and performed 5-fold cross-validation on the training and validation sets again. Each fold was evaluated using its corresponding trained model. We then selected the best-performing model based on its accuracy on the validation dataset and assessed it on the unseen test dataset using various metrics, including accuracy, sensitivity, specificity, and the AUC score. To further increase the robustness of our evaluation, we repeated this process ten times, each using a unique random seed to split the training and validation dataset. Finally, we computed the mean and the standard deviation of test accuracies, sensitivities, specificities, and AUCs, across the ten rounds to obtain the final performance and the confidence intervals (CI) of each model, where the true value of the parameter falls within the corresponding CI with 95% confidence level. This thorough evaluation process allowed us to ensure the reliability of our results and provide a more comprehensive understanding of the performance of each model.

The proposed LSTM, CNN, MTMinCutPool, and GAF-CNN are trained with 2000, 1000, 500, and 500 epochs, respectively, with a learning rate of $1e-5$, $5e-5$, $1e-6$, and $1e-6$. In all cases, the early stopping patience was set to 50, and a batch size of 32 was used. Each model's parameters are updated throughout the training process using the BCE loss and the stochastic gradient descent. In the case of the MTMinCutPool model, in addition to the BCE loss, the mincut loss, which is generated from the graph coarsen process in Section 3.6.4, is incorporated as an additional component of the overall loss function.

4.3 Results and Discussion

4.3.1 Main Results

The evaluation results of the proposed models and the baseline models are presented in Tables 4.1 and 4.2, respectively. The actual mean performance of each evaluation metric with their corresponding CIs for both the 4 proposed and the 3 baseline models are calculated and displayed in the last column of each table, which provides a high-level overview of each group's respective performance and also helps us to gain insights into the relative performance of each model. We highlighted the highest test scores for each metric in bold for both the baseline models and the proposed models.

Our experimental findings reveal that the mean performances of our proposed models outperform those of the state-of-the-art baseline approaches for all evaluation metrics. Specifically, the average accuracy, sensitivity, specificity, and AUC score of the proposed models are all higher than those of the baseline models. GAF-CNN demonstrates the best overall performance among all proposed models, achieving the highest test accuracy, sensitivity, and AUC score. The LSTM model exhibits the highest specificity. The GAF-Attention model performs best among all baseline models, with the highest accuracy, specificity, and AUC score. Additionally, the MLSTM-FCN model attains the highest sensitivity score. GAF-CNN has the highest accuracy of 0.639 and

Table 4.1: Train and validation accuracy, sensitivity, specificity, and AUC values for the proposed models.

		LSTM	CNN	GAF-CNN	MT-MinCutPool	Model Mean
Accuracy	train	0.646	0.916	0.659	0.688	–
	valid	0.69	0.687	0.692	0.69	–
	test	0.611 ± 0.02	0.606 ± 0.02	**0.639** ± 0.008	0.604 ± 0.013	0.615 ± 0.031
Sensitivity	train	0.618	0.931	0.766	0.76	–
	valid	0.675	0.715	0.78	0.756	–
	test	0.631 ± 0.073	0.694 ± 0.025	**0.777** ± 0.02	0.728 ± 0.039	0.71 ± 0.087
Specificity	train	0.676	0.901	0.544	0.612	–
	valid	0.704	0.658	0.597	0.619	–
	test	**0.594** ± 0.054	0.533 ± 0.028	0.532 ± 0.021	0.5 ± 0.037	0.54 ± 0.073
AUC	test	0.634 ± 0.016	0.657 ± 0.022	**0.692** ± 0.007	0.661 ± 0.011	0.661 ± 0.03

the highest sensitivity of 0.777 among all models. In contrast, the GAF-Attention model has the highest specificity of 0.637 and the highest AUC score of 0.695.

We employed the AUC-ROC curve to visualize each model's performance better. The ROC is a probability curve that plots the TPR against the FPR, and AUC represents the degree of each model's separability. The higher the AUC, the better the model distinguishes between positive TB patients and healthy controls. We can see from Figure 4.1 that the order of model performance from high to low (in terms of AUC-ROC) is as follows: GAF-Attention > GAF-CNN > MT-MinCutPool > CNN > MLSTM-FCN > LSTM > MTPool.

The training and validation accuracy and loss curves for each selected model with the highest accuracy score on the validation dataset are shown in Figures 4.2–4.5.

4.3.2 Discussion

The MT-MinCutPool model is modified according to the MTPool model. It performs relatively well compared to the proposed LSTM and CNN models since it has higher sensitivity and AUC scores. Furthermore, its accuracy, sensitivity, specificity, and AUC score are all higher than those obtained using the MTPool model. This might be due to the following two reasons. Firstly, in MT-MinCutPool, we use the Laplacian matrix to represent the graph topology, which contains more graph information than the correlation coefficient matrix employed in MTPool. Secondly, MinCutPool is designed to preserve important information by retaining highly connected nodes in the graph. Unlike other widely-used graph pooling approaches such as GraphSAGE [50], which

Table 4.2: Train and validation accuracy, sensitivity, specificity, and AUC values for the baseline models.

