STRUCTURED DEEP LEARNING: THEORY AND APPLICATIONS

by

FANGQI ZHU

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT ARLINGTON

AUGUST 2020

To my father Jun Zhu and my mother Zhongying Liu, who support and inspire me from the starting point and make me who I am.

ACKNOWLEDGEMENTS

I would like to thank my supervising professor Dr. Qilian Liang for constantly motivating and encouraging me, and also for his invaluable advice during the course of my doctoral studies. His unique "coaching system" helps me to manage the research project in terms of short term goal and stretch goal, so that I am able to get more control about the research progress and quality instead of trapping in some dilemma and lose the directions. Without his training, I would not be able have the chances to explore the broad scope of deep learning and signal processing. Also, I would not be able have the chances to reach out to the research opportunities from the industry. I wish to thank my academic advisors Dr. Jonathan Bredow, Dr. Ioannis Dimitris Schizas, Dr. Yunze Sun, Dr. Yan Wan for their interest in my research and for taking time to serve in my dissertation committee.

I would also like to extend my appreciation to Seagate Technology LLC for providing me the financial support (research internship program) for my doctoral studies during the final year. I want to thank Dr. Darrell Louder and Mr. Allan Luk with the Operations and Technology Advanced Analytics Group (OTAAG) team for hiring me as the data science and machine learning intern. I am especially grateful to Dr. Nicholas Propes, Dr. Addishiwot Woldesenbet, Dr. Zhiqiang Xing, Dr. Chao Feng and Mr. Bruce King for their interest in my research and for the helpful discussions and invaluable comments for structured deep learning and its applications in anomaly detection for industrial manufacturing.

I am grateful to all the instructors, mentors and supervisors who taught me during the years I spent in the universities, first in China and then in Unites States. I would like to thank my supervisor of master degree Dr. Jing Liang, who used to be the outstanding graduate from University of Texas at Arlington, for encouraging and inspiring me to pursue graduate studies in United States. I also would like to express my gratitude to Dr. Ioannis Dimitris Schizas, who his broad, systematic and rigorous guidance and in mathematics and theoretical machine learning.

Finally, I would like to express my deep gratitude to my father Jun Zhu and mother Zhongying Liu who their long supporting over 20 years and always serve as the diligent and perseverance role model for me. I appreciate the loyalty of my family and without the strong family, I would not have a chance to record my footprints for over half of the United States of America.

July 30, 2020

ABSTRACT

STRUCTURED DEEP LEARNING: THEORY AND APPLICATIONS

FANGQI ZHU, Ph.D.

The University of Texas at Arlington, 2020

Supervising Professor: Qilian Liang

The increasing amount of data generation has boosted the broad range of research in big data and artificial intelligence. Besides the success of the deep learning in wide range of research area, it meets its pitfalls on the following three problems:

- *Data hungry*: current models often require to feed GB, TB even PB level of data, which is easily overfit.
- *Hard to generalize*: deep learning constructs representations that memorize their training data rather than generalize to unseen scenarios
- *Missing critical information*: off-the-self deep learning framework may not fully utilize the underlying information of the data

We investigate the above problems to find the underlying structural information in the data and framework of the neural network and try to understand how these two interact with each other. In this study, two typical structured learning models are mainly investigated: time series and graph data.

The long-existing vanishing and gradient problems cause the bottleneck of convergence problem, which impede the process of keeping memory in sequential models. We formulate this problem by considering the weights variation in the recurrent

weight matrix and utilize the constraints of the Stiefel manifold to design decomposition methods. Two orthogonal constrained recurrent neural network (OCRNN) and their corresponding training algorithms are proposed and demonstrated that they outperform previous structures on the synthetic memory keeping dataset.

In order to verify the performance of the OCRNN on real problems, we investigate the problem of throat polyp detection problem based on the acoustic sampling data. By preprocessing the acoustic data using standard time-frequency expansion, the processed features are sent to the OCRNNs. The OCRNN can save 1/10 of the total number of parameters compared to the standard RNN while achieving competitive performance, which demonstrate its computational efficiency.

The creation of the mixed model with the one-dimesion convolutional neural network (1dCNN) and OCRNN is deployed on the sleep stage scoring problem based on the EEG signals. The 1dCNN is leveraged to extract fine and coarse time-frequency feature map for the downstream OCRNN. We demonstrate that the 1dCNN-OCRNN can save both the time complexity and spatial complexity and reach a comparatively better performance compare to previous benchmark results.

The graph neural network based model for image anomaly detection is introduced in the final part. The superpixels serve as the mapping from pixel-wise image to graph feature and topology information. The hierarchical variational graph autoencoder with pooling and upsampling operations is adopted for semi-supervised anomaly detection fashion. The feasibility of this model is demonstrated on road surface anomaly detection, which demonstrates its competitive ability and savings of the storage of the neural network overload.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

CHAPTER 1

Introduction

1.1 Motivation of the Structural Deep Learning

With the emerging with high-performance computing platform (GPU, TPU) as well as the ability to deploy the large-scale neural network, deep learning represents the revival of the artificial intelligence. Deep learning demonstrates its power through its application in areas such as social network, financial, engineering, industrial manufacturing, chemical and biomedical study, physics and astronomy. Besides its great success in delivering good results for certain tasks, we still have not a good understanding of how it works under the hood and treat it as the black-box to plug and play for most of the time. We have also witnessed the situation that deep learning model cannot reach the expected performance due to the data quality problem, improper choice of the model and inferior design of the inter-connected module inside the deep neural networks. In order to find the interaction among the data and framework, the structured deep learning is proposed to shed light on this problem.

There are different ways to define the "structured deep learning" and the definition for the thesis is focusing on the structure of the neural network and its interaction of data, regardless the detailed data format such as texts, tables, image formats. For example, deep neural network induced by the inherent sparsity of data [1], model compression by adoption of pruning approach [2] and random regularization technique [**?**] can be leveraged to reduce the increasing high demand for computing resources. Beyond that, the research work in this thesis focuses on the two typical structures of the data, the time-series and graphs, The aim is to design the correlated structured deep

learning frameworks to better under mining the inherent information in the data as well as design the low-cost high efficiency neural network to reach a broad deployment of the deep learning model on applications with low computation load such as edge computing and internet of things.

1.2 Essentials of Deep Learning

1.2.1 Activation Function and Multi-Layer Perceptron Model

In this section, some fundamental building blocks of the structured deep learning used in this study will be introduced. The deep neural network starts from imitating the neuron structure in the human brain and try to create the mathematical model to describe the chemical signal's propagation for the interconnections within the complex brain network. To begin with, a single artificial neuron or perceptron, which can be represented as:

$$\mathbf{y} = \sigma(\mathbf{x}^T\mathbf{w} + \mathbf{b}), \tag{1.1}$$

where $\mathbf{x}$ is the input to the artificial neuron, $\mathbf{w}$ is the training weights, $\mathbf{b}$ is the bias term. A nonlinear function σ activates the neuron to accomplish certain functions.

The choice of the nonlinear function is question-dependant. For example, to model the Bernoulli distribution, the sigmoid function is adopted with the variation between [0, 1], which is suited for binary classification problem,

$$\sigma(x) = \frac{1}{1 + \exp(-x)}. \tag{1.2}$$

Rectified linear unit (ReLU) function, which is the most widely used activation functions in deployment of deep learning, with the range between 0 and 1,

$$\sigma(x) = \max(0, x). \tag{1.3}$$

We will adopt a modified ReLU function in the initialization of the structured recurrent neural network in the following chapters.

We can extend the formula 1.1 to multiple neurons, which form the basic building blocks of the deep learning – multilayer perceptron (MLP). Multilayer perceptron is a deep feedforward network which the information flow directly from the beginning to the end with no feedback loops (the deep neural network which contains feedback loops is the recurrent neural network and will be covered in the next section). It defines a mapping $\mathbf{y} = f(\mathbf{x}; \boldsymbol{\theta})$ such that it tries to learn the parameter vector $\boldsymbol{\theta}$ with the best approximation. The MLP can be representated as follows:

$$\mathbf{y} = \sigma_n(\mathbf{W}_n\sigma_{n-1}(\mathbf{W}_{n-1}\sigma_{n-2}\sigma_1(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1) + \cdots + \mathbf{b}_n), \tag{1.4}$$

where $\mathbf{W}_n, n = 1, \ldots, N$ and $\mathbf{b}_n, n = 1, \ldots, N$ are the weight matrices and bias vectors respectively. $\sigma_n, n = 1, \ldots, N$ are the activation functions. The intermediate activation functions are most likely to be the same, but the activation function of the latest layer will be different sometimes depend on the machine learning tasks and the way to define to the loss functions. The MLP will be regarded as the universal function approximator due to the theoretically it can approximate nonlinear function of any order with given constraints.

1.2.2 Stochastic gradient descent and optimization

Large training sets are generally good for the generalization of the deep learning algorithms. It will avoid the overfitting problems and cover the sample space as much as possible. But large training sets are also more computationally expensive, especially when training algorithms is imposed on the whole datasets directly. Con-

sider the set of data pair $(\mathbf{x}_i, y_i), i = 1, \ldots, N$ and the corresponding differentiable loss function is given as:

$$J(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{x},y\sim\hat{p}_{data}} L(\mathbf{x}, y, \boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m} L(\mathbf{x}_i, y_i, \boldsymbol{\theta}), \tag{1.5}$$

where m is the number of the samples in the current minibatch. These minibatches are drawn uniformly in the whole training datasets, using techiniques like random shuffle. The expectation term is designed to be approximately estimated using a small set of samples. The estimate of the gradient is:

$$\mathbf{g} = \frac{1}{m} \bigtriangledown_{\boldsymbol{\theta}} \sum_{i=1}^{m} L(\mathbf{x}_i, y_i, \boldsymbol{\theta}). \tag{1.6}$$

The stochastic gradient descent algorithm is to minimize the cost along the direction of the gradient,

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \alpha\mathbf{g} \tag{1.7}$$

where α is the learning rate. The optimization process may not be lead to an expected local minimum at the expected amount of time, but it mostly likely will find a comparativey low value of the cost function quickly enough.

When applying stochastic gradient descent to non-convex loss functions of the deep neural networks, the convergence guarantee may not hold due to the difficulty to find the global minimum, and is sensitive to the values of the initial parameters. It is expected to have a good optimization algorithm to tackle this situation. The idea of introducing the momentum into the gradient descent help with the convergence. Particularly, the Nesterov momentum reduces the convergence error from $\mathcal{O}(1/k)$ to $\mathcal{O}(1/k^2)$. The updpated rule is given by:

$$\boldsymbol{v} \leftarrow \alpha\boldsymbol{v} - \epsilon\nabla_{\boldsymbol{\theta}} \left[\frac{1}{m}\sum_{i=1}^{m} L\left(\boldsymbol{f}\left(\boldsymbol{x}^{(i)}; \boldsymbol{\theta} + \alpha\boldsymbol{v}\right), \boldsymbol{y}^{(i)}\right)\right] \tag{1.8}$$

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \boldsymbol{v} \tag{1.9}$$

The Adam and Rmsprop optimization are the two algorithms adopted mostly in this manuscript, driven by the momentum mechanism [3].

1.3 Background of structured recurrent neural network

Time-series related problems such as prediction, classification, correlation analysis have long been investigated for decades. Traditional methods like statistical analysis generalized linear model, causality analysis mainly focus on the hand-crafted features, which are able to quickly deliver results. But the precision and robustness of these models become the bottleneck and these methods cannot behave well on long memory, non-stationary time-series data. Recurrent neural network (RNN) is an extensible large-scale structure that does not require hand-crafted process and can be designed to automatically deploy for different time-series task. RNN mimics the process to storage the memory and output representation of the input data for specific deep learning tasks. It takes sequence with the same or variable length $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_\tau$, utilize the property of parameter sharing during the propagation of the weights, to generate outputs with adjustable length. The tasks like time series analysis, sequential modeling, time-series prediction and natural language processing drives the development of more powerful learning frameworks and algorithms. With the emerging of the deep learning, more and more attentions are attracted by the RNN for its power of solving temporal dynamic problems of varying complexity.

A typical sequence-to-vector framework of the RNN is demonstrated in Fig. 1.1, which will be adopted for the classification tasks in Chapter 3 and 4. The hidden state h^t can be regarded as a dynamic parametric system which takes the output of the last state and current input. Then it propagates the current state to the next state and fuses the new input to generate the next hidden state. We can express the progress in the following:

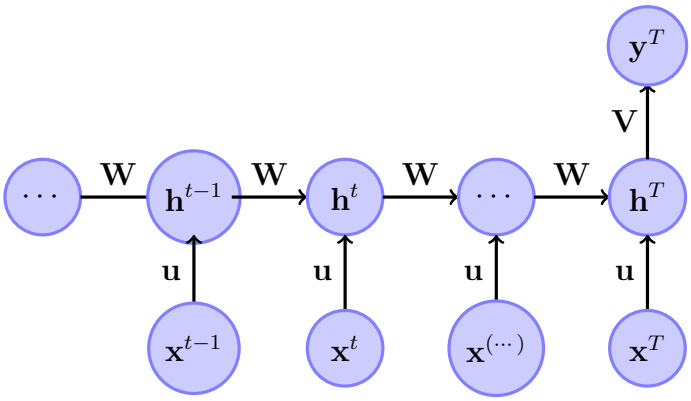


Figure 1.1. Schematic diagram of RNN structure.

$$\mathbf{h}^{(t)} = f(\mathbf{h}^{(t-1)}, \mathbf{x}^t; \theta), \quad \text{for } t = 1, \ldots, \tau, , \tag{1.10}$$

where θ is the parameter of the current state of the RNN. For vanilla RNN, it will take the form as $\mathbf{h}^{(t)} = \phi(\mathbf{W}_x^T \mathbf{x}_t + \mathbf{W}_y^T \mathbf{y}_{t-1} + \mathbf{b})$, where b is the bias term (optional) and ϕ is some chosen nonlinear activation function such as ReLU or hyperbolic tangent. The final output $\mathbf{y}^T$ is:

$$\mathbf{y}^T = \phi(\mathbf{W}_x^T \mathbf{x}_T + \mathbf{W}_y^T \mathbf{y}_{T-1} + \mathbf{b}). \tag{1.11}$$

The hidden state is also be regarded as the memory cell, which will cache the information for back-propagation through time in the training phase. However, due to the long memory requirement in general RNN tasks, the problem of vanishing and gradient exploding problem appeals. In order to get insights of what is inside the "black box" of the RNN, people start to focus on the nature of the back propagation of the RNN recently and find out that the problem lies on the restriction of the spectral norm of the recurrent matrix in order to solve the gradient explosion and vanishing problem [4]. Essentially, one of the key ideas is to find good ways of parameterizing the recurrent neural networks and improve the conditioning of the cost gradient [5].

Generally there are three types of research directions to the SRNN. The first category is related to the the classical gate-control mechanism based RNNs, such as the long-short term memory network (LSTM), gated recurrent unit network (GRU), peephole-LSTM and attention mechanism based model [6, 7]. For gate-control mechanism based RNNs, several modules are designed to keep, drop and distill the information by backpropagating the RNN with hyper-parameter tuning. It is the mostly adopted first-try model for time-series related tasks. The second category is the transformer network [8, 9]. The transformer network is specialized for the natural language processing scope that utilized very deep customized bi-directional encoder network to achieve tasks like machine translation. Although it is very powerful and achieve the state-of-the-art performance on several benchmark tests, it requires a huge computation load and currently there is no low-cost distributed deployment solution to the best of the knowledge. Therefore, transformer network is beyond the scope of comparison of this study. The third category is the orthogonal constrained based RNN (OCRNN) [10, 11]. Different from the previous two mechanisms, the OCRNN address the vanishing gradients and exploding problem by directly working on the recurrent weight matrix in the network. The orthogonal constraints and the correspond decomposition methods directly control the interaction between the data and the framework to keep a longer memory and make the RNN robust against vanishing gradients and exploding problem. It does not necessarily resort to high-performance platform and can be flexibly deployed on tasks with different orders of complexity. This direction provides a more "structured" flavor compared with the rest two and is selected as the research direction in this study.

1.4 Background of Structured Graph Neural Network

We have witnessed the success of deep-learning models for regular grid data structure like speech (1d grid time-series), image (2d grid, not include channel dimension, w.l.o.g), and video, in which there are an underlying Euclidean structure. Recently there has been a growing interest in trying to extend the framework on structured data, particularly the non-Euclidean data. There are some applications that belongs to this domain: social network with different types of connections and heterogeneous message pass on the network; sensor network which fuses data among different types of connections, chemical molecular with different atoms and compounds demonstrate different properties and structures, to name a few. These applications have their own objects and related relationships in varied domains, which suited for using graph to model them.

Traditional graph-based machine learning model will mainly treat the graph-level as the preprocessing steps and combine it with the standard machine learning framework. In order to address the similarity among the node pairs or graph-pairs, the kernel method is introduced to form the graph kernel to solve the similarity related problems [12]. It can also be formulated as the feature engineering steps for downstream tasks such as the graph clustering [13]. To classify the graphs for some supervised task, the graph kernel is able to combine with the SVM classifier [14]. In order to address the information of signal flow inside the graph network, the random walk model is leveraged to characterize the message passing process [15].

The above methods heavily rely only hand-crafted domain knowledge and the inherent irregular data structure is approached with decorate algorithms. It is able to extend those design to large-scale setting such as knowledge graph, natural language processing, drug discovery, image based anomaly detection, and visual understanding. Generally, we may categorize the types of the problems as follows:

- *Node classification*: classify each of the nodes into one of C classes, i.e., to produce a node class probability matrix $\mathbf{Y} \in \mathbb{R}^{N \times C}$ such that $\mathbf{Y}_{ij} = \mathbb{P}(\text{Node}\, i \in \text{Class}\, j)$
- *Graph classification*: Given a set of graph $\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_n$, classify these graphs as one of the category C classes.
- *Link prediction*: Given a set of graph $\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_n$, for the upcoming new graphs, predict whether two nodes in a network are likely to have a link.
- *Community Detection*: Given a graph $\mathcal{G}$, nodes of the graph network can be easily grouped into (potentially overlapping) sets of nodes such that each set of nodes is densely connected internally.
- and more ...

Graph neural networks (GNNs) are deep learning based methods that operate on graph domain. Due to its convincing performance and high interpretability, GNN has been a widely applied graph analysis method recently. In the following paragraphs, we will illustrate the fundamental motivations of graph neural networks. Graph neural networks have been explored in a wide range of problem domains across supervised, semi-supervised, unsupervised and reinforcement learning settings. In this section, we simply divide the applications in three scenarios:

- Structural scenarios where the data has explicit relational structure, such as physical systems, molecular structures and knowledge graphs;
- Non-structural scenarios where the relational structure is not explicit include image, text, etc;
- Other application scenarios such as generative models and combinatorial optimization problems

it is important to incorporate domain knowledge into the model when constructing a graph or choosing architectures. For example, building a graph based on the relative

distance may be suitable for traffic forecasting problems, but may not work well for a weather prediction problem where the geographical location is also important. Second, a graph-based model can usually be built on top of other architectures rather than as a stand-alone model. For example, the computer vision community usually adopts CNNs for detecting objects and then uses graph-based deep learning as a reasoning module. For NLP problems, GCNs can be adopted as syntactic constraints. As a result, key challenge is how to integrate different models. These applications also show that graph-based deep learning not only enables mining the rich value underlying the existing graph data but also helps to naturally model relational data as graphs, greatly widening the applicability of graph-based deep learning models.

In the following study, a hierarchical graph generative model with superpixel based features is proposed to solve the image based anomaly detection problem. Some mathematical preliminaries are provided in the following subsection for the self-contained purpose.