		MTPool	**GAF-Attention**	**MLST-FCN**	**Model Mean**
Accuracy	train	0.563	0.755	0.733	–
	valid	0.577	0.663	0.693	–
	test	0.517 ± 0.024	**0.631** ± 0.017	0.586 ± 0.026	0.578 ± 0.044
Sensitivity	train	0.682	0.78	0.757	–
	valid	0.683	0.688	0.733	–
	test	0.655 ± 0.119	0.622 ± 0.063	**0.664** ± 0.067	0.647 ± 0.17
Specificity	train	0.436	0.728	0.709	–
	valid	0.464	0.636	0.651	–
	test	0.4 ± 0.125	**0.637** ± 0.072	0.521 ± 0.071	0.519 ± 0.182
AUC	test	0.538 ± 0.02	**0.695** ± 0.015	0.648 ± 0.02	0.627 ± 0.036

learns to aggregate feature information from each node's nearby neighbors, MinCutPool directly identifies similar nodes that are strongly connected and aggregate them into one cluster. This approach ensures that important information is preserved during the pooling process. Thus, by employing the graph Laplacian to build the graph structure and using MinCutPool to replace the Adaptive Pooling layer in MTPool, we get a better performance than the MTPool model.

To better adapt to small data sets in our experiment, the proposed CNN and LSTM are lightweight models with fewer layers to extract adequate input features. As a result, both models showed an overall superior performance in comparison to the average performance of the baseline models.

Furthermore, the results of our proposed GAF-CNN model demonstrate superior performance compared to the baseline GAF-Attention model in terms of accuracy and sensitivity. Specifically, GAF-CNN achieved an accuracy of 0.639 and the highest sensitivity of 0.777 among all models, while the GAF-Attention model achieved an accuracy of 0.631 and a sensitivity of 0.622. However, the GAF-Attention model has the highest specificity of 0.637 among all models, and its AUC, i.e., 0.695, is slightly higher than that of GAF-CNN, i.e., 0.692. These results suggest that both models have their unique strengths and limitations and may be more suitable for different applications depending on the specific requirements of the task. Our task focuses on TB diagnosis, where accurately identifying positive cases is of utmost importance due to the potentially severe consequences associated with missing a positive case. In this context, sensitivity plays a vital role as it measures the true positive rate and reflects the ability of each model to effectively detect positive cases. Consequently, when evaluating the models based on sensitivity, the GAF-CNN model performs best compared to all other models considered.

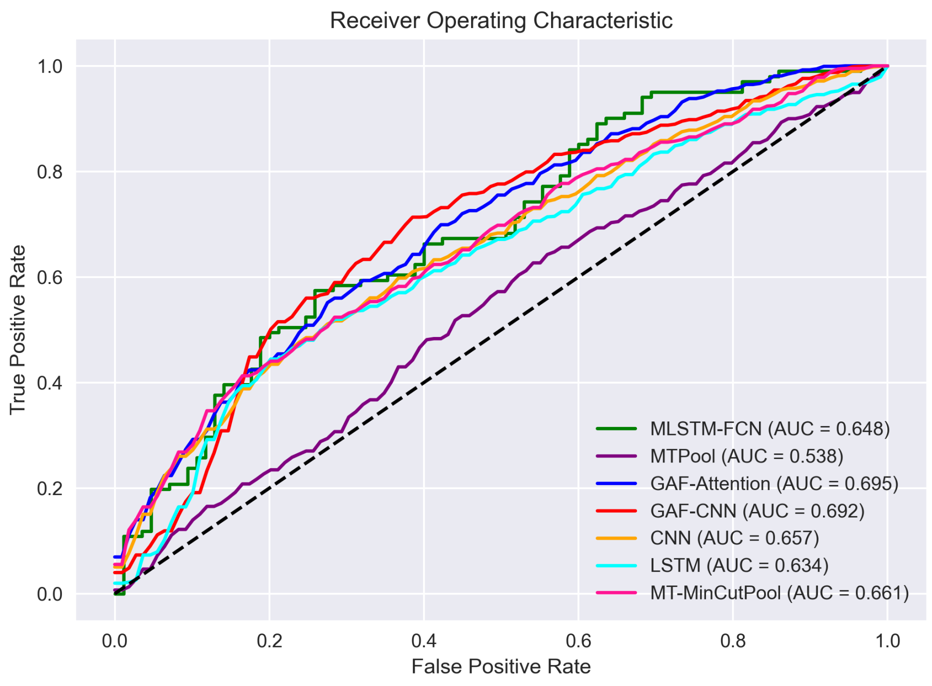


Figure 4.1: The Receiver Operating Characteristic (ROC) curve of each model.

The effectiveness of using GAF image conversion and the GAF-CNN makes us believe that these methods can be extended to a wider range of contexts and similar applications beyond our immediate Tuberculosis MTSC task. Specifically, we suggest further investigation in the following potential areas:

1. Medical diagnosis: GAF image conversion could further assist in identifying a range of medical conditions, from electrocardiogram (ECG) rhythms to signals of Alzheimer's. GAF image conversion in these fields may allow for better clinical decision-making and enhance the accuracy of machine-learning models.

2. Financial time-series analysis: GAF image conversion could be leveraged to predict stock prices and fluctuations in the currency market and further enhance the effectiveness of ML algorithms in predicting trends and changes.

3. Speech recognition: GAF-based image classification could enable more accurate identification of speech patterns, phonemes, and speech denoising. The use of images could be particularly effective in noisy environments where traditional audio inputs may be challenged.

In conclusion, converting time series into images has excellent potential; further experimentation and research in these suggested areas are recommended to explore the full potential of this method.

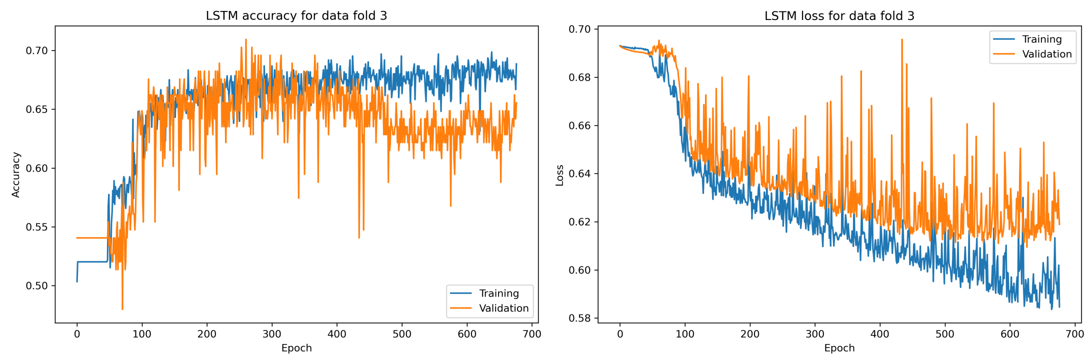


Figure 4.2: The training and validation accuracy and loss curve for the best LSTM model.

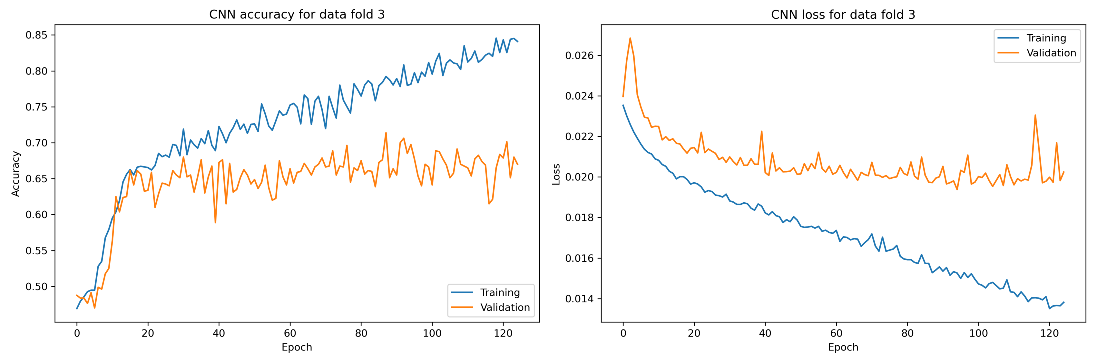


Figure 4.3: The training and validation accuracy and loss curve for the best CNN model.

To better visualize the results, we also compute the confusion matrices for the proposed models (shown in Figure 4.6) and the baseline models (shown in Figure 4.7).

4.4 Conclusions

This chapter presented the main results of our proposed DL models for MTSC Tuberculosis classification and compared them with various state-of-the-art baseline approaches. The experimental evaluation process, including dataset splitting, stratified cross-validation, and multiple runs, ensured the reliability of our results.

Our findings demonstrated that lightweight models are better suited for small-dataset problems and the average performance of our proposed models outperformed that of the baseline methods in all aspects. Specifically, the GAF-CNN model achieved the highest accuracy and specificity, indicating its superior effectiveness for MTSC tasks. Additionally, our proposed MT-MinCutPool model demonstrated its viability for this problem by outperforming the corresponding baseline MTPool model in all evaluation metrics. These results suggest that our proposed approaches can be extended to other domains that face similar challenges.