1.4.1 Preliminaries of the GCN

The undirected graph, which means the connection between the nodes (vertices) do not contain directions, is the only case considered in this work. An undirected graph G is a pair of sets $G(\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is the set of vertices and $E \in \mathcal{V} \times \mathcal{V}$ is the set of edges of the network. $\mathbf{A}$ is the adjacency matrix that describes the connection between the nodes, such that $\mathbf{A}_{i,j} = 1$ if node i connects to node j and 0 otherwise. For undirected graph, $\mathbf{A}_{i,j} = \mathbf{A}_{j,j}$, thus $\mathbf{A}$ is the symmetric matrix. The degree matrix $\mathbf{D}$ is the summation of the connection of the neighborhood, such that $D_{ii} = \sum_j \mathbf{A}_{i,j}$. The Laplacian can be represented as $\mathbf{L} = \mathbf{D} - \mathbf{A}$.

$$\mathbf{L} \longrightarrow \mathbf{L}_N = \mathbf{I} + \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2} \longrightarrow \tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2} \tag{1.12}$$

where $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$, $\tilde{\mathbf{D}}_{ii} = \sum_j \tilde{\mathbf{A}}_{ij}$, $\mathbf{L}_N$ is the normalized Laplacian matrix.

For instance, the following graph in Fig 1.2 is a graph with 5 nodes and 5 edges. And we have,

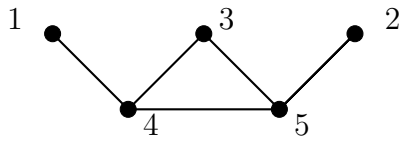


Figure 1.2. The simple graph with 5 nodes and 5 edges.

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \longrightarrow \mathbf{L} = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 \\ -1 & 3 & -1 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & -1 & -1 & 3 & -1 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix}. \tag{1.13}$$

1.4.2 Graph Neural Network

Since Laplacian matrix $\mathbf{L}$ is a symmetric, we can impose the eigenvalue decomposition on $\mathbf{L}$,

$$\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T, \tag{1.14}$$

where $\mathbf{U} = [\mathbf{u}_0, \ldots, \mathbf{u}_N] \in \mathbb{R}^{N \times N}$ is the matrix of the eigenvectors, which is also called the graph Fourier basis. $\mathbf{\Lambda}$ is the diagonal matrix of the eigenvalues where $\mathbf{\Lambda} = \text{diag}(\lambda_0, \ldots, \lambda_N) \in \mathbb{R}^{N \times N}$. These relations are similar to the filters in signal processing so that we can extend the similar ideas to graph signal processing. The graph Fourier transform of a input signal $\mathbf{x} \in \mathbb{R}^n$ is $\hat{\mathbf{x}} = \mathbf{U}^T\mathbf{x}$ and the graph inverse Fourier transform can be represented as $\mathbf{x} = \mathbf{U}\hat{\mathbf{x}}$.

The spectral graph filtering operation is kind of convolution-like operation on graph. Denoted by $*_G$, the graph convolution can be represented as:

$$\mathbf{x} *_G \mathbf{y} = \mathbf{U}\left((\mathbf{U}^T\mathbf{x}) \odot (\mathbf{U}^T\mathbf{y})\right). \tag{1.15}$$

A signal x which is filtered by g will get the output y as [16]:

$$y = g_\theta(L)x = g_\theta(\mathbf{U}\mathbf{\Lambda}\mathbf{U}^T)(x) = \mathbf{U}g_\theta\mathbf{\Lambda}\mathbf{U}^T x. \tag{1.16}$$

The computation of the matrix multiplication of the eigendecomposition will take up to $\mathcal{O}^3$ complexity. In order to solve this problem, the polynomial approximator $g_\theta(\mathbf{\Lambda}) = \sum_{k=0}^{K-1}\theta_k\mathbf{\Lambda}^k$ is considered. Specifically, the Chebyshev polynomial is adopted due to the recursive relation and it forms an orthogonal basis for $L^2([-1,1], dy/\sqrt{1-y^2})$,

$$g_\theta(\Lambda) = \sum_{k=0}^{K-1}\theta_k T_k(\tilde{\Lambda}), \tag{1.17}$$

where $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$ with $T_0(x) = 1$ and $T_1(x) = x$. It can be further simplified to the first order case such that,

$$g_{\theta'} \star x \approx \theta'_0 x + \theta'_1\left(L - I_N\right)x = \theta'_0 x - \theta'_1\mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}x. \tag{1.18}$$

We have the following relations:

$$\begin{aligned} g_\theta \star x &= \mathbf{I} + \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}} \\ \mathbf{I} + \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}} &\longrightarrow = \tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{A}\tilde{\mathbf{D}}^{-\frac{1}{2}} \\ \mathbf{Z} &= \tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{A}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{X}\mathbf{\Theta} \end{aligned}$$

Based on the first order approximation above, we may have the forward propagation of the GNN model as [17]:

$$\mathbf{Z} = f(\mathbf{X}, \mathbf{A}) = \text{softmax}(\hat{\mathbf{A}}\text{ReLU}(\hat{\mathbf{A}}\mathbf{X}\mathbf{W}^{(0)}))\mathbf{W}^{(1)}) \tag{1.19}$$

with the related cross entropy loss function (use the classification as the example, can be changed for different tasks) ,

$$\mathcal{L} = -\sum_{l\in\mathcal{Y}_L}\sum_{f=1}^{F} Y_{lf}\ln Z_{lf} \tag{1.20}$$

1.5 Overview of Proposal

Chapter 2 presents the projection-like based and polar-decomposed manifold retraction methods and designs the corresponding orthographic retraction RNN (or-RNN) and polar based RNN (pdRNN). A new coordinate-descent like iteration algorithm is proposed for updating the weight coefficients of the recurrent matrix and it outperforms previous baseline RNN and cayley-transform based RNN on standard copying and adding memory synthetic tests. Furthermore, a pseudo-spectrum based visualization methods is introduced to visualize of the weights evolving and address the relation between orthogonality and robustness of the RNN.

Chapter 3 deploy the orthogonal constrained RNN (OCRNN) on the acoustic signal based polyp detection problem. Running vowels are recorded using acoustic sampling apparatus and labeled as either normal or abnormal and it is formulated as the binary classification problem. A time-frequency expansion technique is served as the preprocessing for the downstream of the RNN processing. It is demonstrated that the OCRNN can achieve comparatively results in terms of accuracy and loss with much smaller total number of parameters, which saves the computation cost.

Chapter 4 investigates the problem of combining the one-dimensional convolutional neural network (1dCNN) and OCRNN for sleep staging scoring based on the signal channel EEG signals. The raw EEG signal is first preprocessed by fine and coarse filtered 1dCNN to get the two-level time-frequency feature map. Then the two-level feature vectors are concatenated and sent to the OCRNN to get the

scoring. It is shown that the 1dCNN-OCRNN can achieve comparatively better results in accuracy categorical F1-score. Moreover, the time complexity for training (1/3 of training epochs vs. previous benchmark) and spatial complexity (1/6 of total number of network parameters vs previous benchmark) manifests the potential of the proposed model.

Chapter 5 proposes an end-to-end superpixel based graph anomaly detection framework for anomaly detection in pixelwise image anomaly detection problems. There is no available way to directly transform the image to the graph data that fits for graph neural network. The SLIC algorithm is leveraged to extract the superpixels to summarize and compress the information within the specified region. These regions are further transformed to graph with feature matrix and adjacency matrix by a conditional random field approach. The variational graph autoencoder with pooling and upsampling layer is utilized training on the normal samples to learn the characteristics or the normal cases and inference is carried out on the test set to make decisions based on distance based metrics for anomaly detection. Experiment results demonstrate that comparative performance can be achieved with 60 times saving in terms of network complexity.

Chapter 6 finalizes with the conclusion. The main achievements of this dissertation are highlighted and future research directions are listed.

CHAPTER 2

Structured Recurrent Neural Network with Orthogonal Constraints: Theory

2.1 Motivations

One of the difficulties that optimization algorithms of the deep neural network must overcome arises when the computational graph becomes extremely deep. In recurrent neural network (RNN), by repeatedly applying the same recurrent operation unit at each time step, the very deep computational graphs is formulated as a long temporal sequence. These compositions can result in extremely nonlinear behavior and complex dynamics, which leads to accumulation of error and easily leads to vanishing gradients and exploding.

The problem of vanishing and exploding in the deep learning, especially for sequential modeling related tasks, has long played as the pivotal bottleneck during the process of back propagation through time (BPTT) on the RNN and it is still an open topic [3]. Some intuitive thoughts are that some of the information may be redundant. Therefore, the forget and update mechanism are adopted to form the gated control recurrent neural network such as long short term memory (LSTM) network [6, 18] and gated recurrent unit (GRU) network [7]. Both of them have been widely used in both academic and industrial level end-to-end learning framework. But they still have limitations and they have not responded to the nature of the bottleneck directly.

The function composition employed by RNNs somewhat resembles matrix multiplication. Consider the special case where recurrence relation as a very simple RNN lacking a nonlinear activation function, and lacking inputs $\mathbf{x}$. The recurrence relation

can be represented as $\mathbf{h}^{(t)} = \mathbf{W}\mathbf{h}^{(t-1)}$, which removes the current input $\mathbf{x}$ and assumes that we consider the ReLU function with inputs larger than or equal to 0. Taking the whole connected recurrent units into consideration, we have:

$$\mathbf{h}^{(t)} = (\mathbf{W}^t)^{\mathrm{T}}\mathbf{h}^{(0)} \tag{2.1}$$

If we further assumes that recurrent matrix can take the eigen-decomposition (generally will be matrix that is diagonalizable or nondefective) $\mathbf{W} = \mathbf{Q}\mathbf{\Sigma}\mathbf{Q}^{\mathrm{T}}$ with orthogonal $\mathbf{Q}$, formula 2.1 can be further simplified as:

$$\mathbf{h}^{(t)} = (\mathbf{Q}^t)^{\mathrm{T}}\mathbf{\Sigma}^t\mathbf{Q}\mathbf{h}^{(0)} \tag{2.2}$$

Repeated multiplication of the shared parameters across the whole cascades of RNN leads to difficulties, especially through the process of back-propagation throught time (BPTT). Typically, the vanishing and exploding gradient problem refers to the fact that gradients through such a graph are also scaled according to $\mathrm{diag}(\lambda^t)$.Any eigenvalues λ_i that are not near an absolute value of 1 will either explode if they are greater than 1 in magnitude or vanish if they are less than 1 in magnitude. Therefore, any element of $\mathbf{h}^{(0)}$ that is not aligned with the largest eigenvector will eventually be discarded.

Let $\mathcal{E}$ be some differentiable loss function of current state $\mathbf{h}_t$ to measure the performance metrics of the RNN on some given task (e.g. classifiction or prediction), according to the chain rule, we have:

$$\frac{\partial \mathcal{E}}{\partial \mathbf{h}_t} = \frac{\partial \mathcal{E}}{\partial \mathbf{h}_T}\frac{\partial \mathbf{h}_T}{\partial \mathbf{h}_t} = \frac{\partial \mathcal{E}}{\partial \mathbf{h}_T}\prod_{k=t}^{T-1}\frac{\partial \mathbf{h}_{k+1}}{\partial \mathbf{h}_k} = \frac{\partial \mathcal{E}}{\partial \mathbf{h}_T}\prod_{k=t}^{T-1}\mathbf{D}_{k+1}\mathbf{W}_k^T \tag{2.3}$$

where $\mathbf{D}_{k+1} = \mathrm{diag}\left(\sigma'\left(z_{k+1}\right)\right)$ is the Jacobian matrix of the pointwise nonlinearity. When considering the setting of RNN without complex gated-control like LSTM or GRU, by leveraging the viewpoint of dynamic systems, the above problem is formulated as the stability of the recurrent weight matrix among the hidden layer. The

Jacobian matrix $\frac{\partial \mathbf{h}_{k+1}}{\partial \mathbf{h}_k}$ of the consecutive states is expressed as the $\mathbf{W}^T \text{diag}(\sigma'(\mathbf{x}_k))$ and the stability lies on the spectral norm of recurrent weight matrix (singular value) of $\mathbf{W}$, where $\mathbf{W}$ is the weight matrix between consecutive hidden layer [4].

As the early study pointed out [19], it is expected that the we can bypass the problem by letting RNN staying in a region of parameter space where the gradients do not vanish or explode. Ideally, this kind of RNN can store the cascaded memories in a way that is robust to small perturbations. Whenever the model is able to represent long term dependencies, the gradient of a long term interaction has exponentially smaller magnitude than the gradient of a short term interaction, which will take a very long time to learn long-term dependencies, because the signal about these dependencies will tend to be hidden by the smallest fluctuations arising from short-term dependencies [20].

The above problem has been addressed in a new formulation in recent years. Starting from the seminal work [10], by decoupling the relation of the weight matrix, hidden states and input signal, the decomposition of the recurrent weight matrix such that it satisfies the region of $\mathbf{W}^T\mathbf{W} = \mathbf{I}$. It leverages the strict orthogonal and unitary constraints on the recurrent weight matrix, and the factorization of the recurrent weight matrix as the multiplication of seven basic unitary basis matrices. The capacity limit of this formulation is pointed out in [21] and a full capacity model is proposed based on optimization on the Stiefel manifold. Some other related formulations have been proposed such as leveraging Householder transform [22] and unitary matrix decomposition [23]. However, the problem of whether it is necessary to add strict unitary constraints on the recurrent weight matrix is questioned by [24] and the kronecker recurrent model is proposed with soft unitary constraints. Later this phenomenon has been verified on the complex evolution recurrent neural network[25].

They drop out the restrictions of unitary constraints on the diagonal matrices compared to [10] and employ on the copying problem baseline task and an industrial level automatic signal recognition database to verify that strict restrictions will degrade the performance in some cases.

Some more concise and flexible formulation based on Cayley transform has been proposed there in [26, 27]. The iteration of the gradient based on properties of skew-symmetric matrices has simplified the propagation process so that computation cost has reduced. One of the problems of this formulation is that the scaling matrix $\mathbf{D}$ is required to be determined for different tasks and the number of negative ones and the position of the 1s' and -1s' are hyper-parameters for tuning. Later on, the modified version for the complex number version is provided with the name "Scaled Cayley Unitary RNN (scuRNN)"[28]. The difference lies upon the employment of Wirtinger calculus to modify on the gradient descent and activation function. The scaling matrix $\mathbf{D}$ lies on the unit circle and argument of angles are also updated as well.

2.2 Problem Formulation

2.2.1 Stiefel Manifold and Orthographic Retraction

Before we go deeper into the formulation of the structured recurrent neural network and its related algorithms, we give some corresponding descriptions and definitions that will help to develop the constraints and formulate the problem. We define the matrix manifold as the suitable set $\mathcal{M}$ that is a collection of coordinate patches or charts [29]. The coordinate patches or charts are able to do the differential calculus on $\mathcal{M}$.

Consider $\mathcal{N}$ as another manifold and $\mathcal{N} \in \mathcal{M}$, we say that $\mathcal{N}$ is the submanifold of $\mathcal{M}$. Then $\mathcal{N}$ admits at most one differentiable structure that makes it an embedded submanifold of $\mathcal{M}$. One special case of the submanifold is the Stiefel manifold, which is an embedded submanifold of $\mathbb{R}^{n\times p}$. Given the Stiefel manifold $St(n,m) = \{\mathbf{X} \in \mathbb{R}^{n\times m} : \mathbf{X}^T\mathbf{X} = \mathbf{I}\}$, the related tangent space and normal space can be represented as the following [30]:

$$\text{tangent space} : T_{St(n,m)}(\mathbf{X}) = \mathbf{X}\boldsymbol{\Omega} + \mathbf{X}_\perp\mathbf{K} \tag{2.4}$$

$$\text{norm space} : N_{St(n,m)}(\mathbf{X}) = \mathbf{X}\mathbf{S} \quad \mathbf{S} \in \mathcal{S}, \tag{2.5}$$

where $\boldsymbol{\Omega}$ is a skew-symmetric matrix and $\mathbf{K} \in \mathbb{R}^{(n-m)\times m}$. $\mathcal{S}$ stands for the set of symmetric matrix. According to the theorem, we have the orthographic retraction as:

$$R(\mathbf{X}, \mathbf{X}\boldsymbol{\Omega} + \mathbf{X}_\perp\mathbf{K}) = \mathbf{X} + \mathbf{X}\boldsymbol{\Omega} + \mathbf{X}_\perp\mathbf{K} + \mathbf{X}\mathbf{S}, \tag{2.6}$$

and it satisfies $R(\mathbf{X}, \mathbf{X}\boldsymbol{\Omega} + \mathbf{X}_\perp\mathbf{K})^T R(\mathbf{X}, \mathbf{X}\boldsymbol{\Omega} + \mathbf{X}_\perp\mathbf{K}) = \mathbf{I}$ [29]. Including all the conditions from equations (2.4]) - (2.6), we can derive the following:

$$\mathbf{S}^2 + \left(\mathbf{I} + \boldsymbol{\Omega}^T\right)\mathbf{S} + \mathbf{S}\left(\mathbf{I} + \boldsymbol{\Omega}\right) + \boldsymbol{\Omega}^T\boldsymbol{\Omega} + \mathbf{K}^T\mathbf{K} = \mathbf{0} \tag{2.7}$$

One of the special case is when $n = m$, i.e., the orthogonal group $\mathbf{O}_n$. Therefore $\mathbf{K} = \mathbf{0}$ and the corresponding Riccati equation can be simplified as (given the fact that for skew matrix $\boldsymbol{\Omega}$, we have $\boldsymbol{\Omega}^T + \boldsymbol{\Omega} = \mathbf{0}$):

$$\mathbf{S}^2 + 2\mathbf{S} + \boldsymbol{\Omega}^T\boldsymbol{\Omega} = \mathbf{0}. \tag{2.8}$$

The solution to the above question is $\mathbf{S} = \pm\mathbf{I} + \sqrt{\mathbf{I} - \boldsymbol{\Omega}^T\boldsymbol{\Omega}}$. Given the fact that the eigendecomposition of the matrix $\boldsymbol{\Omega}^T\boldsymbol{\Omega} = \mathbf{Q}^T\boldsymbol{\Sigma}\mathbf{Q} = \mathbf{Q}^T\text{Diag}(\sqrt{\lambda}_1, \sqrt{\lambda}_2, \ldots, \sqrt{\lambda}_n)\mathbf{Q}$, we have:

$$\mathbf{S}_\pm = \mathbf{Q}^T\text{Diag}(-1 \pm \sqrt{1-\lambda}_1, \ldots, -1 \pm \sqrt{1-\lambda}_n)\mathbf{Q}. \tag{2.9}$$

Here we take the value with minimum norm, i.e., $\mathbf{S}_+ = -\mathbf{I} + \sqrt{\mathbf{I} - \mathbf{\Omega}^T\mathbf{\Omega}}$, substitute the $\mathbf{S}_+$ in the equation [2.6], we have:

$$\begin{aligned} R(\mathbf{X}, \mathbf{X}\mathbf{\Omega} + \mathbf{X}_\perp\mathbf{K}) &= \mathbf{X} + \mathbf{X}\mathbf{\Omega} + \mathbf{X}_\perp\mathbf{K} + \mathbf{X}\mathbf{S} \\ &= \mathbf{X} + \mathbf{X}\mathbf{\Omega} + \mathbf{X}\mathbf{S} \\ &= \mathbf{X} + \mathbf{X}\mathbf{\Omega} + \mathbf{X}\left(-\mathbf{I} + \sqrt{\mathbf{I} - \mathbf{\Omega}^T\mathbf{\Omega}}\right) \\ &= \mathbf{X}\left(\mathbf{\Omega} + \sqrt{\mathbf{I} - \mathbf{\Omega}^T\mathbf{\Omega}}\right). \end{aligned} \tag{2.10}$$

The formula 2.10 is crucial starting point to introduce an efficient decomposition of the structured asymmetric matrix and it will be utilized multiple times to develop two decomposition framework for the following two SRNNs.