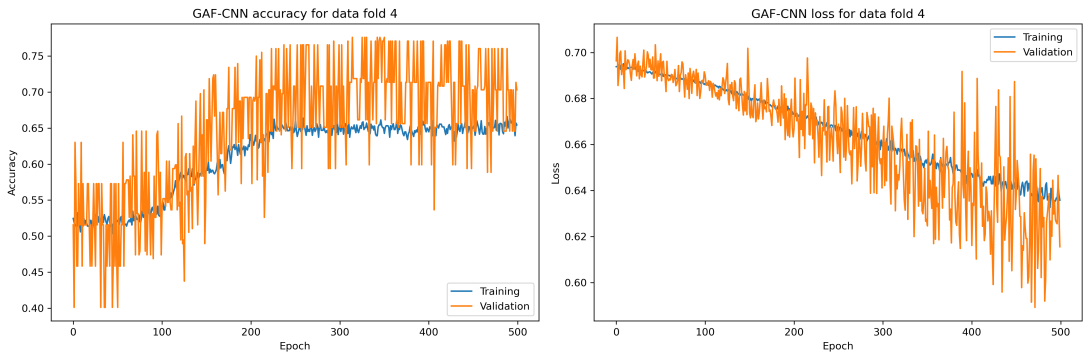


Figure 4.4: The training and validation accuracy and loss curve for the best GAF-CNN model.

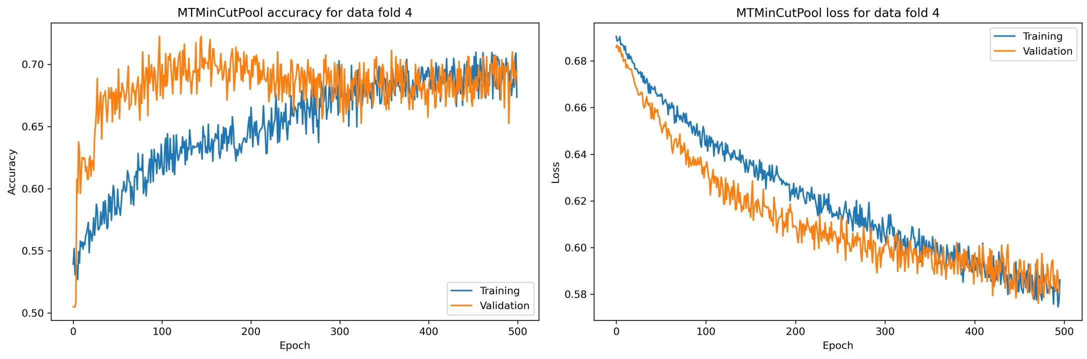
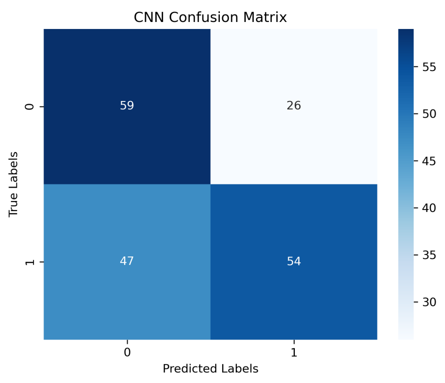
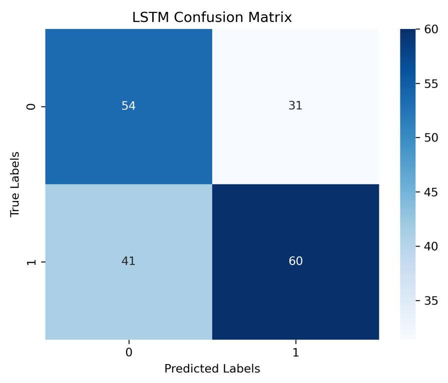


Figure 4.5: The training and validation accuracy and loss curve for the best MTMinCutPool model.

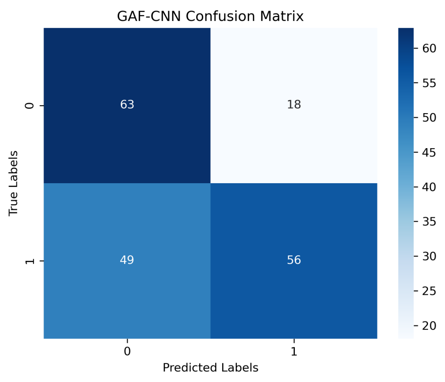
In summary, this chapter presents a logical and formal progression of our research, outlining our methodology, experimental design, and empirical results in a coherent and structured manner.



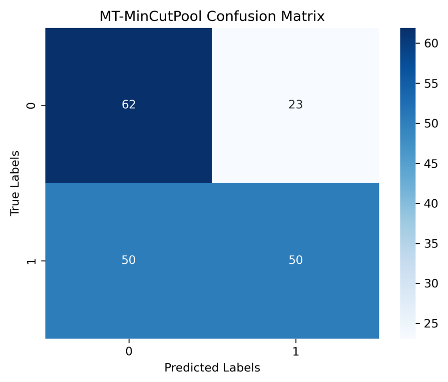
CNN confusion matrix



LSTM confusion matrix

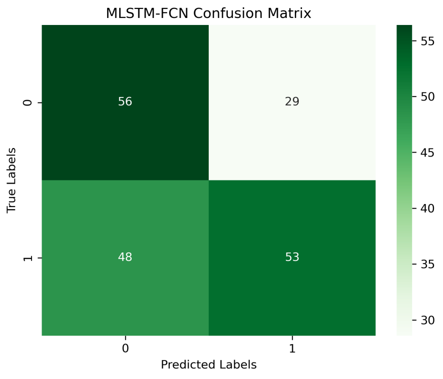


GAF-CNN confusion matrix

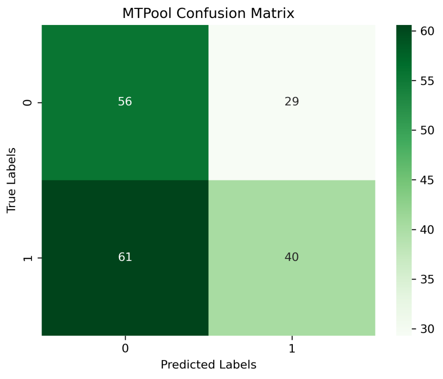


MT-MinCutPool confusion matrix

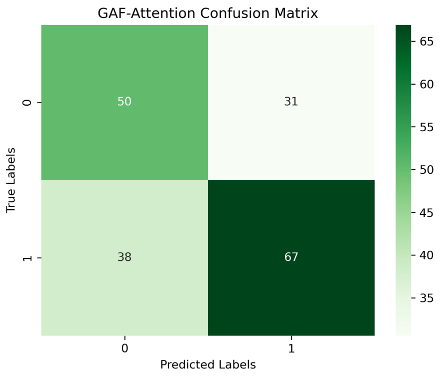
Figure 4.6: The average confusion matrices computed from 10 runs for each of the proposed models: CNN, LSTM, GAF-CNN, and MT-MinCutPool.



MLSTM-FCN confusion matrix



MTPool confusion matrix



GAF-Attention confusion matrix

Figure 4.7: The average confusion matrices computed from 10 runs for each of the baseline models: MLSTM-FCN, MTPool, and GAF-Attention.

Chapter 5

Conclusions

5.1 Summary

This study presented several DL-based approaches, including LSTM, CNN, GAF-CNN, and MT-MinCutPool, to tackle the small TB dataset classification problem with 928 samples containing multivariate time series sensor signals. Firstly, We demonstrated a standard pipeline of some data preprocessing approaches. Subsequently, we evaluated both the proposed DL models and the baseline models on the TB dataset and then compared their results with several state-of-the-art baseline methods.

Our proposed methods exhibit better overall performance compared to the baseline models. In addition, we observed that lightweight models performed better than complex models, highlighting their efficacy for small dataset scenarios. At the same time, the complex models are typically more appropriate for larger dataset scenarios. Specifically, the proposed MT-MinCutPool model outperformed MTPool in all aspects, demonstrating its viability and effectiveness in classifying multivariate time series sensor signals.

The findings of this study have practical implications for TB diagnosis, as the proposed models can assist in the early detection of the disease. Moreover, our approach could be applied to other medical conditions involving small datasets with multivariate time series signals. Overall, the results of this study underscore the potential of DL-based methods for improving medical diagnosis and decision-making.

5.2 Future Work

While our study has demonstrated promising results in classifying the TB dataset using DL approaches, it is essential to acknowledge certain limitations that must be considered in interpreting the findings.

One of the primary limitations of our work is the focus on a single TB dataset. Although our proposed models exhibited an overall better performance compared to several state-of-the-art baseline approaches, the efficacy of the models may vary when applied to different types of datasets or other medical domains.

Furthermore, our study is limited by the small size of the TB dataset with only 928 samples available for analysis. While we have demonstrated the effectiveness of the proposed lightweight models for small datasets, it's worth exploring the proposed models or even more complex models with a larger dataset. And it's also worthwhile exploring other DL models suitable for small dataset classification tasks.

Besides the limitations, this study may also include the following factors that may influence the results:

1. The severity of the target condition (TB) varied among the study population, and this heterogeneity might have introduced variability and increased the uncertainty in the model predictions. Future research could explore the incorporation of severity scores to improve accuracy.

2. The limited data size is a potential factor that may impact the quality of the results, where it's quite challenging to capture the full features associated with TB patterns from the MTS signals with the only available 928 samples.

3. The sensor data used in this study, although multivariate time series signals, may not encompass all the necessary information required to accurately characterize TB disease patterns. The signals measured by the sensors might not fully capture the nuanced variations in TB disease due to the underlying mechanisms and the complexity of the disease process. This limitation in information content could contribute to the challenges faced in achieving higher accuracies.

In light of the limitations mentioned above, several avenues for future research can be explored to build upon our findings and further advance the field of TB classification using DL models.

1. First, future research could extend our work by exploring the efficacy of the proposed models on other types of TB datasets or similar tasks in the field of multivariate time series classification problems. This could help to determine our findings' generalizability and validate the proposed models' effectiveness across different datasets or contexts, especially in medical domains, as this could have significant implications for medical diagnosis and treatment. Such efforts could help develop more robust and versatile machine-learning models that can be applied across different medical contexts.