In parallel to the work on unitary decomposition of the weight matrix in RNN, there has been an increasing interest for optimization over the orthogonal group and the Stiefel manifold in neural networks. From some existed literature of the Stiefel manifold and orthogonal group, the above decomposition belongs what is so called manifold retraction [29, 30, 31]. The formulation that utilizes one of the manifold retraction for constraints $\mathbf{W}^T\mathbf{W} = \mathbf{I}$ or $\mathbf{W}^H\mathbf{W} = \mathbf{I}$. One of the possible and efficient ways to decompose the recurrent weight matrix is to leverage the Cayley transform. An real number based orthogonal RNN is proposed by the parameterization of the weight matrix using a scaled version of the Cayley transform [21, 26, 27]. The iteration of the gradient based on properties of skew-symmetric matrices has simplified the propagation process so that computation cost has reduced.

In order to solve the above mentioned problem of constructing the structured recurrent neural network, one of the motivations to decompose the recurrent weight matrix is to utilize the manifold retraction accompanied with the scaling matrix. Besides the Cayley transform, there are other types of manifold retraction methods that are employed to achieve manifold learning. We compares the following three

popular types of manifold retraction mentioned in section 2.1.2 that are designed to satisfy the orthogonal constraints: [30, 31]:

1. Cayley transform: $R_X(\mathbf{X}\mathbf{\Omega}) = (\mathbf{I} + \mathbf{\Omega})^{-1}(\mathbf{I} - \mathbf{\Omega})\mathbf{X}$
2. projection-like: $R_X(\mathbf{X}\mathbf{\Omega}) = (\mathbf{\Omega} + \sqrt{\mathbf{I} - \mathbf{\Omega}^T\mathbf{\Omega}})\mathbf{X}$
3. polar decomposition: $R_X(\mathbf{X}\mathbf{\Omega}) = (\mathbf{I} + \mathbf{\Omega})(\mathbf{I} - \mathbf{\Omega}^2)^{-1/2}\mathbf{X}$
   $= (\mathbf{I} + \mathbf{\Omega})(\mathbf{I} + \mathbf{\Omega}^T\mathbf{\Omega})^{-1/2}\mathbf{X}$

For the square root part of the polar decomposition, since $\mathbf{\Omega}$ is the skew symmetric matrix such that $\mathbf{\Omega}^T = -\mathbf{\Omega}$, and then we substitute into the above formula. Notice that both the polar decomposition and projection-like method rely has the second order term $\mathbf{\Omega}^T\mathbf{\Omega}$. Since it is a symmetric matrix, we adopt the eigenvalue decomposition and diagonalization $\mathbf{\Omega}^T\mathbf{\Omega} = \mathbf{Q}\mathbf{\Sigma}\mathbf{Q}^T = \mathbf{Q}\text{Diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)\mathbf{Q}^T$. We substitute this relation into the three expressions above and use them to replace the weight matrix $\mathbf{W}$, so that we can derive the following three related OC-based RNNs and the related coordinate descent like training algorithms as below:

2.2.2 scoRNN and the corresponding training algorithms

Scaled cayley transform RNN (scoRNN) adopts the first cayley transform formulation in the recurrent weight matrix. We restate the definition and details of the algorithm the scoRNN as follows. For more details about the proof, we refer the readers to [27].

Theorem 1 *Given a differentiable loss function $L = L(\boldsymbol{W})$ w.r.t the recurrent matrix $\boldsymbol{W}$ of the structured RNN. Let $\boldsymbol{W} = (\boldsymbol{I} + \mathbf{\Omega})^{-1}(\boldsymbol{I} - \mathbf{\Omega})\boldsymbol{D}$, where $\mathbf{\Omega} \in \mathbb{R}^{n\times n}$ is the skew-symmetric matrix and $\boldsymbol{D}$ is the fixed diagonal matrix consisting of -1 and 1 entries. We further define $\boldsymbol{Z} = (\boldsymbol{I} + \mathbf{\Omega})^{-1}(\boldsymbol{I} - \mathbf{\Omega})$. Then the gradient of $L(\boldsymbol{W})$ with respect to $\mathbf{\Omega}$ is:*

$$\frac{\partial L}{\partial \mathbf{\Omega}} = \boldsymbol{V}^T - \boldsymbol{V}, \tag{2.11}$$

where $\boldsymbol{V} = (\boldsymbol{I} + \boldsymbol{\Omega})^{-T} \frac{\partial L}{\partial \boldsymbol{W}} (\boldsymbol{D} + \boldsymbol{W}^T)$

2.2.3 orRNN and the corresponding training algorithms

Orthographic retraction RNN (orRNN) is proposed as part of our previous work, which adopts the second projection-like decomposition method. We restate the definition and the coordinate descent like algorithm to train the orRNN can be formulated as follows. For more details about the proof, we refer the readers to [].

Theorem 2 *Given a differentiable loss function L w.r.t the recurrent matrix* $\boldsymbol{W}$ *of the structured RNN. Let* $\boldsymbol{W} = \left(\boldsymbol{\Omega} + \sqrt{\boldsymbol{I} - \boldsymbol{\Omega}^T \boldsymbol{\Omega}}\right) \boldsymbol{D}$*, where* $\boldsymbol{\Omega} \in \mathbb{R}^{n \times n}$ *is the skew-symmetric matrix and* $\boldsymbol{D}$ *is the fixed diagonal matrix. We further define* $\boldsymbol{Z} = \boldsymbol{\Omega} + \sqrt{\boldsymbol{I} - \boldsymbol{\Omega}^T \boldsymbol{\Omega}}$*. Since* $\boldsymbol{\Omega}$ *is a skew-symmetric matrix, we can employ eigendecomposition on the term* $\boldsymbol{\Omega}^T \boldsymbol{\Omega} = \boldsymbol{Q} \boldsymbol{\Sigma} \boldsymbol{Q}^T$*. Then we leverage a coordinate descent based back-propagation to calculate the following gradient terms:*

$$\frac{\partial L}{\partial \boldsymbol{\Omega}} = \left(\frac{\partial L}{\partial \boldsymbol{W}} \boldsymbol{D}\right) - \left(\frac{\partial L}{\partial \boldsymbol{W}} \boldsymbol{D}\right)^T \tag{2.12}$$

$$\frac{\partial L}{\partial \boldsymbol{Q}} = \left[\left(\frac{\partial L}{\partial \boldsymbol{W}} \boldsymbol{D}\right) + \left(\frac{\partial L}{\partial \boldsymbol{W}} \boldsymbol{D}\right)^T\right] \boldsymbol{Q} \tilde{\boldsymbol{\Sigma}} \tag{2.13}$$

$$\frac{\partial L}{\partial \tilde{\boldsymbol{\Sigma}}} = Diag\left(\boldsymbol{Q}^T \left(\frac{\partial L}{\partial \boldsymbol{W}} \boldsymbol{D}\right) \boldsymbol{Q}\right) \tag{2.14}$$

Algorithm 2 Algorithm for orRNN

Input: $\mathbf{\Omega}$, $\mathbf{D}$

Output: $\mathbf{W}$

Initialization : $\mathbf{\Omega}$, $\mathbf{D}$, loss function $L(\mathbf{W})$

eigendecomposition of $\mathbf{Z}$

LOOP Process

for $k = 1$ to n (given iteration) **do**

$\frac{\partial L}{\partial \mathbf{\Omega}} = \left(\frac{\partial L}{\partial \mathbf{W}}\mathbf{D}\right) - \left(\frac{\partial L}{\partial \mathbf{W}}\mathbf{D}\right)^T$

$\frac{\partial L}{\partial \mathbf{Q}} = \left[\left(\frac{\partial L}{\partial \mathbf{W}}\mathbf{D}\right) + \left(\frac{\partial L}{\partial \mathbf{W}}\mathbf{D}\right)^T\right]\mathbf{Q}\tilde{\mathbf{\Sigma}}$

$\frac{\partial L}{\partial \mathbf{\Sigma}} = \text{Diag}\left(\mathbf{Q}^T\left(\frac{\partial L}{\partial \mathbf{W}}\mathbf{D}\right)\mathbf{Q}\right)$

$\mathbf{\Omega}^{k+1} = \mathbf{\Omega}^k - \alpha\frac{\partial L}{\partial \mathbf{\Omega}}$

$\mathbf{Q}^{k+1} = \mathbf{Q}^k - \alpha\frac{\partial L}{\partial \mathbf{Q}}$

$\mathbf{\Sigma}^{k+1} = \mathbf{\Sigma}^k - \alpha\frac{\partial L}{\partial \mathbf{\Sigma}}$

$\mathbf{W}^{k+1} = \left(\mathbf{\Omega}^{k+1} + \mathbf{Q}^{k+1}\mathbf{\Sigma}^{k+1}\mathbf{Q}^{k+1^T}\right)\mathbf{D}$

end for=0

Theorem 3 *Given a differentiable loss function L w.r.t the recurrent matrix $\boldsymbol{W}$ of the structured RNN. Let $\boldsymbol{W} = \left(\mathbf{\Omega} + \sqrt{\boldsymbol{I} - \mathbf{\Omega}^T\mathbf{\Omega}}\right)\boldsymbol{D}$, where $\mathbf{\Omega} \in \mathbb{R}^{n\times n}$ is the skew-symmetric matrix and $\boldsymbol{D}$ is the fixed diagonal matrix. We further define $\boldsymbol{Z} = \mathbf{\Omega} + \sqrt{\boldsymbol{I} - \mathbf{\Omega}^T\mathbf{\Omega}}$. Since $\mathbf{\Omega}$ is a skew-symmetric matrix, we can employ eigendecomposition on the term $\mathbf{\Omega}^T\mathbf{\Omega} = \boldsymbol{Q}\boldsymbol{\Sigma}\boldsymbol{Q}^T$. Then we leverage a coordinate descent based back-propagation to calculate the following gradient terms:*

$$\frac{\partial L}{\partial \boldsymbol{\Omega}} = \left(\frac{\partial L}{\partial \boldsymbol{W}}\boldsymbol{D}\right) - \left(\frac{\partial L}{\partial \boldsymbol{W}}\boldsymbol{D}\right)^T \tag{2.15}$$

$$\frac{\partial L}{\partial \boldsymbol{Q}} = \left[\left(\frac{\partial L}{\partial \boldsymbol{W}}\boldsymbol{D}\right) + \left(\frac{\partial L}{\partial \boldsymbol{W}}\boldsymbol{D}\right)^T\right]\boldsymbol{Q}\tilde{\boldsymbol{\Sigma}} \tag{2.16}$$

$$\frac{\partial L}{\partial \tilde{\boldsymbol{\Sigma}}} = Diag\left(\boldsymbol{Q}^T\left(\frac{\partial L}{\partial \boldsymbol{W}}\boldsymbol{D}\right)\boldsymbol{Q}\right) \tag{2.17}$$

Proof: We simplify the $\mathbf{Z}$ as follows:

$$\begin{aligned}
\mathbf{Z} &= \boldsymbol{\Omega} + \sqrt{\mathbf{I} - \boldsymbol{\Omega}^T\boldsymbol{\Omega}} \\
&= \boldsymbol{\Omega} + \sqrt{\mathbf{I} - \mathbf{Q}\boldsymbol{\Sigma}\mathbf{Q}^T} \\
&= \boldsymbol{\Omega} + \sqrt{\mathbf{Q}\left(\mathbf{I} - \boldsymbol{\Sigma}\right)\mathbf{Q}^T} \\
&= \boldsymbol{\Omega} + \mathbf{Q}\left(\mathbf{I} - \boldsymbol{\Sigma}\right)^{\frac{1}{2}}\mathbf{Q}^T \\
&= \boldsymbol{\Omega} + \mathbf{Q}\tilde{\boldsymbol{\Sigma}}\mathbf{Q}^T
\end{aligned}$$

where $\tilde{\boldsymbol{\Sigma}} = \text{diag}\left(\sqrt{1-\lambda_1}, \sqrt{1-\lambda_2}, \ldots, \sqrt{1-\lambda_n}\right)$ and $\lambda_1, \lambda_2, \ldots, \lambda_n$ are eigenvalues of $\boldsymbol{\Sigma}$.

$$\begin{aligned}
\frac{\partial L(\mathbf{W})}{\partial \boldsymbol{\Omega}_{i,j}} &= \text{Tr}\left[\left(\frac{\partial L}{\partial \mathbf{W}}\mathbf{D}\right)^T\frac{\partial \mathbf{Z}}{\partial \boldsymbol{\Omega}_{ij}}\right] \\
&= \text{Tr}\left[\left(\frac{\partial L}{\partial \mathbf{W}}\mathbf{D}\right)^T\frac{\partial\left(\boldsymbol{\Omega} + \mathbf{Q}\tilde{\boldsymbol{\Sigma}}\mathbf{Q}^T\right)}{\partial \boldsymbol{\Omega}_{ij}}\right] \\
&= \text{Tr}\left[\left(\frac{\partial L}{\partial \mathbf{W}}\mathbf{D}\right)^T\frac{\partial \boldsymbol{\Omega}}{\partial \boldsymbol{\Omega}_{ij}}\right] \\
&= \text{Tr}\left[\left(\frac{\partial L}{\partial \mathbf{W}}\mathbf{D}\right)^T\left(\mathbf{E}_{i,j} - \mathbf{E}_{j,i}\right)\right] \\
&= \left[\left(\left(\frac{\partial L}{\partial \mathbf{W}}\mathbf{D}\right)^T\right)^T\right]_{i,j} - \left[\left(\frac{\partial L}{\partial \mathbf{W}}\mathbf{D}\right)^T\right]_{i,j} \\
&= \left(\frac{\partial L}{\partial \mathbf{W}}\mathbf{D}\right)_{i,j} - \left(\frac{\partial L}{\partial \mathbf{W}}\mathbf{D}\right)^T_{i,j}.
\end{aligned}$$

For the third equality above, we use the fact that the element-wise derivative of a skew-symmetric matrix is $(\mathbf{E}_{i,j} - \mathbf{E}_{j,i})$, where $\mathbf{E}_{i,j}$ stands for the singleentry matrix that contains 1 at i-th row and j-th column and rest of the elements are 0. For the fifth entry, we utilize the proof from the appendix of [27] to transform the trace to the element-wise expression. The readers are suggested to read the reference for more details. And therefore, we have:

$$\frac{\partial L}{\partial \mathbf{\Omega}} = \left(\frac{\partial L}{\partial \mathbf{W}}\mathbf{D}\right) - \left(\frac{\partial L}{\partial \mathbf{W}}\mathbf{D}\right)^T \tag{2.18}$$

Similarity, we can derive the $\frac{\partial L}{\partial \mathbf{Q}}$ as:

$$\begin{aligned}
\frac{\partial L(\mathbf{W})}{\partial \mathbf{Q}_{ij}} &= \mathrm{Tr}\left[\left(\frac{\partial L}{\partial \mathbf{W}}\mathbf{D}\right)^T \frac{\partial \mathbf{Z}}{\partial \mathbf{Q}_{ij}}\right] \\
&= \mathrm{Tr}\left[\left(\frac{\partial L}{\partial \mathbf{W}}\mathbf{D}\right)^T \frac{\partial\left(\mathbf{\Omega} + \mathbf{Q}\tilde{\mathbf{\Sigma}}\mathbf{Q}^T\right)}{\partial \mathbf{Q}_{ij}}\right] \\
&= \mathrm{Tr}\left[\left(\frac{\partial L}{\partial \mathbf{W}}\mathbf{D}\right)^T \frac{\partial\left(\mathbf{Q}\tilde{\mathbf{\Sigma}}\mathbf{Q}^T\right)}{\partial \mathbf{Q}_{ij}}\right] \\
&= \mathrm{Tr}\left[\left(\frac{\partial L}{\partial \mathbf{W}}\mathbf{D}\right)\left(\left(\mathbf{Q}\tilde{\mathbf{\Sigma}}\right)\mathbf{E}_{j,i} + \mathbf{E}_{i,j}\left(\tilde{\mathbf{\Sigma}}\mathbf{Q}^T\right)\right)\right] \\
&= \mathrm{Tr}\left[\left(\frac{\partial L}{\partial \mathbf{W}}\mathbf{D}\right)\left(\left(\mathbf{Q}\tilde{\mathbf{\Sigma}}\right)\mathbf{E}_{j,i}\right)\right] \\
&\quad + \mathrm{Tr}\left[\left(\frac{\partial L}{\partial \mathbf{W}}\mathbf{D}\right)\mathbf{E}_{i,j}\left(\tilde{\mathbf{\Sigma}}\mathbf{Q}^T\right)\right] \\
&= \left[\left(\frac{\partial L}{\partial \mathbf{W}}\mathbf{D}\right)^T\left(\mathbf{Q}\tilde{\mathbf{\Sigma}}\right)\right]_{i,j} + \left[\left(\frac{\partial L}{\partial \mathbf{W}}\mathbf{D}\right)\left(\mathbf{Q}\tilde{\mathbf{\Sigma}}^T\right)\right]_{i,j}.
\end{aligned}$$

Since $\tilde{\mathbf{\Sigma}}$ is the diagonal matrix and $\tilde{\mathbf{\Sigma}}^T = \tilde{\mathbf{\Sigma}}$, we have:

$$\frac{\partial L(\mathbf{W})}{\partial \mathbf{Q}} = \left[\left(\frac{\partial L}{\partial \mathbf{W}}\mathbf{D}\right)^T + \left(\frac{\partial L}{\partial \mathbf{W}}\mathbf{D}\right)\right]\mathbf{Q}\tilde{\mathbf{\Sigma}} \tag{2.19}$$

Last, for $\frac{\partial L(\mathbf{W})}{\partial \tilde{\mathbf{\Sigma}}}$, we can derive as the following:

$$\begin{aligned}\frac{\partial L(\mathbf{W})}{\partial \tilde{\mathbf{\Sigma}}_{ij}} &= \mathrm{Tr}\left[\left(\frac{\partial L}{\partial \mathbf{W}}\mathbf{D}\right)^T \frac{\partial \mathbf{Z}}{\partial \tilde{\mathbf{\Sigma}}_{ij}}\right] \\ &= \mathrm{Tr}\left[\left(\frac{\partial L}{\partial \mathbf{W}}\mathbf{D}\right)^T \frac{\partial\left(\mathbf{\Omega} + \mathbf{Q}\tilde{\mathbf{\Sigma}}\mathbf{Q}^T\right)}{\partial \tilde{\mathbf{\Sigma}}_{ij}}\right] \\ &= \mathrm{Tr}\left[\left(\frac{\partial L}{\partial \mathbf{W}}\mathbf{D}\right)^T \frac{\partial\left(\mathbf{Q}\tilde{\mathbf{\Sigma}}\mathbf{Q}^T\right)}{\partial \tilde{\mathbf{\Sigma}}_{ij}}\right]\end{aligned}$$

Since $\frac{\partial\left(\mathbf{Q}\tilde{\mathbf{\Sigma}}\mathbf{Q}^T\right)}{\partial \tilde{\mathbf{\Sigma}}_{ij}} = \mathbf{Q}_{:,i}\mathbf{Q}_{:,i}^T = \mathbf{Q}\mathbf{e}_i\mathbf{e}_i^T\mathbf{Q}_i^T = \mathbf{Q}\mathbf{E}_{i,i}\mathbf{Q}^T$, where $\mathbf{Q}_{:,i}$ stands for the matrix contains only i-th column of $\mathbf{Q}$ and the rest elements are 0, $\mathbf{e}_i$ is single-entry column vector where only the i-th element is 1 and rest entries are 0, and $\mathbf{E}_{i,i}$ is the single-entry matrix where only the (i,j)-th entry is 1 and rest entries are 0. Therefore, we can simply as:

$$\begin{aligned}\mathrm{Tr}\left[\left(\frac{\partial L}{\partial \mathbf{W}}\mathbf{D}\right)^T \frac{\partial\left(\mathbf{Q}\tilde{\mathbf{\Sigma}}\mathbf{Q}^T\right)}{\partial \tilde{\mathbf{\Sigma}}_{ij}}\right] &= \mathrm{Tr}\left[\left(\frac{\partial L}{\partial \mathbf{W}}\mathbf{D}\right)^T \mathbf{Q}\mathbf{E}_{i,i}\mathbf{Q}^T\right] \\ &= \left[\left[\left(\frac{\partial L}{\partial \mathbf{W}}\mathbf{D}\right)^T \mathbf{Q}\right]^T \left[\mathbf{Q}^T\right]^T\right]_{i,i} \\ &= \left(\mathbf{Q}^T\left(\frac{\partial L}{\partial \mathbf{W}}\mathbf{D}\right)\mathbf{Q}\right)_{i,i}\end{aligned}$$

Therefore, we have the followings:

$$\frac{\partial L(\mathbf{W})}{\partial \tilde{\mathbf{\Sigma}}} = \mathrm{Diag}\left(\mathbf{Q}^T\left(\frac{\partial L}{\partial \mathbf{W}}\mathbf{D}\right)\mathbf{Q}\right) \tag{2.20}$$

2.2.4 pdRNN and the corresponding training algorithms

Similarly, for the polar decomposition RNN (pdRNN), it shares a similar formulation as the orRNN with some differences in the details about the coordinate descent algorithm. We have the following theorem and the proof the theorem is given in the appendix.