2. Second, given the limited size of the TB dataset used in our study, it would be beneficial to investigate the performance of the proposed models on larger datasets to determine their scalability and identify any potential trade-offs between model complexity and performance.

3. Third, another avenue worth exploring for small dataset classification tasks is utilizing DL models beyond the traditional approaches. Transfer learning (TL),

in particular, has gained increasing attention due to its ability to construct robust classifiers using a limited amount of data from the target domain. This is achieved through pre-trained models trained on substantial amounts of labeled data in the source domain. Thus, if a more extensive and similar source domain dataset can be found, the potential of transfer learning is worth considering as a solution to the problem of small TB dataset classification tasks. It offers an opportunity to leverage the existing knowledge and structures of the pre-trained models to address the challenges posed by limited target domain data.

4. Fourth, one of the most widely used solutions for the limited dataset problem in DL is using surrogate data. Surrogate data refers to artificially generated data similar to the original dataset. Various techniques can be employed to generate surrogate data, including data augmentation and generative DL models. Data augmentation is a technique that involves transforming the existing data to create new data points that are still representative of the original dataset. This can be achieved through flipping, rotating, jittering, scaling, permutation, cropping, adding noise, or shifting the data slightly [51]. On the other hand, Generative DL models are designed to learn the underlying distribution of the original dataset and generate new data points that are similar to the original data. One such model is the Variational Auto Encoder (VAE), which can generate new data by sampling from the learned distribution of the original dataset. VAEs are particularly useful in small dataset scenarios, where they can cause new data points that are representative of the original data and can be used to improve the performance of DL models.

5. Fifth, to improve the quality of the sensor data and enhance the capture of TB disease patterns, future research could explore additional data sources or complementary modalities. This could involve integrating other types of data, such as clinical measurements or biomarkers, that may provide richer information related to TB disease. Additionally, the acquisition of larger datasets and more comprehensive data collection strategies would be essential to further investigate the relationships between sensor data and TB disease patterns.

6. Finally, as interpretability becomes increasingly essential in the field of DL, future research should explore the interpretability of the proposed models in the context of tuberculosis diagnosis. This could include using visualization techniques or feature importance analysis to provide insights into the model's decision-making process.

Bibliography

[1] Z. K., “Tuberculosis: a global health problem,” *J. Health Popul. Nutr.*, vol. 28, pp. 111–113, 04 2010.

[2] R. Vishinkin, R. Busool, E. Mansour, F. Fish, A. Esmail, P. Kumar, A. Gharaa, J. C. Cancilla, J. S. Torrecilla, G. Skenders, *et al.*, “Profiles of volatile biomarkers detect tuberculosis from skin,” *Advanced Science*, vol. 8, no. 15, p. 2100235, 2021.

[3] M. K. Nakhleh, H. Amal, R. Jeries, Y. Y. Broza, M. Aboud, A. Gharra, H. Ivgi, S. Khatib, S. Badarneh, L. Har-Shai, *et al.*, “Diagnosis and classification of 17 diseases from 1404 subjects via pattern analysis of exhaled molecules,” *ACS Nano*, vol. 11, no. 1, pp. 112–125, 2017.

[4] Y. Y. Broza, R. Vishinkin, O. Barash, M. K. Nakhleh, and H. Haick, “Synergy between nanomaterials and volatile organic compounds for non-invasive medical evaluation,” *Chem. Soc. Rev.*, vol. 47, no. 13, pp. 4781–4859, 2018.

[5] F. Röck, N. Barsan, and U. Weimar, “Electronic nose: current status and future trends,” *Chem. Rev.*, vol. 108, no. 2, pp. 705–725, 2008.

[6] T.-P. Huynh, M. Khatib, R. Srour, M. Plotkin, W. Wu, R. Vishinkin, N. Hayek, H. Jin, O. M. Gazit, and H. Haick, “Composites of polymer and carbon nanostructures for self-healing chemical sensors,” *Adv. Mater. Technol.*, vol. 1, no. 9, p. 1600187, 2016.

[7] L. Brigato and L. Iocchi, “A close look at deep learning with small data,” in *2020 25th International Conference on Pattern Recognition (ICPR)*, 2021.

[8] T. Shaikhina, D. Lowe, S. Daga, D. Briggs, R. Higgins, and N. Khovanova, “Machine learning for predictive modelling based on small data in biomedical engineering,” *IFAC-PapersOnLine*, vol. 48, no. 20, pp. 469–474, 2015.

[9] Y. Hirata, Y. Katori, H. Shimokawa, H. Suzuki, T. A. Blenkinsop, E. J. Lang, and K. Aihara, “Testing a neural coding hypothesis using surrogate data,” *J. Neurosci. Methods*, vol. 172, no. 2, pp. 312–322, 2008.

[10] K. Chopra and S. Singh, “Newer diagnostic tests for tuberculosis, their utility, and their limitations,” *Curr. Med. Res. Pract.*, vol. 10, no. 1, pp. 8–11, 2020.

[11] M. MacGregor-Fairlie, S. Wilkinson, G. S. Besra, and P. Goldberg Oppenheimer, "Tuberculosis diagnostics: overcoming ancient challenges with modern solutions," *Emerging Top. Life Sci.*, vol. 4, no. 4, pp. 435–448, 2020.

[12] C. M. Gill, L. Dolan, L. M. Piggott, and A. M. McLaughlin, "New developments in tuberculosis diagnosis and treatment," *Breathe*, vol. 18, no. 1, 2022.

[13] J. A. Santos, A. Leite, P. Soares, R. Duarte, and C. Nunes, "Delayed diagnosis of active pulmonary tuberculosis-potential risk factors for patient and healthcare delays in portugal," *BMC Public Health*, vol. 21, no. 1, pp. 1–13, 2021.

[14] Y. Y. Broza and H. Haick, "Nanomaterial-based sensors for detection of disease by volatile organic compounds," *Nanomedicine*, vol. 8, no. 5, pp. 785–806, 2013.

[15] G. Konvalina and H. Haick, "Sensors for breath testing: from nanomaterials to comprehensive disease detection," *Acc. Chem. Res.*, vol. 47, no. 1, pp. 66–76, 2014.

[16] U. Tisch, S. Billan, M. Ilouze, M. Phillips, N. Peled, and H. Haick, "Volatile organic compounds in exhaled breath as biomarkers for the early detection and screening of lung cancer," *Int. J. Clin. Rev.*, 2012.

[17] A. P. Turner and N. Magan, "Electronic noses and disease diagnostics," *Nat. Rev. Microbiol.*, vol. 2, no. 2, pp. 161–166, 2004.

[18] H. F. Nweke, Y. W. Teh, M. A. Al-garadi, and U. R. Alo, "Deep learning algorithms for human activity recognition using mobile and wearable sensor networks: State of the art and research challenges," *Expert Syst. Appl.*, vol. 105, pp. 233–261, 2018.

[19] A. Rajkomar, E. Oren, K. Chen, A. M. Dai, and N. Hajaj, "Scalable and accurate deep learning with electronic health records," *NPJ Digit. Med.*, vol. 1, p. 18, 05 2018.

[20] T. L. Nwe, T. H. Dat, and B. Ma, "Convolutional neural network with multi-task learning scheme for acoustic scene classification," in *2017 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pp. 1347–1350, 2017.

[21] D. Xiao, J. Su, and H. Che, "Research on stock price time series prediction based on deep learning and autoregressive integrated moving average," *Sci. Program.*, vol. 2022, p. 4758698, 03 2022.

[22] Z. Wang and T. Oates, "Encoding time series as images for visual inspection and classification using tiled convolutional neural networks," in *Workshops at the twenty-ninth AAAI conference on artificial intelligence*, vol. 1, 01 2015.

[23] R. Pascanu, C. Gulcehre, K. Cho, and Y. Bengio, "How to construct deep recurrent neural networks," *arXiv preprint arXiv:1312.6026.*, 2013.

[24] F. Karim, S. Majumdar, H. Darabi, and S. Chen, "Lstm fully convolutional networks for time series classification," *IEEE Access*, vol. 6, pp. 1662–1669, 2018.

[25] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[26] K. Kawakami, *Supervised sequence labelling with recurrent neural networks.* PhD thesis, Technical University of Munich, 2008.

[27] F. M. Bianchi, D. Grattarola, and C. Alippi, "Spectral clustering with graph neural networks for graph pooling," in *ICML*, pp. 874–883, Nov. 2020.

[28] A. Ng, M. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," *Adv. Neural Inf. Process. Syst.*, vol. 14, 2001.

[29] E. Keogh and C. A. Ratanamahatana, "Exact indexing of dynamic time warping," *Knowl. Inf. Syst.*, vol. 7, pp. 358–386, 2005.

[30] C. A. Ratanamahatana and E. Keogh, "Everything you know about dynamic time warping is wrong," in *Third workshop on mining temporal and sequential data*, vol. 32, Citeseer, 2004.

[31] Y. Zheng, Q. Liu, E. Chen, Y. Ge, and J. L. Zhao, "Exploiting multi-channels deep convolutional neural networks for multivariate time series classification," *Front. Comput. Sci.*, vol. 10, pp. 96–112, 2016.

[32] Y. Y. Broza, X. Zhou, M. Yuan, D. Qu, Y. Zheng, R. Vishinkin, M. Khatib, W. Wu, and H. Haick, "Disease detection with molecular biomarkers: from chemistry of body fluids to nature-inspired chemical sensors," *Chem. Rev.*, vol. 119, no. 22, pp. 11761–11817, 2019.