Theorem 4 *Given a differentiable loss function L w.r.t the recurrent matrix $\boldsymbol{W}$ of the structured RNN. Let $\boldsymbol{W} = ((\boldsymbol{I}+\boldsymbol{\Omega})(\boldsymbol{I}+\boldsymbol{\Omega}^T\boldsymbol{\Omega})^{-\frac{1}{2}})\boldsymbol{D}$, where $\boldsymbol{\Omega} \in \mathbb{R}^{n\times n}$ is the skew-symmetric matrix and $\boldsymbol{D}$ is the fixed diagonal matrix. We further define $\boldsymbol{Z} = (\boldsymbol{I}+\boldsymbol{\Omega})(\boldsymbol{I}+\boldsymbol{\Omega}^T\boldsymbol{\Omega})^{\frac{1}{2}}$. Since $\boldsymbol{\Omega}$ is a skew-symmetric matrix, we can employ eigen-decomposition on the term $\boldsymbol{\Omega}^T\boldsymbol{\Omega} = \boldsymbol{Q}\boldsymbol{\Sigma}\boldsymbol{Q}^T$. Then we leverage a coordinate descent based back-propagation to calculate the following gradient terms:*

$$\frac{\partial L}{\partial \boldsymbol{\Omega}} = \left[\left(\frac{\partial L}{\partial \boldsymbol{W}}\boldsymbol{D}\right)\boldsymbol{Q}\tilde{\boldsymbol{\Sigma}}\boldsymbol{Q}^T\right] - \left[\boldsymbol{Q}\tilde{\boldsymbol{\Sigma}}\boldsymbol{Q}^T\left(\frac{\partial L}{\partial \boldsymbol{W}}\boldsymbol{D}\right)^T\right] \tag{2.21}$$

$$\frac{\partial L}{\partial \boldsymbol{Q}} = \left[\left(\frac{\partial L}{\partial \boldsymbol{W}}\boldsymbol{D}\right)^T(\boldsymbol{I}+\boldsymbol{\Omega})\boldsymbol{Q}\tilde{\boldsymbol{\Sigma}}\right] - \left[(\boldsymbol{I}+\boldsymbol{\Omega})^T\left(\frac{\partial L}{\partial \boldsymbol{W}}\boldsymbol{D}\right)\boldsymbol{Q}\tilde{\boldsymbol{\Sigma}}\right] \tag{2.22}$$

$$\frac{\partial L}{\partial \tilde{\boldsymbol{\Sigma}}} = Diag\left(\boldsymbol{Q}^T(\boldsymbol{I}+\boldsymbol{\Omega})^T\left(\frac{\partial L}{\partial \boldsymbol{W}}\boldsymbol{D}\right)\boldsymbol{Q}\right) \tag{2.23}$$

Proof:

For proof of the $\frac{\partial L(\mathbf{W})}{\partial \boldsymbol{\Omega}}$, we have the following derivation:

$$\begin{aligned}
\frac{\partial L(\mathbf{W})}{\partial \boldsymbol{\Omega}_{ij}} &= \mathrm{Tr}\left[\left(\frac{\partial L}{\partial \mathbf{W}}\mathbf{D}\right)^T\frac{\partial \mathbf{Z}}{\partial \boldsymbol{\Omega}_{ij}}\right] \\
&= \mathrm{Tr}\left[\left(\frac{\partial L}{\partial \mathbf{W}}\mathbf{D}\right)^T\frac{\partial\left((\mathbf{I}+\boldsymbol{\Omega})\mathbf{Q}\tilde{\boldsymbol{\Sigma}}\mathbf{Q}^T\right)}{\partial \boldsymbol{\Omega}_{ij}}\right] \\
&= \mathrm{Tr}\left[\left(\frac{\partial L}{\partial \mathbf{W}}\mathbf{D}\right)\left(\left(\mathbf{E}_{i,j}-\mathbf{E}_{j,i}\right)\mathbf{Q}\tilde{\boldsymbol{\Sigma}}\mathbf{Q}^T\right)\right] \\
&= \mathrm{Tr}\left[\left(\frac{\partial L}{\partial \mathbf{W}}\mathbf{D}\right)^T\left(\mathbf{E}_{i,j}\mathbf{Q}\tilde{\boldsymbol{\Sigma}}\mathbf{Q}^T\right)\right] \\
&\quad + \mathrm{Tr}\left[\left(\frac{\partial L}{\partial \mathbf{W}}\mathbf{D}\right)^T\mathbf{E}_{j,i}\left(\mathbf{Q}\tilde{\boldsymbol{\Sigma}}\mathbf{Q}^T\right)\right] \\
&= \left[\left(\frac{\partial L}{\partial \mathbf{W}}\mathbf{D}\right)^T\left(\mathbf{Q}\tilde{\boldsymbol{\Sigma}}\mathbf{Q}^T\right)\right]_{i,j} + \\
&\quad \left[\left(\mathbf{Q}\tilde{\boldsymbol{\Sigma}}\mathbf{Q}^T\right)\left(\frac{\partial L}{\partial \mathbf{W}}\mathbf{D}\right)^T\right]_{i,j}.
\end{aligned}$$

In order to derive the partial derivative of the loss function against the orthogonal term $\mathbf{Q}$, i.e., $\frac{\partial L}{\partial \mathbf{Q}}$, we introduce the following corollary:

Corollary 1: according to the [32] chapter 2.4.2, we have the following matrix partial derivative:

$$\frac{\partial \mathbf{A}\mathbf{X}\mathbf{B}\mathbf{X}^T}{\partial \mathbf{X}_{i,j}} = \mathbf{A}\frac{\partial \mathbf{X}\mathbf{B}\mathbf{X}^T}{\partial \mathbf{X}_{i,j}} = \mathbf{A}\mathbf{X}\mathbf{B}\mathbf{E}_{j,i} - \mathbf{A}\mathbf{E}_{i,j}\mathbf{B}\mathbf{X}^T \tag{2.24}$$

where $\mathbf{E}_{i,j}$ is the single-entry matrix that only element on ith row and jth column will be 1 and the rest of the elements in the matrix are 0. Based on the corollary, we can plug in the partial derivative of the loss function against the orthogonal term $\frac{\partial L}{\partial \mathbf{Q}}$ to derive the followings:

$$\begin{aligned}
\frac{\partial L(\mathbf{W})}{\partial \mathbf{Q}_{ij}} &= \mathrm{Tr}\left[\left(\frac{\partial L}{\partial \mathbf{W}}\mathbf{D}\right)^T \frac{\partial \mathbf{Z}}{\partial \mathbf{Q}_{ij}}\right] \\
&= \mathrm{Tr}\left[\left(\frac{\partial L}{\partial \mathbf{W}}\mathbf{D}\right)^T \frac{\partial\left((\mathbf{I}+\mathbf{\Omega})\mathbf{Q}\tilde{\mathbf{\Sigma}}\mathbf{Q}^T\right)}{\partial \mathbf{Q}_{ij}}\right] \\
&= \mathrm{Tr}\left[\left(\frac{\partial L}{\partial \mathbf{W}}\mathbf{D}\right)^T (\mathbf{I}+\mathbf{\Omega})\mathbf{Q}\tilde{\mathbf{\Sigma}}\mathbf{E}_{j,i}\right] \\
&\quad - \mathrm{Tr}\left[\left(\frac{\partial L}{\partial \mathbf{W}}\mathbf{D}\right)^T (\mathbf{I}+\mathbf{\Omega})\,\mathbf{E}_{i,j}\tilde{\mathbf{\Sigma}}\mathbf{Q}^T\right] \\
&= \left[\left(\frac{\partial L}{\partial \mathbf{W}}\mathbf{D}\right)^T (\mathbf{I}+\mathbf{\Omega})\mathbf{Q}\tilde{\mathbf{\Sigma}}\right]_{i,j} - \\
&\quad \left[(\mathbf{I}+\mathbf{\Omega})^T\left(\frac{\partial L}{\partial \mathbf{W}}\mathbf{D}\right)\mathbf{Q}\tilde{\mathbf{\Sigma}}\right]_{i,j}.
\end{aligned}$$

Therefore, we can extend this to the whole elements of the matrix and have proved that $\frac{\partial L}{\partial \mathbf{Q}} = \left[\left(\frac{\partial L}{\partial \mathbf{W}}\mathbf{D}\right)^T (\mathbf{I}+\mathbf{\Omega})\mathbf{Q}\tilde{\mathbf{\Sigma}}\right]_{i,j} - \left[(\mathbf{I}+\mathbf{\Omega})^T \left(\frac{\partial L}{\partial \mathbf{W}}\mathbf{D}\right)\mathbf{Q}\tilde{\mathbf{\Sigma}}\right]$ And finally, we derive the $\frac{\partial L}{\partial \tilde{\mathbf{\Sigma}}}$ as follows,

$$\begin{aligned}
\frac{\partial L(\mathbf{W})}{\partial \tilde{\mathbf{\Sigma}}_{ij}} &= \mathrm{Tr}\left[\left(\frac{\partial L}{\partial \mathbf{W}}\mathbf{D}\right)^T \frac{\partial \mathbf{Z}}{\partial \tilde{\mathbf{\Sigma}}_{ij}}\right] \\
&= \mathrm{Tr}\left[\left(\frac{\partial L}{\partial \mathbf{W}}\mathbf{D}\right)^T \frac{\partial\left((\mathbf{I}+\mathbf{\Omega})\mathbf{Q}\tilde{\mathbf{\Sigma}}\mathbf{Q}^T\right)}{\partial \tilde{\mathbf{\Sigma}}_{ij}}\right] \\
&= \mathrm{Tr}\left[\left(\frac{\partial L}{\partial \mathbf{W}}\mathbf{D}\right)^T (\mathbf{I}+\mathbf{\Omega})\,\mathbf{Q}\mathbf{E}_{i,j}\mathbf{Q}^T\right] \\
&= \left[\mathbf{Q}^T(\mathbf{I}+\mathbf{\Omega})^T\left(\frac{\partial L}{\partial \mathbf{W}}\mathbf{D}\right)\mathbf{Q}\right]_{i,j}.
\end{aligned}$$

Since $\tilde{\mathbf{\Sigma}}_{ij}$ is a diagonal matrix, we can represent the above fourth equality as $\left[\mathbf{Q}^T(\mathbf{I}+\mathbf{\Omega})^T\left(\frac{\partial L}{\partial \mathbf{W}}\mathbf{D}\right)\mathbf{Q}\right]_{i,j} = \left[\mathbf{Q}^T(\mathbf{I}+\mathbf{\Omega})^T\left(\frac{\partial L}{\partial \mathbf{W}}\mathbf{D}\right)\mathbf{Q}\right]_{i,i}$, which is essentially the diagonal elements of the term in the square bracket. Hence we have proved that $\frac{\partial L}{\partial \tilde{\mathbf{\Sigma}}} = \mathrm{Diag}\left(\mathbf{Q}^T(\mathbf{I}+\mathbf{\Omega})^T\left(\frac{\partial L}{\partial \mathbf{W}}\mathbf{D}\right)\mathbf{Q}\right)$. The corresponding algorithm is given in algorithm 3.

2.2.5 Details of The Architecture

For the activation function, we consider the modReLU function, which attracts the attentions recently due to its flexibility cooperation with the bias term [27, 10]. We employ here to replace the original ReLU function in the output unit and the modReLU function is defined as follows:

$$\begin{aligned}
\sigma_{\mathrm{modReLU}}(z) &= \frac{z}{|z|}\sigma_{\mathrm{ReLU}}(|z|+b) \\
&= \begin{cases} \frac{z}{|z|}(|z|+b) & \text{if } |z|+b \leq 0 \\ 0 & \text{if } |z|+b < 0. \end{cases}
\end{aligned} \tag{2.25}$$

Algorithm 3 Algorithm for pdRNN

Input: $\mathbf{\Omega}$, $\mathbf{D}$

Output: $\mathbf{W}$

Initialization : $\mathbf{\Omega}$, $\mathbf{D}$, loss function $L(\mathbf{W})$

1: eigendecomposition of $\mathbf{Z}$

LOOP Process

2: **for** $k = 1$ to n (given iteration) **do**

3: $\frac{\partial L}{\partial \mathbf{\Omega}} = \left[\left(\frac{\partial L}{\partial \mathbf{W}}\mathbf{D}\right)\mathbf{Q}\tilde{\mathbf{\Sigma}}\mathbf{Q}^T\right] - \left[\mathbf{Q}\tilde{\mathbf{\Sigma}}\mathbf{Q}^T\left(\frac{\partial L}{\partial \mathbf{W}}\mathbf{D}\right)^T\right]$

4: $\frac{\partial L}{\partial \mathbf{Q}} = \left(\frac{\partial L}{\partial \mathbf{W}}\mathbf{D}\right)^T(\mathbf{I}+\mathbf{\Omega})\mathbf{Q}\tilde{\mathbf{\Sigma}} - (\mathbf{I}+\mathbf{\Omega})^T\left(\frac{\partial L}{\partial \mathbf{W}}\mathbf{D}\right)\mathbf{Q}\tilde{\mathbf{\Sigma}}$

5: $\frac{\partial L}{\partial \tilde{\mathbf{\Sigma}}} = \text{Diag}\left(\mathbf{Q}^T(\mathbf{I}+\mathbf{\Omega})^T\left(\frac{\partial L}{\partial \mathbf{W}}\mathbf{D}\right)\mathbf{Q}\right)$

6: $\mathbf{\Omega}^{k+1} = \mathbf{\Omega}^k - \alpha\frac{\partial L}{\partial \mathbf{\Omega}}$

7: $\mathbf{Q}^{k+1} = \mathbf{Q}^k - \alpha\frac{\partial L}{\partial \mathbf{Q}}$

8: $\mathbf{\Sigma}^{k+1} = \mathbf{\Sigma}^k - \alpha\frac{\partial L}{\partial \mathbf{\Sigma}}$

9: $\mathbf{W}^{k+1} = ((\mathbf{I}+\mathbf{\Omega}^{k+1})(\mathbf{I}+{\mathbf{\Omega}^{k+1}}^T\mathbf{\Omega}^{k+1})^{-\frac{1}{2}})\mathbf{D}$

10: **end for**=0

For the initialization of the parameter matrix $\mathbf{\Omega}$, we choose the same setting as what mentioned in [27], i.e., a block diagonal form by a special orthogonal transformation as follows:

$$\begin{bmatrix} 0 & \lambda_1 & 0 & \cdots & 0 & \\ -\lambda_1 & 0 & & & & \\ 0 & & 0 & \lambda_2 & & 0 \\ & & -\lambda_2 & 0 & & \\ \vdots & & & & \ddots & \vdots \\ & & & & 0 & \lambda_r \\ 0 & & 0 & \cdots & -\lambda_r & 0 \end{bmatrix} \tag{2.26}$$

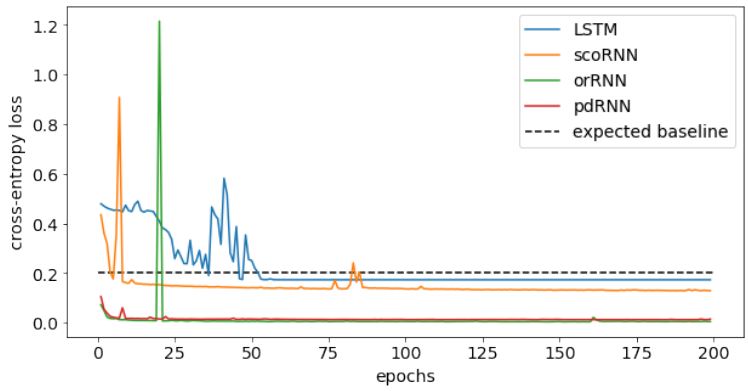
where $\lambda_j = \sqrt{\frac{1-\cos(n_j)}{1+\cos(n_j)}},\ j = 1, \ldots, r$ and n_j is sampled from the uniform distribution $\mathcal{U}[0, \pi/2]$.

2.3 Baseline Experiments and Pseudo Spectrum Visualization

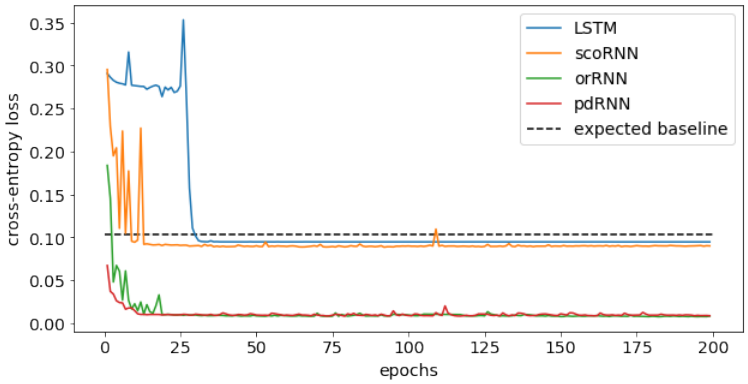
In the following two baseline experiments (adding and copying task), we compare our orRNN against the scoRNN and the vanilla LSTM. For each experiment, we have tuned the hyperparameters for demonstration purpose. The generalization of our structure to other similar sequential learning problems is required for tuning to get expected performance. We first introduce the experiment setting of the two synthetic dataset.

2.3.1 Synthetic Dataset: Copying

The copying problem is a standard synthetic problem that is adopted by some previous research [10, 21, 23, 27]. Digits 0-9 serve as the input to the recurrent neural network, where 0 is regarded as the blank mark and 9 is regarded as the marker. The RNN receives an input sequence of length $T + 2L$, where T is the length of the blank mark and L is the length of the copying sequence which are uniformly sampled from classes 1-8. The number 9 is placed L time steps from the end, which serves as the starting anchor for repeating the copying sequence. The goal for the RNN is to output zeros until it sees a 9. Therefore, the copying sequence must propagate from the beginning to the end of the sequence for a RNN to successfully learn this task, making it critical to avoid vanishing/exploding gradients. A baseline strategy with which to compare machine performance is that of outputting 0 until the machine sees a 9, and then outputting 10 elements randomly sampled from classes 1-8. The expected cross-entropy for such a strategy is $\frac{L\log(8)}{T+2L}$. An input-output example with



(a)



(b)

Figure 2.1. Cross entropy of scoRNN, orRNN and pdRNN on the copying problem with delay length (a) T=1000 and (b) T=2000.

sequence length of 5+10 (5 is length of the copying sequence, $T = 10$) is given for illustration purpose.

$$\text{Input} : [5\,3\,1\,6\,2\,0\,0\,0\,0\,0\,0\,0\,0\,0\,9\,0\,0\,0\,0]$$

$$\text{Output} : [0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,5\,3\,1\,6\,2]$$

2.3.2 Synthetic Dataset: Adding

We adopt the same problem setting as [27] to redo the adding problem and we briefly restate the problem here. Two sequences of length T which are concurrently

imported to the RNN, the first one contains numbers which are sampled uniformly within $\mathcal{U}[0, 1)$, where $\mathcal{U}$ stands for the uniform distribution. The second one contains two markers of "1" while the rest of the entries are zeros, where the first "1" lies randomly within the first half of the sequence (within the interval $[1, \frac{T}{2})$) and the second "1" lies randomly within the second half of the sequence (within the interval $[\frac{T}{2}, T)$). The label for each pair of sequences is the summation of the two entries marked by "1". The goal is to detect the relevant information in the first sequence among random noise. Due to the vanishing gradient and exploding problems, the training becomes challenging as the increasing of the sequence length. The baseline is provided as predicting "1" regardless of the sequence, which gives an expected mean squared error (MSE) of approximately 0.167. An input-output example with sequence length of ten is given as follows (for illustrating purpose , not the exact input for the RNN).

$$\begin{aligned} \text{Input} : & [0.01, 0.02, 0.78, 0.05, 0.34, 0.64, 0.32, 0.89, 0.12, 0.45] \\ & [0, \quad 0, \quad 1, \quad 0, \quad 0, \quad 0, \quad 0, \quad 0, \quad 1, \quad 0] \\ \text{Output} : & \ 0.78 + 0.12 = 0.90 \end{aligned}$$

2.3.3 Performance Analysis and Orthogonality Checking

We list the number of parameters in all RNN and dimension of the weight matrix for both the adding and copying tasks in Table 2.1 and Table 2.2. For adding task with sequence length of 200, we keep the same parameter setting as shown in [27] for scoRNN and LSTM. For our orRNN, we only use a recurrent weight matrix of dimension of 50. The total number of the parameters in orRNN is approximately 1/5 of the scoRNN and 1/11 of the LSTM and we can outperform both of two in adding task. For adding task with sequence length of 400 and 800, we keep the same

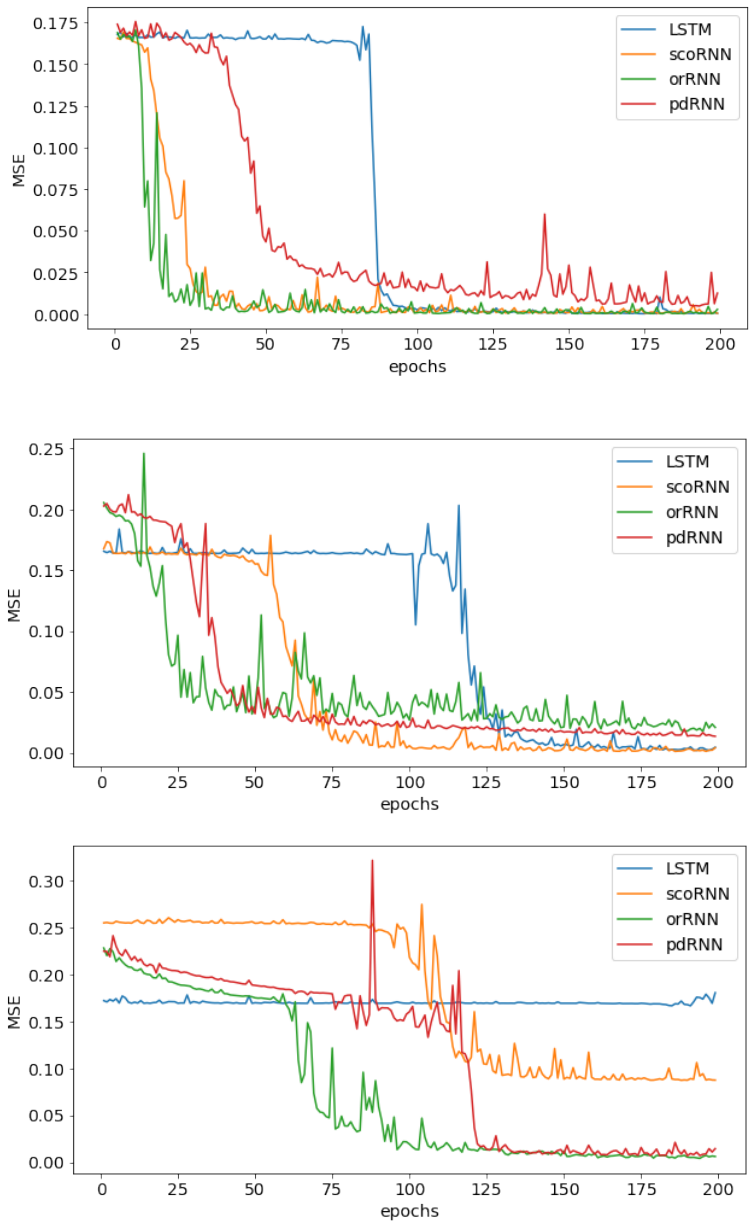


Figure 2.2. Test set mean squared error (MSE) scoRNN, orRNN and pdRNN on the adding problem with sequence lengths of T=200(top), T=400(middle), and T=800(bottom).

parameter setting as shown in [27] for scoRNN and LSTM and for our orRNN, we use a recurrent weight matrix of dimension of 100. The total number of the parameters in orRNN is less than that of the scoRNN and 1/3 of the LSTM. As shown in Fig. 2.2, for all the three adding task, the orRNN and scoRNN can outperform the LSTM. The orRNN outperform the scoRNN in Adding 200 and have almost the same performance as scoRNN for Adding 400 and 800 with less parameters.

For copying, we still keep the same setting as shown in [27] for scoRNN and LSTM and for orRNN, we only use a recurrent weight matrix of dimension of 100. The total number of the parameters in orRNN is less than scoRNN and approximately 1/4 of the LSTM and we can outperform both of two in the two copying tasks. According to the description in section IV.A of [27], the expected baseline in the copying task will be $\frac{L\log(8)}{T+20}$, where the L stands for the length of the copying part, which is set as 100 in our experiment setting. As shown in Fig. 2.1, The orRNN has the best performance in both of the two copying tasks and reach below the theoretical baseline while the rest of the two can only have slightly better performance than the theoretically expected baseline.

Furthermore, we leverage Frobenious norm $\left\|\mathbf{W}^T\mathbf{W}-\mathbf{I}\right\|_F$ as a metric to determine the orthogonality check for the recurrent weight matrix and we track the variation of this metric during the iterations. We just use the adding task as the reference to point out some findings. As shown in Fig. 2.3 and Fig. 2.4, the scoRNN is better than our orRNN for keeping the orthogonality. The trend of the orRNN will rising for some iterations and reach a kind of "saturation" or stable status while the scoRNN almost maintain a tiny vibration during its iterations. But as pointed out in [25], it is not necessary to strictly constrain the orthogonality conditions, and based on our results here, we demonstrate the similar findings as well.

Table 2.1. Comparison of the weight matrix dimension and number of parameters of Adding task

Model	Sequence Length	# Dimension	# Parameters
orRNN	200	50	10201
scoRNN	200	170	58481
LSTM	200	170	117811
orRNN	400	100	42009
scoRNN	400	170	58481
LSTM	400	170	117811
orRNN	800	50	10201
scoRNN	800	170	58481
LSTM	800	170	117811

Table 2.2. Comparison of the weight matrix dimension and number of parameters of Copying task

Model	Sequence Length	# Dimension	# Parameters
orRNN	1000	100	42009
scoRNN	1000	190	76009
LSTM	1000	190	154479
orRNN	2000	100	42009
scoRNN	2000	190	76009
LSTM	2000	190	154479

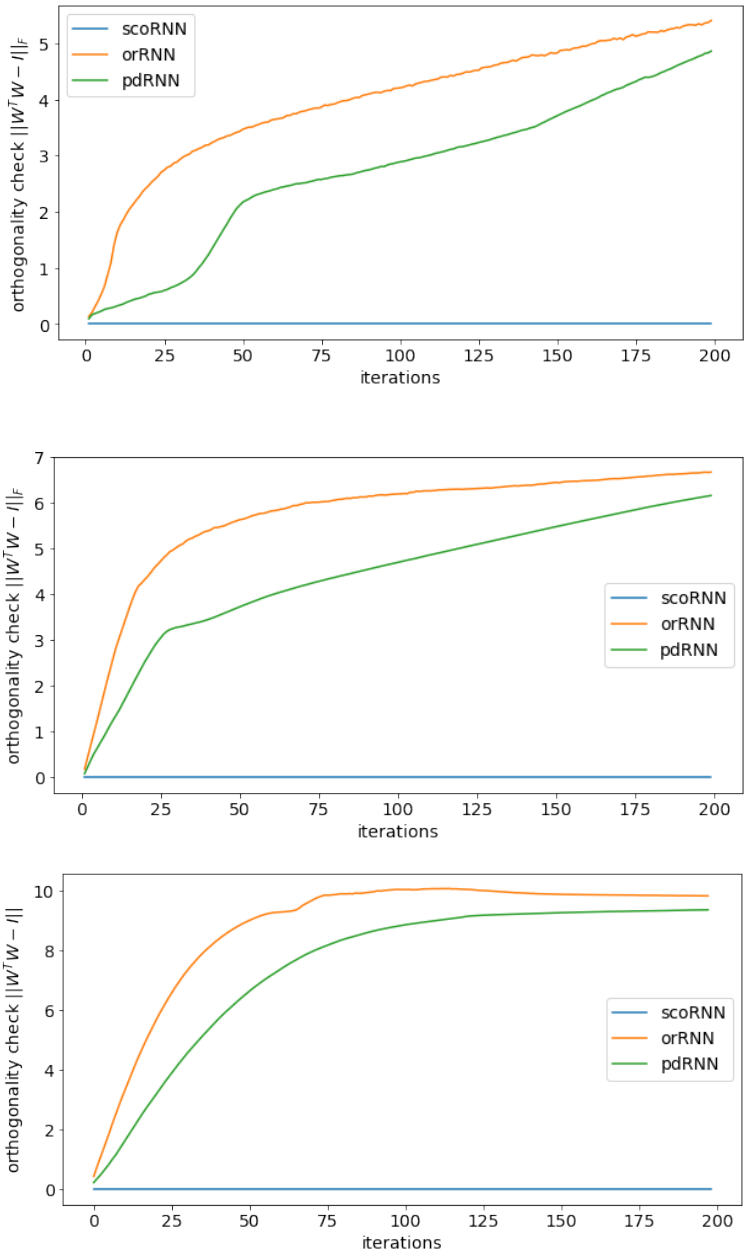
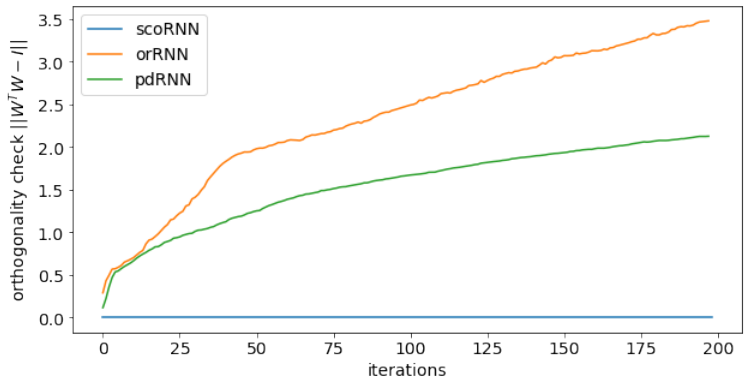
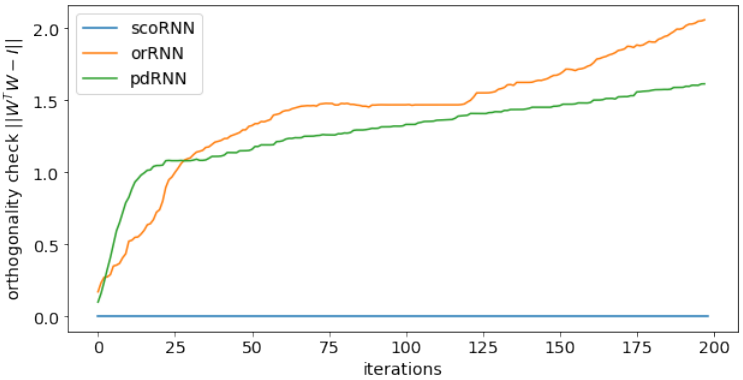


Figure 2.3. Orthogonality check of scoRNN, orRNN and pdRNN on the adding problem with sequence lengths of T=200(top), T=400(middle), and T=800(bottom).



(a)



(b)

Figure 2.4. Orthogonality check of scoRNN, orRNN and pdRNN on the copying problem with delay length (a) T=1000 and (b) T=2000.

2.3.4 Pseudospectrum visualization and orthogonality analysis

As mentioned in the previous sections, there is a large proportion of work has concentrated upon controlling the gain of the weight matrix W. This is done by ensuring that $\mathbf{W}$ is close to an orthogonal matrix. Such a factorization gives us a convenient way to bound the spectral norm. From the Keriss matrix theorem, we have the upper and lower bound of the norm of the power of the matrix as follows:

Theorem 5 *Define the Kreiss constant of a matrix* $\boldsymbol{W}$ *as follows:*

$$\mathcal{K}(\boldsymbol{W}) \equiv \sup_{\epsilon>0} \frac{\rho_\epsilon(\boldsymbol{W}) - 1}{\epsilon} = \sup_{|z|>1} (|z| > 1) \left\| (z - \boldsymbol{W})^{-1} \right\|$$

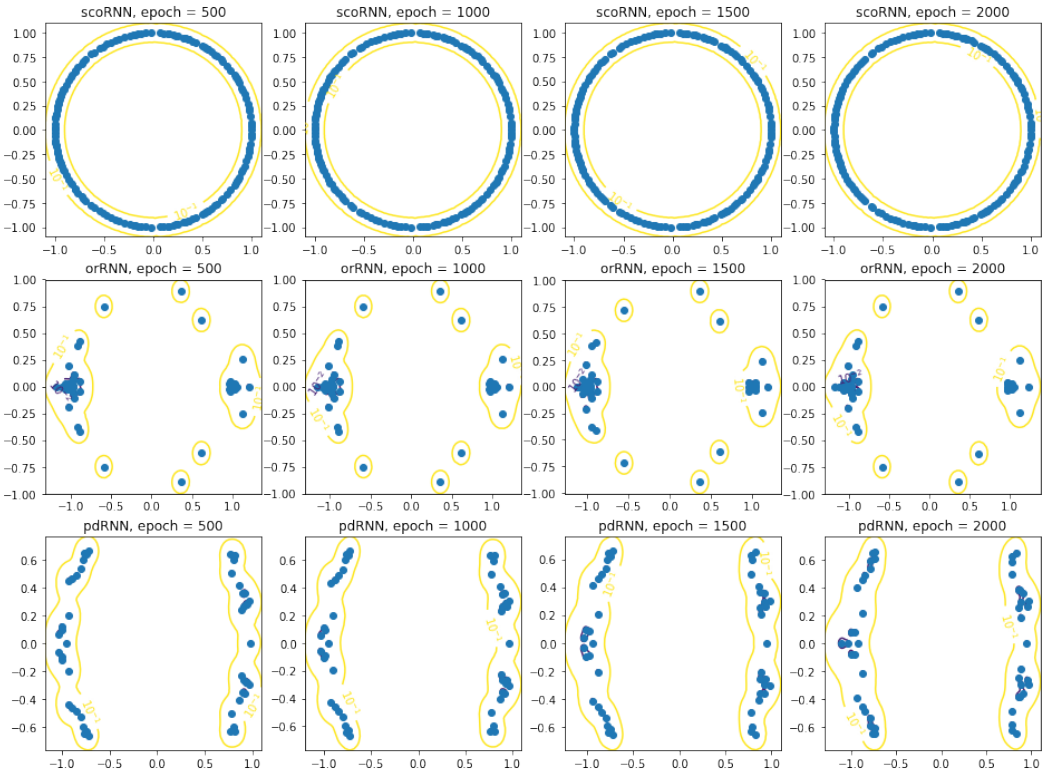


Figure 2.5. pseudospectrum of the adding task with sequence length 200 of the scoRNN, orRNN and pdRNN at iteration epochs of 500, 1000, 1500 and 2000.

The upper and lower bound of the $\left\| \boldsymbol{W}^k \right\|$ *can be given as follows:*

$$1 - \frac{kM}{\|(z - \boldsymbol{W})^{-1}\|} \leq \left\| \boldsymbol{W}^k \right\| < e(k+1)\mathcal{K}(\boldsymbol{W}), \tag{2.27}$$

where $M = \sup_{k \leq 0} \left\| \boldsymbol{W}^k \right\|$.

Both the upper bound and lower bound is controlled by the the norm of the resolvent $\|z - W\|^{-1}$ term. One way to measure and visualize the non-normality is by leveraging the concept of pseudospectrum which is the open subset of the complex plane bounded by the ϵ^{-1} level of the norm of the resolvent $\|z - W\|^{-1}$. We use it to track the evolvement of the weight matrix during the iterations and give the formal definition of the ϵ-pseudospectrum as follows [33, 34]:

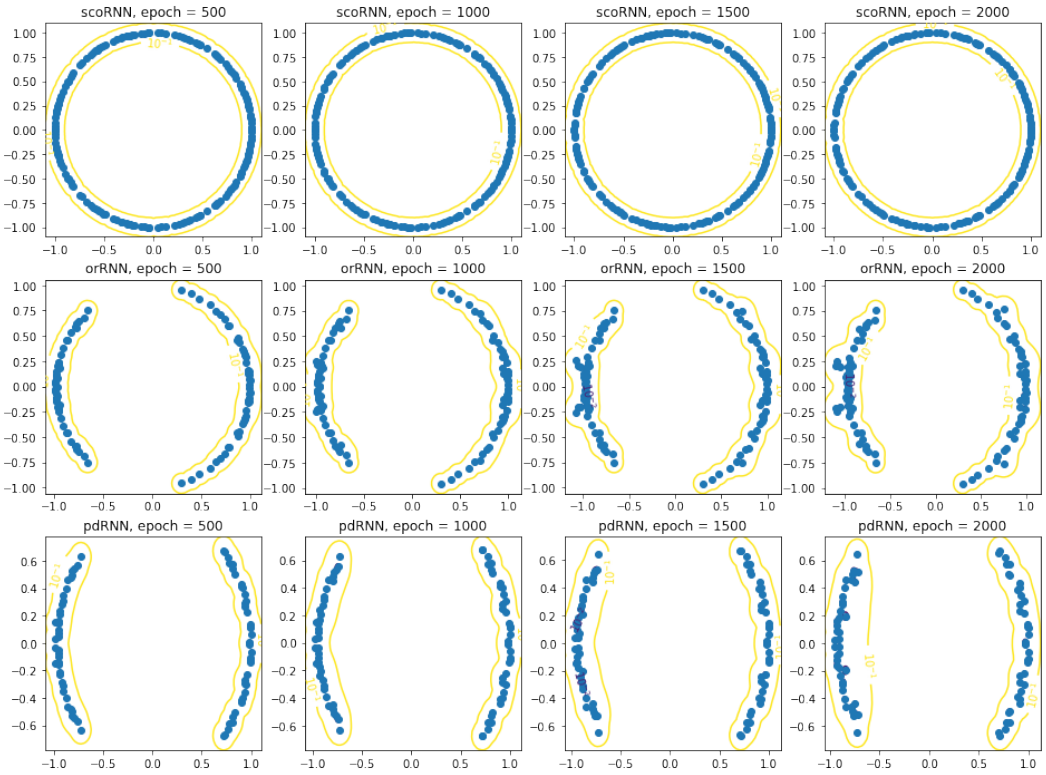


Figure 2.6. pseudospectrum of the copying task with sequence length 200 of the scoRNN, orRNN and pdRNN at iteration epochs of 500, 1000, 1500 and 2000.