[33] D. Singh and B. Singh, "Investigating the impact of data normalization on classification performance," *Appl. Soft Comput.*, vol. 97, p. 105524, 2020.

[34] R. Vishinkin, R. Busool, E. Mansour, F. Fish, A. Esmail, P. Kumar, A. Gharaa, J. C. Cancilla, J. S. Torrecilla, G. Skenders, M. Leja, K. Dheda, S. Singh, and H. Haick, "Profiles of volatile biomarkers detect tuberculosis from skin," *Adv. Sci.*, vol. 8, no. 15, p. 2100235, 2021.

[35] R. Gravina, P. Alinia, H. Ghasemzadeh, and G. Fortino, "Multi-sensor fusion in body sensor networks: State-of-the-art and research challenges," *Inf. Fusion*, vol. 35, no. C, pp. 68–80, 2017.

[36] R. Brena, A. Aguileta, L. Trejo, E. Molino Minero Re, and O. Mayora, "Choosing the best sensor fusion method: A machine-learning approach," *Sensors*, vol. 20, p. 2350, 04 2020.

[37] J. Benesty, J. Chen, Y. Huang, and I. Cohen, “Pearson correlation coefficient,” in *Noise reduction in speech processing*, ch. 5, pp. 37–40, Springer, Berlin, Heidelberg, 2009.

[38] T. N. Sainath, O. Vinyals, A. Senior, and H. Sak, “Convolutional, long short-term memory, fully connected deep neural networks,” in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4580–4584, 2015.

[39] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, “Deep learning for time series classification: a review,” *Data Min. Knowl. Discov.*, vol. 33, pp. 917–963, 07 2019.

[40] M. Längkvist, L. Karlsson, and A. Loutfi, “A review of unsupervised feature learning and deep learning for time-series modeling,” *Pattern Recognit. Lett.*, vol. 42, pp. 11–24, 2014.

[41] C. Wei and K. Shi, “Multi-scale attention convolutional neural network for time series classification,” *Neural Netw.*, vol. 136, pp. 126–140, 04 2021.

[42] Y. Sharma, N. Coronato, and D. E. Brown, “Encoding cardiopulmonary exercise testing time series as images for classification using convolutional neural network,” in *2022 44th Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, pp. 1611–1614, 2022.

[43] Z. Duan, H. Xu, Y. Wang, Y. Huang, A. Ren, Z. Xu, Y. Sun, and W. Wang, “Multivariate time-series classification with hierarchical variational graph pooling,” *Neural Netw.*, vol. 154, pp. 481–490, 2022.

[44] G. Zerveas, S. Jayaraman, D. Patel, A. Bhamidipaty, and C. Eickhoff, “A transformer-based framework for multivariate time series representation learning,” in *27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 2114–2124, 2021.

[45] A. P. Ruiz, M. Flynn, J. Large, M. Middlehurst, and A. Bagnall, “The great multivariate time series classification bake off: a review and experimental evaluation of recent algorithmic advances,” *Data Min. Knowl. Discov.*, vol. 35, no. 2, pp. 401–449, 2021.

[46] F. Karim, S. Majumdar, H. Darabi, and S. Harford, “Multivariate lstm-fcns for time series classification,” *Neural Netw.*, vol. 116, pp. 237–245, 2019.

[47] X. Dong, D. Thanou, P. Frossard, and P. Vandergheynst, “Learning laplacian matrix in smooth graph signal representations,” *IEEE Trans. Signal Process.*, vol. 64, no. 23, pp. 6160–6173, 2016.

[48] S. Yan, Y. Xiong, and D. Lin, “Spatial temporal graph convolutional networks for skeleton-based action recognition,” in *AAAI conference on artificial intelligence.*, vol. 32, 2018.

[49] B. Yu, H. Yin, and Z. Zhu, “Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting.,” International Joint Conferences on Artificial Intelligence Organization, 2017.

[50] W. Hamilton, Z. Ying, and J. Leskovec, “Inductive representation learning on large graphs,” *Adv. Neural Inf. Process. Syst.*, vol. 30, 2017.

[51] T. T. Um, F. M. J. Pfister, D. Pichler, S. Endo, M. Lang, S. Hirche, U. Fietzek, and D. Kulić, “Data augmentation of wearable sensor data for parkinson’s disease monitoring using convolutional neural networks,” in *Proceedings of the 19th ACM International Conference on Multimodal Interaction*, (New York, NY, USA), p. 216–220, J. ACM, 2017.

עיבוד וסיווג אותות מערך חיישנים מבוססי ננו-חומר באמצעות למידת מכונה

חיבור על מחקר

לשם מילוי חלקי של הדרישות לקבלת התואר
מגיסטר למדעים בהנדסת חשמל

צ'נשי ליו

הוגש לסנט הטכניון — מכון טכנולוגי לישראל
אייר תשפ"ג　　חיפה　　מאי 2023

עיבוד וסיווג אותות מערך חיישנים מבוססי ננו-חומר באמצעות למידת מכונה

צ'נשי ליו