Definition: Let $\mathbf{W}$ be an n-by-n matrix of complex numbers. For $\epsilon > 0$, The relative ϵ-pseudospectrum of $\mathbf{W}$ is defined as:

$$\sigma_\epsilon(\mathbf{W}) = \{z \in \mathbb{C} : \left\|z - W^{-1}\right\| \leq (\epsilon \left\|A\right\|)^{-1}\} \tag{2.28}$$

There are several open source tools like EigTool (MATLAB), pseudopy (Python) and mpseudo (Python) and we utilize the tool pseudopy [35] to draw the ϵ-pseudospectrum. We plot the pseudospectrum with different epochs to show the changes during the iteration. The iterations of the recurrent weight matrix is essentially to increase the number of power iterations of the weight matrix. As shown in Fig. 2.5 - Fig. 2.6, the process does not alter the resulting contours of pseudospectrum in a substantially. For the adding task, the scoRNN can keep the distribution of the weights uniformly

around the unit circle, which demonstrates that it keeps the best of the orthogonality among all the decomposition. The distribution of the orRNN concentrates around the range [-1, 0] and [1, 0] with a magnitude of 0.25 on the left side and 0.1 at the right side. The distribution of the pdRNN is in the middle of the previous two without holding the circle shape as good as the scoRNN. For the copying task, the scoRNN still keeps the best of the orthogonality, however the orRNN seems to approximate the circle shape more compared to the adding task case. The distribution of pdRNN still lies around the two sides in the real-imaginary plane. The distribution of the data and the hyperparameters of the RNN have impact on the orthogonality of the weights during the iteration.

2.4 Conclusions and Future Work

In this chapter, we propose a new recurrent weight decomposition method for the RNN, with the constraints that $\mathbf{W}^T\mathbf{W} = \mathbf{I}$. By leveraging the eigenvalue decomposition of the intermediate variable, a coordinate descent like back-propagation algorithm is designed for adjusting the weights during the training process. We have carried out experiments on the synthetic baseline dataset; (adding and copying task) as well as the real medical data for polyps detection. Our orRNN has better performance in terms of the cross-entropy loss on both of the two copying tasks than the scoRNN and LSTM. As for the adding tasks, our orRNN is better than scoRNN and LSTM for Adding 200 and keep approximate the same performance as Adding 400 and Adding 800 while using less parameters. For the real dataset of the polyps detection, both of the three RNNs can achieve very high accuracy in terms of two MFCC of the running vowels and orRNN has a slightly better advantage over the rest of the two with less number of parameters in the network.

During the experiment, we have found out that the orRNN and pdRNN cannot keep the orthogonality constraints as low as the scoRNN, which required the tuning of the hyperparameters for specific tasks, which may lead to large vibration at some situations. Up to now, we have not found out the critical reason and an automated method to control this phenomena. Furthermore, the problems related to the inherent of the structured RNN are addressed as the future research goals, which are listed as the followings:

- *Dynamics of the RNN*: model the dynamics of the RNN by introducing the dynamic model such as the differential equation and visualize the variation of the memory.
- *Optimization method*: find other available manifold retraction methods and optimization algorithms to balance the trade off between scaling matrix $\mathbf{D}$ and recurrent weight matrix

CHAPTER 3

Application of Structured RNN for Polyps Detection

3.1 Background

One of the challenges in modern healthcare industry is that doctors and physicians embrace massive amounts of data on patients from different sources, but they lack the enough tools and time to process the heterogeneous data. Intelligent early diagnosis and clinical decision support is necessary to both the patients and medical solution providers. Those data, ranging from images, time series from different sensors and electronic health records, contain abundant information to be explored and exploited.

Time-series data, as one of the long existing and important sources, plays a crucial role for medical diagnosis. Classical model for time-series modeling is mainly based on statistical analysis, state-space description, and dynamical causality. From the statistical point of view, a Cox proportional hazard regression model for predicting stroke and heart disease was proposed [36]. The milestone work of Rabiner summaries the development of hidden Markov Model, which embedded the observation with hidden layer and used the transformation in the hidden states for inference [37]. A review of Wiener-Akaike-Granger-Schweder (WAGS) influence, state-space method, and dynamic causality method for fMRI data is discussed [38]. Although all these three kinds of methods make a huge progress on some specific tasks, they rely on lots of hand-crafted details and assumptions, which may not always be satisfied in the real applications.

In this chapter, we mainly focus on the diagnosis based on acoustic data samples. Similar to the general time-series setting, traditional approaches for symptom detection from the acoustic samples mainly focus on physical modeling. The mechanism of the voices production control is investigated by leveraging three-dimensional continuum model of phonation and glottal pressure analysis [39]. Moreover, some similar model for cause-effect discussion on the relation between vocal fold physiology and voice production is provided [40]. These kind of physical models are able to provide an interpretable model in some extent, however, the man-made model may miss hidden details and also it is not ubiquitous for all situations, i.e., the model is not extensible and flexible.

Therefore, the research direction is tuned to resort to model-free methods such as machine learning methods. The polyps detection problem is formulated as a classification problem and it is solved by the fuzzy logic algorithm [41, 42], however the method is not so accurate due to the coarse membership levels. For the real application of machine learning in symptom identification based on speech signal, an end-to-end machine learning framework is adopted for classification of the Parkinson's disease from the acoustic samples [43]. 132 hand-crafted dysphasia features extracted from the standard 44.1 KHz acoustic samples are employed on the support vector machine (SVM) and random forest (RF) algorithm to test the classification performance.

Compared to the huge amount of labor work on data wrangling and feature engineering, a deep sequential modeling with time-frequency transformation of the acoustic data for polyps detection is proposed [44, 45]. Besides trying only single modality signal, a multimodal formulation by fusing the speech, gait and handwriting for Parkinson's detection is achieved via deep learning [46]. End-to-end deep learning model for voice conversion, speech activity detection and parameter esti-

mation in physics-based sound synthesis are proposed to verify the feasibility and efficiency compared to traditional model [47, 48, 49]. Furthermore, an RNN autoencoder framework is designed to fuse the brain activity signal and vocal vibration to synthesize the voice signal to help dysphasia patients [50].

The OCRNN model is deployed to tackle this problem in this chapter and compared with the traditional RNN models. The highlights of the work are as follows:

- The real world data is preprocessed by time frequency expansion using signal processing based ideas to get the inherent features
- Two OCRNN introduced in last chapter are adopted for training on the row-wise of the time frequency feature map, so that the formulation can be transformed into a binary classification problem
- Demonstration of the performance in terms of the competitive accuracy while saving the storage of the network and computation time.

3.2 Data and Preprocessing

One of the typical symptoms among adults is the vocal cord polyps, which is the result of an acute injury (such as yielding and shouting really loud) or several other causes (such as cigarette smoke). The original diagnosis of this problem is from the pathological viewpoint, and it is highly depended on the expert knowledge, which is not suited to the urgent requirement of self-diagnosis currently. In principle, sounds generated by human beings are filtered by the structure of the vocal tract which comprises of parts like tongue, teeth, etc., which is nonlinear processing comprising of both physical and aerodynamic effects. This kind of structure leads to what kind of sound will come out. The recording of the raw voice signal for automatic diagnosis of the problem is preprocessed by the speech analysis. We collect the data that are about the discrete vocal samples of recording twelve respondents, in which four of

them have throat polyps and the rest are healthy (no throat polyps). In the study, the vowels /a:/ and /i:/ are collected and each of them lasts for approximately 10 seconds.

The polyps identification database comprises of 12 subjects (8 healthy controls and 4 with symptoms). All of the subjects are within the age range between 21-50. The database includes three or four sustained vowel /a:/ and vowel /i:/ from each speaker, recorded at a suitable frequency and amplitude. The vowels /a:/ and /i:/ are collected and each of them lasts for approximate 10 seconds. For convenience, we define four cases: "abnormal vowel /a:/", "abnormal vowel /i:/", "normal vowel /a:/" and "normal vowel /i:/". The samples of vowels within the same scenarios are snipped off a longer vowel sample to choose the stable and continuous part (without long and obvious halting and long waiting at the initial state) of the whole sample for processing. The sampling rate of the vocal data is 192000 and bits per sample is 32. The details of the vocal database are summarized in Table 3.1.

Table 3.1. Vocal dataset summary

Parameters	Values
No. of patients	12(4 with polyps, 8 without polyps)
Age range	21-50
sample rate	192000
samples range	1157200 - 1344000
bits per samples	32
channels	1
normalized data	FALSE

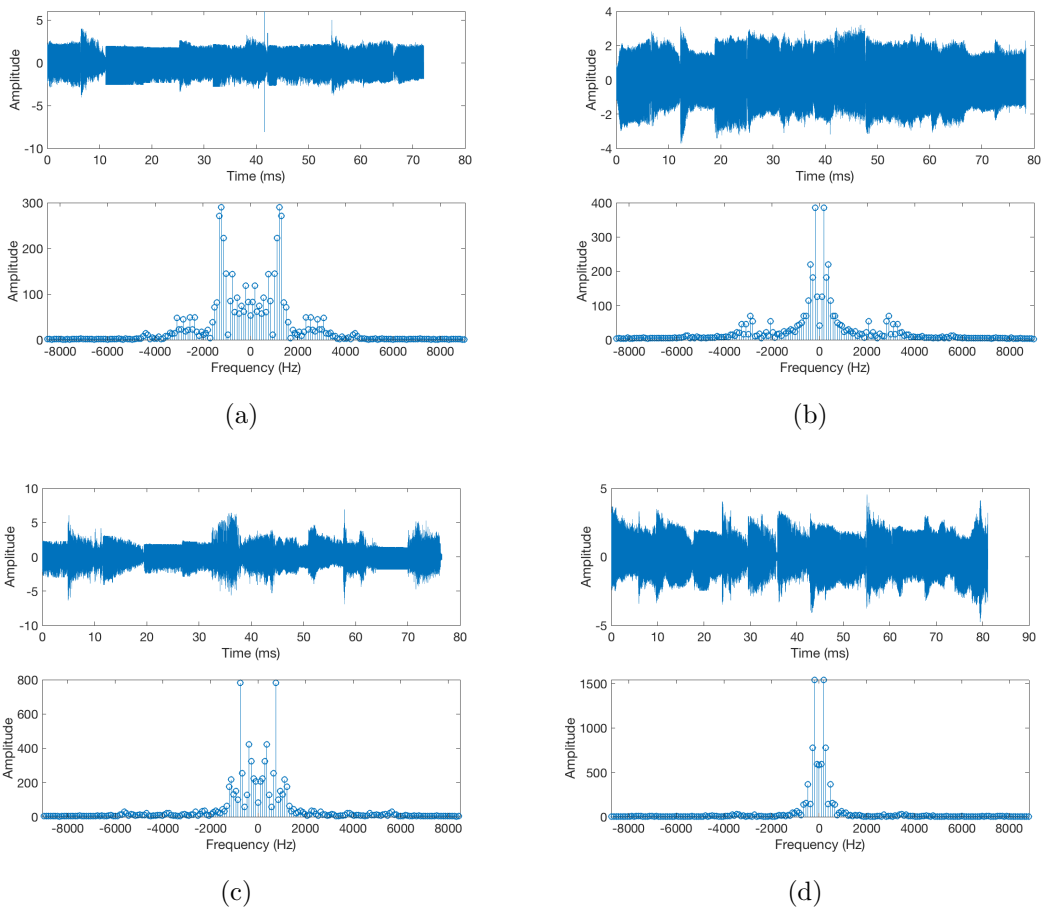


Figure 3.1. Four cases of the running speech ("abnormal vowel /a:/", "abnormal vowel /i:/", "normal vowel /a:/" and "normal vowel /i:/") in time domain and frequency domain.

The time and frequency representation of the four cases is demonstrated in Fig. 3.1. The signal depicted in the time-domain is non-stationary with several abruptly oscillation. While it is hard to clearly distinguish in the time domain, the distribution in the frequency domain shows some difference between normal and abnormal cases. The abnormal running vowels' frequency components concentrate more within [-2000Hz, 2000Hz], while the normal's expand further to approach 4000Hz. Therefore, resorting to the classical time-frequency expansion, the mel-frequency cepstral coefficient (MFCC) is adopted to extend the information in one-dimensional raw vocal

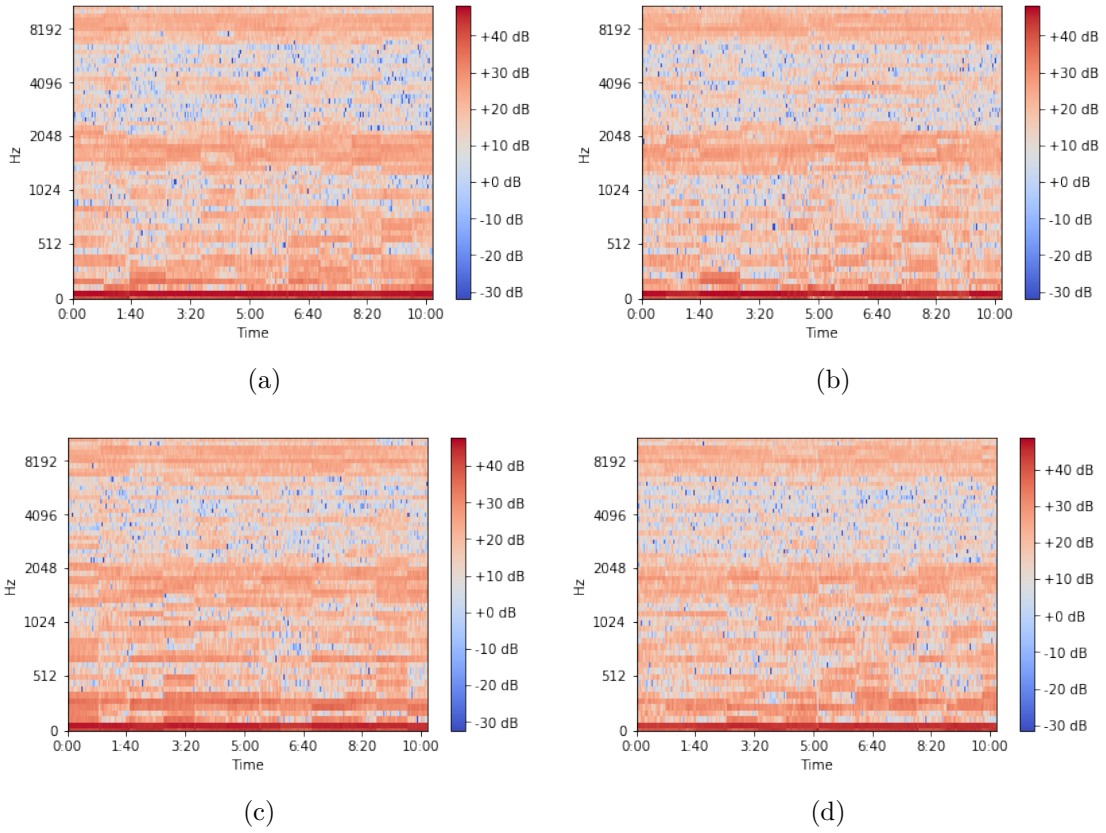


Figure 3.2. MFCC features of four cases of the running speech ("abnormal vowel /a:/", "abnormal vowel /i:/", "normal vowel /a:/" and "normal vowel /i:/").

signal to two-dimensional time-frequency plane. For convenience, we briefly describe the data processing of the MFCC here for reference and the main steps are listed as follows [51, 52]:

1. Remove the DC component and normalize the magnitude of the signal.
2. Leverage a one-lag finite impulse response filter to preemphasize the signal, usually the function is given as: $y[n] = x[n] - 0.97 * x[n-1]$ (optional)
3. Calculate the squared magnitude of the STFT with a given frame length and window length (we choose the Hamming window)

4. Compute the inner product between the triangular basis functions and the signal, and transform it into the Mel-scale in frequency domain according to: $f' = 2595\log_1 0\left(1 + \frac{f}{700}\right)$
5. Calculate the logarithm of the Mel frequency coefficients $10\log_1 0(\cdot)$.
6. Take the cosine transform of the log-scale coefficients according to the following:

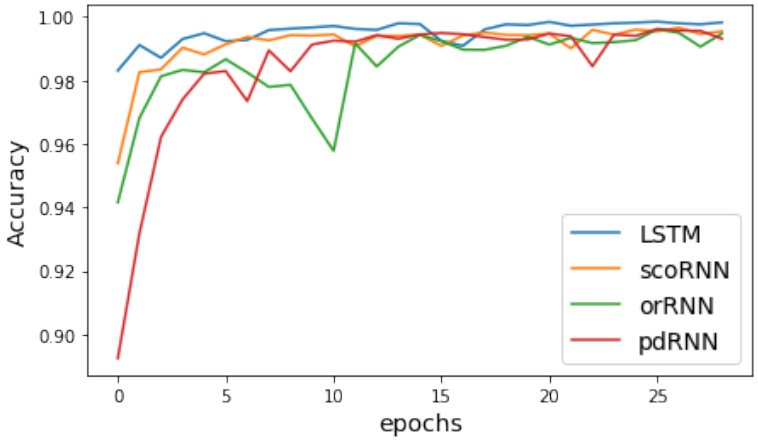
$$c[m] = \sum_{k=0}^{K-1} \cos\left(\frac{\pi m(k-\frac{1}{2})}{K}\right) s[k] \quad k = 0, 1, \ldots, K$$

where $s[k]$ is the log scale coefficients.

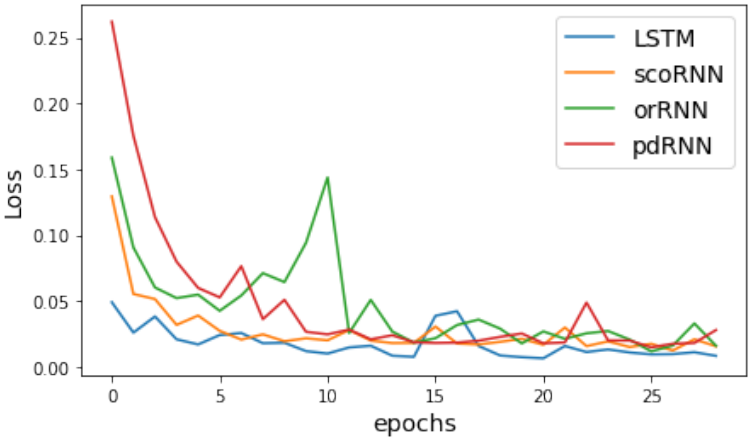
3.3 Experiment Setting and Results Analysis

We have collected multiple running audio signal sequences for each subject and we concatenate the same type together to build two separate running vowels /a:/ and /i:/. We use the python librosa package [53] to generate the MFCC features for each running vowels. We setup the sampling frequency as 192000 sample indexes per and choose 60 MFCC components as the total number of sub-components. We convert an amplitude spectrogram to dB-scaled spectrogram and the visualization results of the four types of MFCC are shown in Fig. 3.2. Observing the four figures, most of the strong components lie on the lower frequency part and it is very hard to distinguish at first glance. The abnormal parts ((b) and (d)) of vowels /a:/ and /i:/ seem to expand more than the normal part ((a) and (c)) along the frequency.

We extract each row of the two-dimensional time-frequency expansion, which is represented as the one-dimensional signal of different frequency component as the input to the RNN. We utilize the following RNN settings to compare the performance of four RNNs. The dimension of the weight matrix orRNN and pdRNN are set as 50 while the dimension of the LSTM and scoRNN is set up as 100 and 80 for consideration of the comparison of the total number of network parameters. The total number of



(a)



(b)

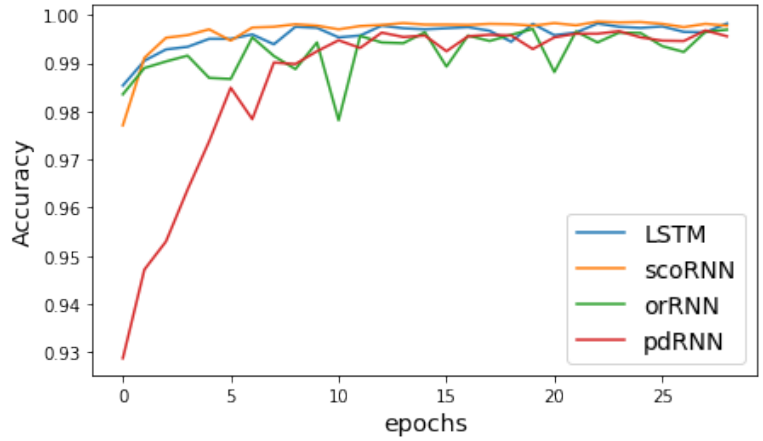
Figure 3.3. Accuracy and loss comparison of four RNNs with MFCC features of running speech "vowel /a:/".

training epochs is 30 and the batch size is setup as 40. We use the Adam optimizer for the hidden weights training and the learning rate is set as 10^{-4} and the input-output weights are trained via the RMSprop optimizer with the learning rate of 10^{-3}. We set the number of negative ones as $n/10$ for the demonstrated results here, where n is the dimension of the recurrent weight matrix. The training-testing ratio for each experiment is set as 4:1. We choose the binary cross-entropy as the loss function because of the classification task setting.

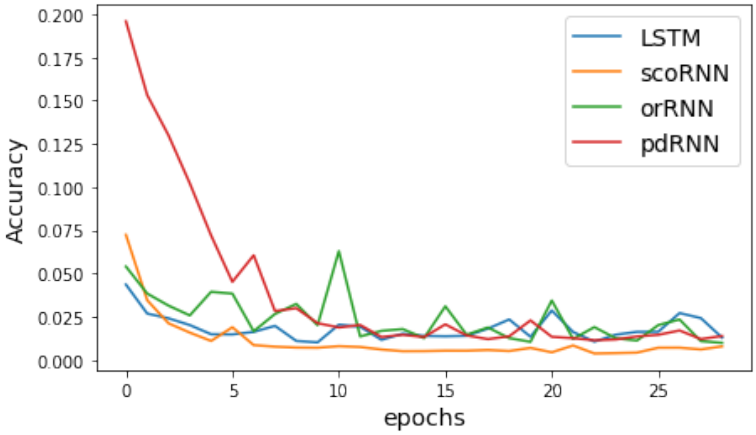
Table 3.2. Comparison of the test accuracy and test loss of the final iteration of three RNNs on the MFCC vocal detection

Performance Index	MFCC vowel /a:/	MFCC vowel /i:/
LSTM test loss	0.02832	0.00997
scoRNN test loss	0.0385	0.00335
orRNN test loss	**0.00789**	**0.00156**
LSTM test accuracy	99.97%	99.91%
scoRNN test accuracy	98.82%	99.94%
orRNN test accuracy	**99.24**%	**99.83**%

As shown in Fig. 3.3 and 3.4, both of the four RNNs can achieve the accuracy above 99% while keep the loss under 0.05. The three orthogonal constrained RNNs seems to be more fluctuating than the LSTM during the iteration. The orRNN and pdRNN can achieve slightly better performance over the LSTM and scoRNN with less total number of parameters (**total number of parameters of orRNN and pdRNN are 10202 while LSTM and scoRNN are 41002 and 13122 respectively**). The comparison demonstrates that orthogonal constrained RNNs can achieve comparative performance against the regular LSTM while using much less computation load, which has the potential to deploy on broader range of the computation platforms for requirements like internet of things and edge computing. The detailed summary of the detection accuracy and cross entropy loss last iteration in terms of test accuracy and cross entropy loss are shown in Table 3.2. Furthermore, we provide the results of the confusion matrix for two vowels against four RNNs are given in fig 3.5 for the demonstration of the anomaly detection for sensitivity and precision check.



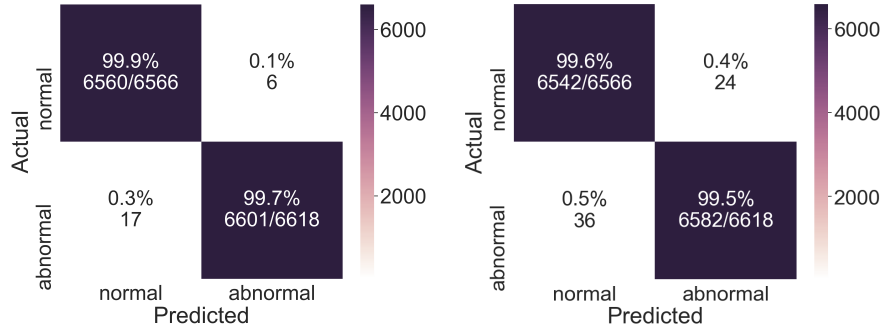
(a)



(b)

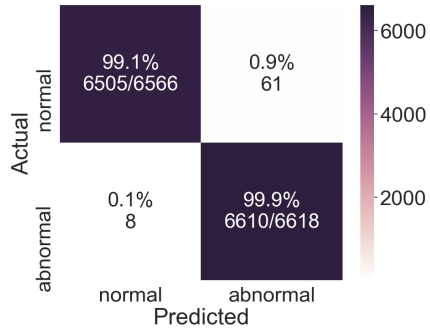
Figure 3.4. Accuracy and loss comparison of four RNNs with MFCC features of running speech "vowel /i:/".

To summary, we utilize these OCRNN frameworks to solve the polyp detection problem. We collect the vocal polyps running vowel data with two vowels for 12 subjects and formulate the problem as the binary classification. The MFCC sets up as the preprocessing step to generate the time-fequency feature map for the nonlinear time-series signal. Experiments results demonstrate that all four RNNs can achieve comparatively promising results while the OCRNNs (pdRNN and orRNN) can still save more network overload and computation cost.

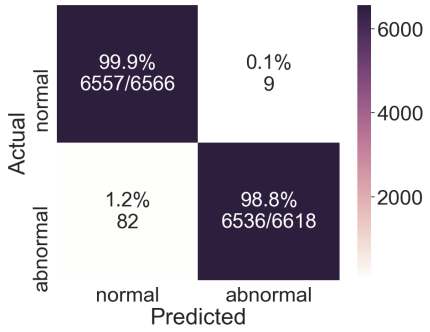


(a) LSTM

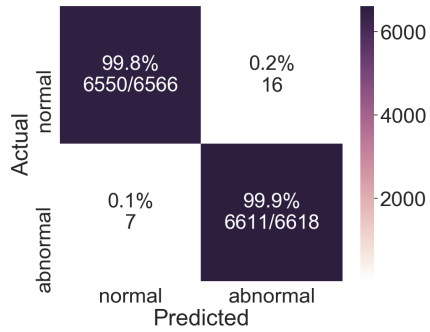
(b) scoRNN



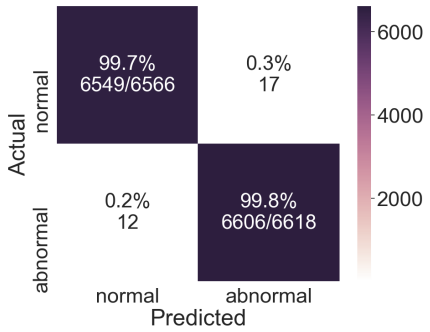
(c) orRNN



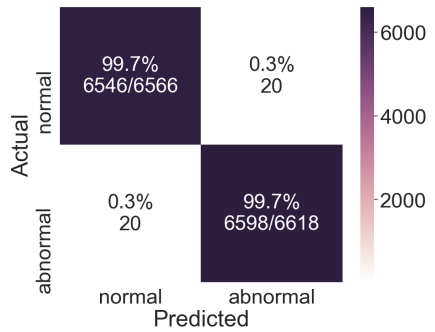
(d) pdRNN



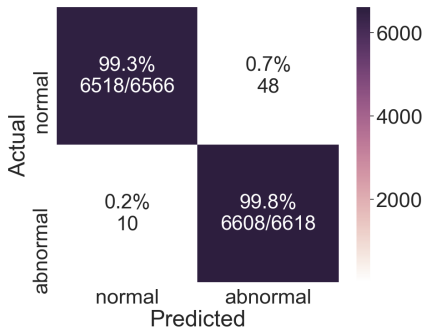
(e) LSTM



(f) scoRNN



(g) orRNN



(h) pdRNN

Figure 3.5. confusion matrix visualization of the four types of RNNs with vowel /a:/ (a)-(d) and vowel /i:/ (e)-(h).

CHAPTER 4

EEG Based Sleep Stage Scoring Using 1dCNN-OCRNN Mixed Model

4.1 Introduction

Electroencephalograph (EEG) is a complex signal that contains information from different frequency bands and different channels (multiple probes for different brain functional areas). This multimodality signal attracts researchers to find approaches to disclose the underlying information so that we can leverage the information to solve problems such as epilepsy, sleep disorders, brain–computer interfacing, and cognitive monitoring [54]. The challenging task of making classification based on EEG signals lies on the fact that EEG is a non-stationary and low signal-to-noise ratio (SNR) signal, so that some single hand-crafted signal processing approach meets the bottleneck [55]. EEG has the high temporal resolution due to the high-speed propagation of the electric field but EEG has a low spatial resolution so that the signals are highly correlated. This increases the difficulty to decompose and expand the details of the signal.

Thanks to the development of the modern computing power to trigger the potential of data science and artificial intelligence, researchers seek to handle these problems with more powerful and automatic toolboxes to understand the EEG signal. For instance, the component analysis such as PCA, ICA and neighborhood component analysis [56, 57, 58] serve as the pre-processing steps for the downstream steps like SVM, k-means [59, 60, 61]. However, this kind of framework belongs to the traditional hand-crafted machine learning algorithms which highly rely on the domain knowledge and they may be subject to the limitation of the applicability and

flexibility. Deep learning, as one categories of the machine learning families, provide the hierarchical representations of input data complicate non-linear activation and flexible connections. Since the proven performance of the deep learning in areas like images, videos, acoustic signal, natural language processing, we witness the rise of the interest for deployment of the deep learning model in EEG understanding as well.

Among the deep learning models, recurrent neural network, as a naturally time-series based deep learning model, mostly will serve as the first-try model for EEG related tasks. For instance, a special kind of RNN named echo state network was utilized to classify the REM Behavior Disorder (RBD) from 118 subjects (with healthy control) based on the collected EEG signals [62]. A more complex multi-task RNN framework was proposed for motion intention recognition based on features extracted from segregated EEG signals [63]. An end-to-end RNN learning framework was designed to learn the EEG signal after removal of the artifact for the movement related cortical potentials (MRCP) in order to recognize lane change decision making [64]. More RNN based on system-level application demonstrate the suitability of RNN in broader applications such as autonomous whole-arm exoskeleton control [65] and decoding of motor imagery movements [66].

However, according to the statistics by [67], around 40% of the studies used convolutional neural networks (CNNs), while 13% used RNNs. The previous studies trained their neural networks more on preprocessed EEG signals. The benefit of the mixed model is that it can leverage the advantages of each submodule of the neural network while also minimize the incompatibility. Some previous studies have demonstrated the mixed model for raw time-series data in other applications such as soil moisture retrieval [41, 68] and signal detection [69, 70], especially the mixed model or different neural network, such as CNN-RNN mixed framework [71]. In terms of the EEG-related mixed model, most of the models focus on the end-to-

end downstream pipeline framework with hand-crafted pre-processing techniques like short-time Fourier transform (STFT), mel-frequency cepstral coefficient (MFCC) and low-pass filter (LPF). These models still will be constrained by physical limitations due to the preprocessing steps so that we seek to more automatic and flexible framework. A one dimensional CNN plus bidirectional long short term memory (LSTM) model is proposed as the fully mixed deep learning framework [72], but we do not find too much related examples like this.

In this paper, we propose 1dCNN-OCRNN model, an one dimensional CNN followed by the orthogonal constrained RNN for the EEG signal based sleep stage scoring task. The highlights of our work are:

- We use the 1dCNN to generate the multi channel feature maps to achieve the similar function as the time-frequency function, but we reduced the complexity of model by adjusting the output and eliminate repeated convolution layer with fewer channels.
- We design our own orthogonal constraints based RNNs with its related coordinate-descent like iteration algorithm for training. These type of RNNs can mitigate the long-existing vanishing and exploding problems by leveraging the self-adjusted weights. Therefore, important information from the EEG multi-channel feature maps are stored in the "memory" of the RNN.
- We demonstrate the results use the real-world sleep staging classification datasets based on EEG signals. We provide solid comparison of the performance of the three OCRNNs and the standard LSTM. Compared to the early results provided in [72], we achieve competitive slightly better results using approximate 1/6 total number of parameters and 1/3 of training epochs. Therefore, by saving the computation cost in both spatial and temporal, our model has the potential to be broadly deployed in real scenario.

The rest of the paper is organized as follows. In section II, the preliminaries of the OCRNNs and its iterative training algorithms are introduced. In section III, the 1dCNN-OCRNN model is introduced and we compared different combinations of the mixture model. In section IV, the experiment of the EEG sleep stage analysis is introduced and the solid performance comparison is analyzed and discussed. Finally, we conclude in section V.

4.2 Frameworks of the 1dCNN-OCRNN mixture model

Inspired by the work from [71, 73], we consider to utilize the 1dCNN to achieve the similar function of the time-frequency expansion. Unlike traditional signal-processing based methods such as short time frequency transform (STFT), mel-frequency cepstral coefficient (MFCC) and low-pass filter (LPF) to distinguish difference frequency components, we do not need to calculate or separate specific frequency component by hand but resort to the 1dCNN to automatically learn and iterate the multi-channel feature maps in order to achieve the similar "time-frequency expansion". Specifically, we consider the following 1dCNN-OCRNN mixture model. The model contains the two parallel CNN modules: fine 1dCNN and coarse 1dCNN. The fine 1dCNN contains smaller strides and filter kernel size in the first convolution layer, followed by several convolution layers and pooling layers. The coarse 1dCNN contains bigger strides and filter kernel in the first convolution layer, followed by several convolution layers and pooling layers with different parameter settings compared to the fine 1dCNN. Finally we utilize the concatenate dense layer to combine the find features and coarse features to represent the broad range of the EEG multi-channel feature maps. The details of the parameters are listed in in Fig. 4.1.

Unlike the work proposed in [73], for our mixed model, we do not need the staged two layers of bi-directional LSTM modules to increase the complexity of the module.

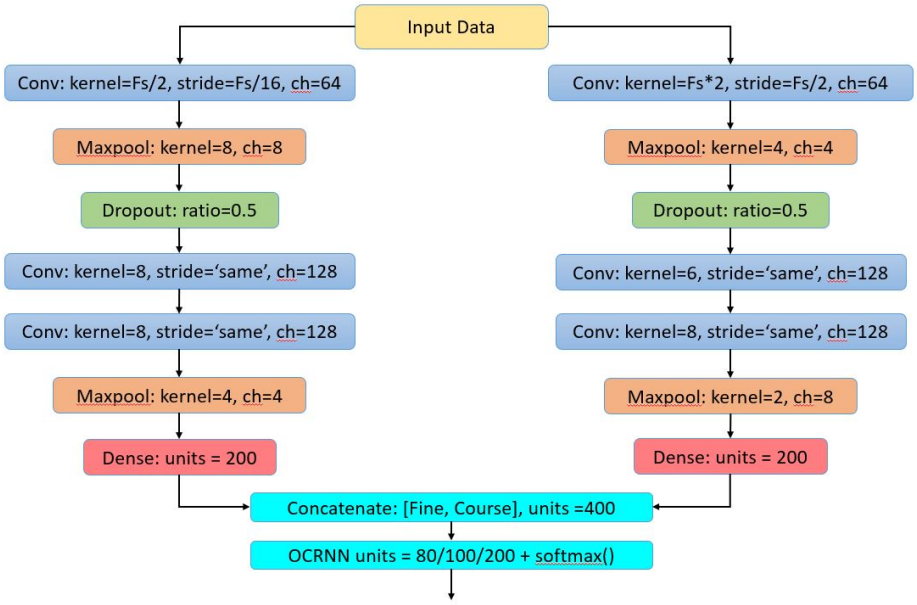


Figure 4.1. Framework of the 1dCNN-OCRNN mixed model.

we simplify our 1dCNN part compared to the model in [73] to further reduce the complexity of our mixture model. We do not need extra dropout layer and parameter tuning to prevent the vanishing gradients and exploding problem due to the benefits of orthogonal constraints from the OCRNN. We choose only one layer of the OCRNN to accept the input from the above output of the previous dense layer. The OCRNN proposed in Section can serve as the plug-and-play module to fit in output of the 1dRNN. We compare the performance of the vanilla LSTM and three other OCRNNs in our experiment.

The total number of the parameters of the four mixed networks are demonstrated in table 4.1. We have tried several different settings and the the weight matrix dimensions we demonstrate here are based on consideration of the approximate the similar number of total parameters as well as the stable of the mixed of network (no huge oscillation during the iteration). As we can see, compared with the setting shown in [72], we only need around 1/6 of the total number of parameters in

mixtured model	weight matrix dimension	total number of parameters
1dCNN-LSTM	120	562445
1dCNN-scoRNN	100	534245
1dCNN-orRNN	80	529445
1dCNN-pdRNN	80	529445

Table 4.1. Comparison of total number of parameters in four mixed model

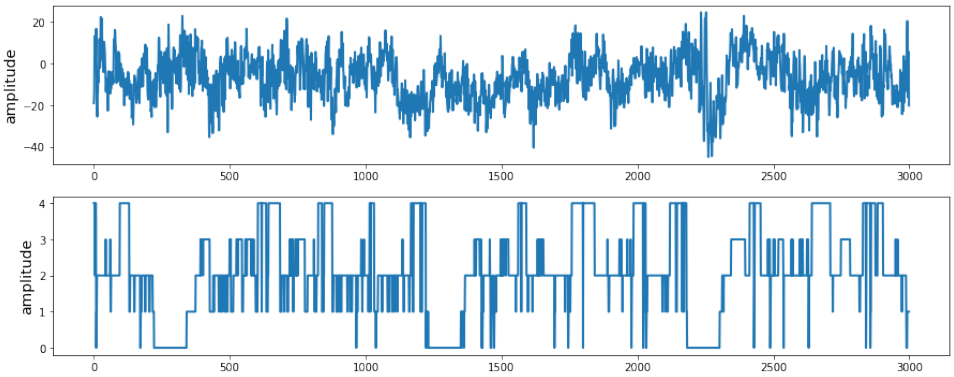


Figure 4.2. Different recordings of the unnormalized raw EEG signal (data) and sleep stage classes (label) that change over time.

the mixed model, which saves the computation cost in the spatial domain. Furthermore, the weights that are required to run the back-propagation will be much less as well, so that the time complexity will be reduced. This demonstrates the potential of deploying our model on some edge computing based platforms as well as approximate real-time diagnosis.

4.3 Experiment and Performance Analysis

To validate the effectiveness of the decomposition and iteration algorithm of OCRNN, as well as the effectiveness of the mixed model, we choose the DeepSleep dataset [72, 73] which is essentially the transformation of the partial of the

sleepEDF dataset [74, 75]. The original sleepEDF database contains 197 whole-night polysomnographic sleep recordings with EEG, EOG, chin-EMG, and event markers. Part of the records also include respiration and body temperature. The DeepSleep dataset takes data of 20 subjects, where 19 of them have 2 full nights of sleep. The dataset is transformed and compressed into the numpy .npz format, where each epoch contains the 30 seconds record of EEG with a label form $\{"W, "N2", "N3", "N4", "REM"\}$. The aim of the research study is to correctly recognize the sleep stage so that will leads to better understanding of the brain activity. Examples of the raw EEG signals and their corresponding sleep stage class labels are shown in fig 4.2.

We build the 1dCNN-OCRNN model to solve the problem. Typically, we have tested the cases of 1dCNN-LSTM, 1dCNN-scoRNN, 1dCNN-orRNN, 1dCNN-pdRNN. The detail of the network setting is demonstrated in table 4.1. The setting of the dimension is to consider the case that the 4 mixed model can achieve competitive results in the final epochs. For the four cases tested in our study, the total number of parameters are less than the original study proposed in the [73], with the potential to achieve competitive results use smaller 1dCNN and total number of parameters with further hyper tuning. We choose the Rmsprop optimizer with learning rate of 10^{-3} for the 1dCNN part and Rmsprop optimizer with learning rate of 10^{-4} for the OCRNN part. The number of the -1s in the initialization of the scaling matrix $\mathbf{D}$ is set as the $n/10$, where n is the dimension of the recurrent weight matrix. The batch size is 32 and the total number of training epochs are 50. Unlike the model proposed in [73], we do not rely on further regularization and changing of the learning rate by leveraging the robustness of the OCRNN due to its consideration of the orthogonal constraints.

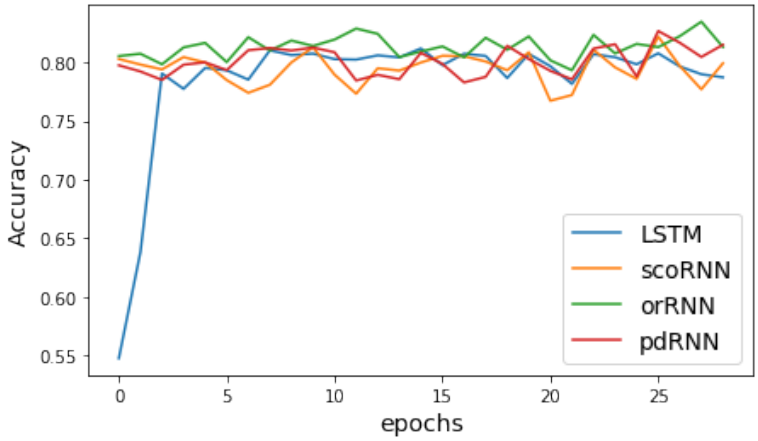


Figure 4.3. Comparison of the testing accuracy of the four 1dCNN-OCRNN models.

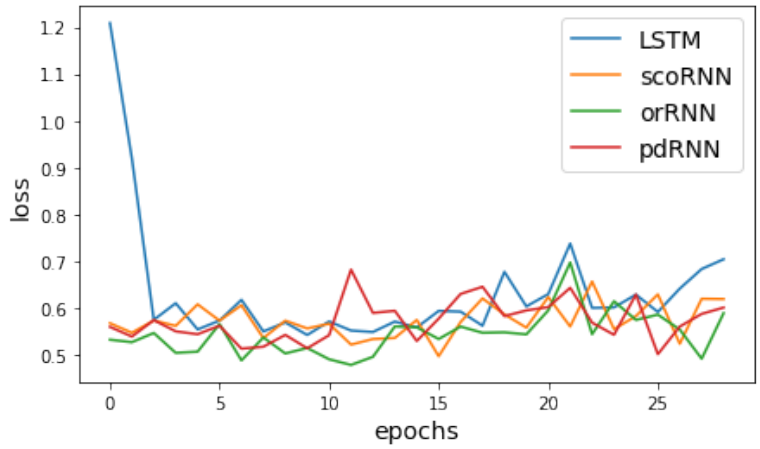


Figure 4.4. Comparison of the testing loss of the four 1dCNN-OCRNN models.

The results of the training and testing accuracy, training loss and testing loss are demonstrated in table, fig. 4.3 and fig. 4.4. As we can see from the results, all the OCRNNs run comparatively accuracy and pdRNN is slightly better than the rest of the three, while LSTM is not as efficient as the OCRNN series. OCRNNs can converge to the minimum much faster than the LSTM (approximate 15-20 epochs) so that we do not need train so many epochs in order to guarantee the accuracy of bottom line. Compare to the results demonstrate in [73] (please refer to

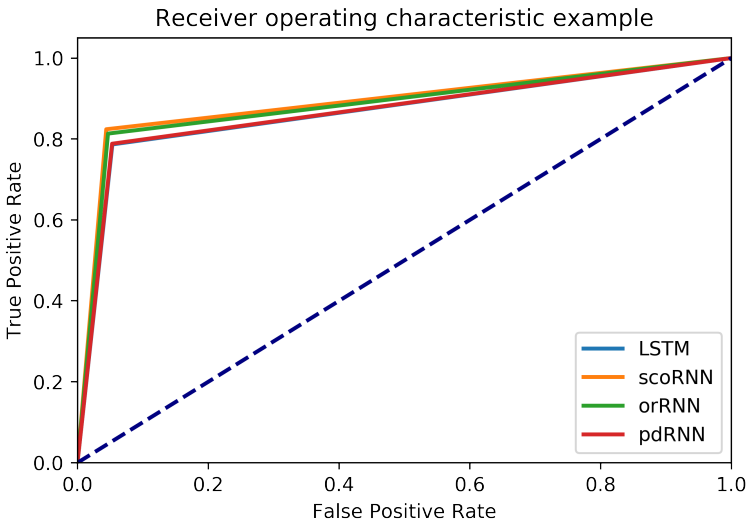


Figure 4.5. Comparison of the ROC curve and AUC score of the four 1dCNN-OCRNN models.

https://www.kaggle.com/phhasian0710/deepsleepnet-201708 for more details), they achieve the similar result with around 3 million total number of parameters and 100 epochs of training (first time reach over 80% at epoch 55) and our pdRNN is slightly better with much less training overload. The trend comparison of the ROC-AUC analysis of the four mixed model is demonstrated in fig. 4.5.

We provides a detailed look at the performance on each sleep stages by leveraging the confusion matrix visualization as well as the calculation of the per class F1-score. As shown in fig. 4.6, we compare the f confusion matrix of five classes "W1", "N2", "N3", "N4", "NRM" among four different mixed models. The imbalance precision performance shown in the second class "N2", while the rest achieves the precision as high as over 90% for class "NRM" in 1dCNN-orRNN model. This phenomenon matches the result demonstrate in [72], which may caused by the imbalanced data problem that requires extra pre-processing. The detailed numerical results for all the performance analysis are given in table4.3.

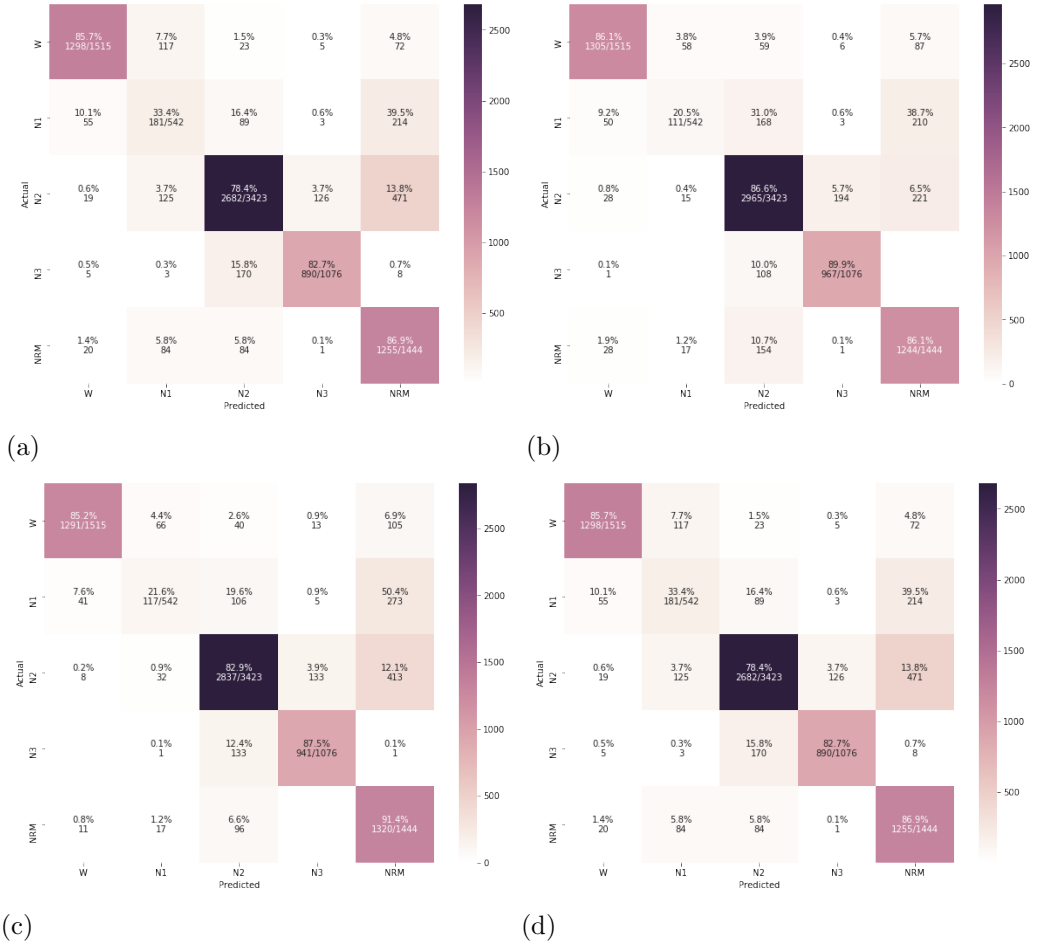


Figure 4.6. Comparison of confusion matrix of five classes "W1", "N2", "N3", "N4", "NRM" among four different mixed models: (a) 1dCNN-LSTM (b) 1dCNN-scoRNN (c) 1dCNN-orRNN (d) 1dCNN-pdRNN.

4.4 Conclusion

In this paper, we proposed a 1dCNN-OCRNN mixed framework to solve the sleep stage scoring problem based on the EEG data. Our model leverage the benefits of 1dCNN by utilizing it to expand the data-driven multi-channel feature maps. It achieves similar function as the traditional time-frequency expansion but it does not count on any hand-crafted feature engineering. It provides two levels of information (the fine and coarse) in the feature maps and they are merged as the

mixtured model	testing accuracy	testing loss
1dCNN-LSTM	81.5%	0.6006
1dCNN-scoRNN	80.23%	0.7039
1dCNN-orRNN	82.40%	0.6189
1dCNN-pdRNN	81.32%	0.5886

Table 4.2. Numerical comparison of testing accuracy and testing losss of four mixed model

mixtured model	'W'	'N2'	'N3'	'N4'	'NRM'
1dCNN-LSTM	86.84%	23.79%	83.00%	81.69%	72.24%
1dCNN-scoRNN	89.17%	29.88%	86.23%	86.07%	77.60%
1dCNN-orRNN	90.09%	30.19%	85.52%	86.81%	74.24%
1dCNN-pdRNN	89.15%	34.41%	82.89%	84.72%	72.46%

Table 4.3. Numerical comparison of performance metrics of four mixed model

input to the OCRNN. As for the OCRNN, we leverage the benefits of empirically put the orthogonal constraints on the recurrent weight matrix and adopt the manifold retraction method to design the related algorithms. We design two OCRNNs (orRNN and pdRNN) and construct three different types of the 1dCNN-OCRNN frameworks (plus the scoRNN proposed in [27]. We demonstrate our results on the real world deepsleep dataset and compare the performance four 1dCNN-RNN framework (1dCNN-LSTM, 1dCNN-scoRNN, 1dCNN-orRNN and 1dCNN-pdRNN). The experiment results demonstrate that by leveraging the power of the mixed model, the 1dCNN-pdRNN achieve comparatively better results in accuracy categorical F1-score. Moreover, the time complexity for training (1/3 of traing epochs vs. previous

benchmark) and spatial complexity (1/6 of total number of network parameters vs previous benchmark) manifests the potential of the proposed model.

Future research may focus on the following two folds. Firstly, the empirical orthogonal constraints play a crucial role in accelerating the convergence while stabilize the mixed model. However, when handling the real world data like EEG, it still meets the challenges that may caused by the abrupt high oscillation on the time domain, which causes the vibration of the accuracy shown in fig. 4.3. A more robust RNN model based on structured based constraints is a potential direction. Second, we can move to complex multiple channel based EEG signal related tasks. In order to handle the problem of the imbalanced performance (for example precision among different stages), a better deployment of sensors with high resolution and better way of fusing the information from multiple sensors by customizing an end-to-end mixed model is one of the potential approaches.

CHAPTER 5

Graph Neural Network Based Anomaly Detection

5.1 Introduction

With the rapid growth of the big data and internet of things (IoT), more sensors are deployed onsite for process control and monitoring the health of the industrial pipeline. Anomaly detection plays a critical role in the prognostic health management in industry 4.0 panorama, as it detects the failure modes or anomaly phenomena coming from all the sensors in real time and raises alert for reducing cost at a scale. From the data perspective, an anomaly can be viewed as an observation which deviates at a distance larger than the rest normal observations under given metrics. Depends on the availability and categories of the labeled data, the anomaly detection task can be divided into the following three types: [76]:

- *supervised anomaly detection*: fully labeled training and testing data, classify the normal and abnormal data on training data, then make inference on testing data to detect anomaly based on classification results
- *semi-supervised anomaly detection*: partial labeled training data with normal samples, training on normal data to learn the characteristics of normal characteristics, predict on the test data based on the difference in distance under given metrics
- *unsupervised anomaly detection*: fully unlabelled data with instance inference, usually has the assumption that fewer anomalies than the majority of the normal instances are comprised in the dataset.

In this chapter, the semi-supervised anomaly detection problem is investigated. Some previous literature have demonstrated using the generative model to solve the anomaly detection problem [77]. The original varitional autoencoder is proposed for solve the intractability of the marginal likelihood problem [78], and by utilization of the latent probability distribution, the reconstruction probability based anomaly detection is proposed in [77]. A generative adversarial network based model is proposed by combining the mutual impact of the generator and discriminator and the second discriminator is utilized for distance based anomaly detection [79]. A symmetric graph convolutional autoencoder based on Laplacian smoothing and sharpening is proposed to design numerical stable decoder and efficient subspace clustering cost. However, the manually setting of the linear combination of sharpening process leads to extra operations for constraining the eigenvalue of normalized Laplacian within the range of [0, 2] [80]. An anomaly detection and segmentation based on deep convolutional autoencoder is proposed for semiconductor manufacturing [81]. The synthetic wafer map is generated using the Poisson progress and it is utilized for neural network training, validation and testing. The real wafer data is used for testing the model performance on unseen data. A semi-supervised model with minimal involvement of domain experts is proposed for the anomaly detection in concrete structures. The convolutional autoencoder is adopted to evaluate the pixel-wise reconstruction mean squared error of the three RGB channels [82].

Compared to deploy the off-the-self model, a growing interest is to study the inner structure of the data so that structured information inside of the images to assist the anomaly detection. Graph is a highly potential candidate for representation learning of the structured information. Graph neural networks (GNNs) are structured models for embedding the relational information as connected nodes and edges. Due to its powerful and flexible to build, it is widely used in different areas including but

not limited to social network, drug discovery, traffic prediction, brain connectivity studies, knowledge graph [83]. GNN has the potential to explore the inner structure of the data, and this kind of representation learning can be extended to anomaly detection .

Inspired by the vartioanl autoencoder (VAE) base anomaly detection proposed in [77], an end-to-end framework variational graph neural network based anomaly detection framework is provided. There is no direct mapping between pixel-wise images to graph. The superpixel method is adopted as the bridge for the mapping to obtain the feature matrix and graph topology. A graph convolutional neural network with pooling layer and upsampling design is proposed to mimic the similar setting as the encoder and decoder in the convolutional VAE. The training is only carried on the normal samples to learn the features of the normal scenario. The difference of the reconstructed images and original images is adopted as the criterion for the threshold based anomaly detection.

The rest of the chapter is arranged as follow. In section 5.2, the SLIC algorithm is introduced to generate the inputs to the variational graph anomaly detector. In section 5.3, the variational autoencoder and variational graph autoencoder are introduced. In section 5.4, the anomaly detection framework and algorithms for anomaly detection are demonstrated. In section 5.5, we verify the framework on road surface crack dataset and compare the VAE and VGAE based anomaly detection. Finally, the conclusion is drawn in section 5.6.

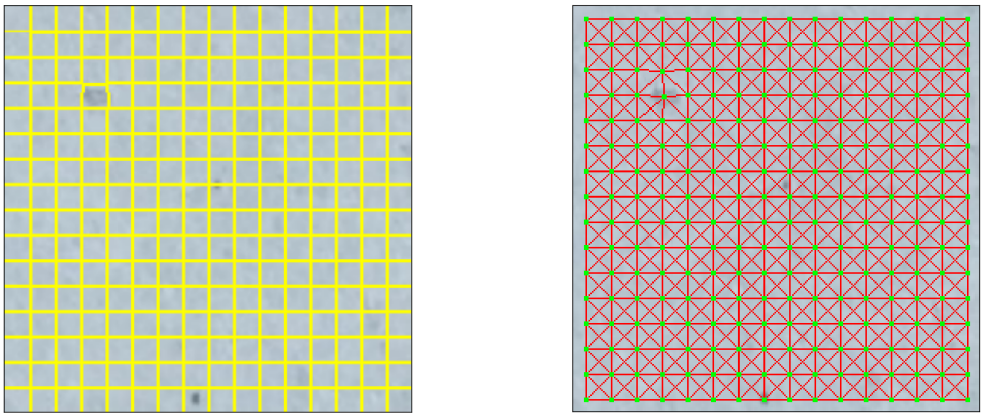
5.2 Superpixel and feature engineering

One of the approaches to explore the inherent structure of the images is by adopting the angles from the graph theory. To create the graph description of the images, a bilateral filtering idea is proposed to represent weight $w_{i,j}$ as the edge

connecting the nodes (corresponding to pixels) i and j [84]. Consider the Gaussian kernel, the weight can be represented as:

$$w_{i,j} = \exp\left(-\frac{\|\mathbf{I}_i - \mathbf{I}_j\|_2^2}{\sigma_l^2}\right) \exp\left(-\frac{\|x_i - x_j\|_2^2}{\sigma_x^2}\right) \tag{5.1}$$

However the pixel-level mapping will increasing the size of the graph at a scale, which brings about computation load for images with high resolution. For instance, a 28-by-28 pixel MNIST handwritten images will lead to a graph with $28^2 = 784$ nodes, and its corresponding adjacency matrix $\mathbf{A} \in \mathbb{R}^{784\times784}$, which is not suited graph-level tasks due to square of the matrix size.



(a) (b)

Figure 5.1. The boundary plot and RAG plot of the **no crack** images of road surface with SLIC algorithm imposing on.

Superpixel algorithms group pixels into perceptually meaningful atomic regions which can be used to replace the rigid structure of the pixel grid. They capture image redundancy, provide a convenient primitive from which to compute image features, and greatly reduce the complexity of subsequent image processing tasks. A

superpixel-driven graph transform is proposed for image compression. The algorithm builds a separate graph for each superpixel, so that the number of nodes and edges are not the same for each graph. This kind of heterogeneous graphs generally not suited for current requirements for graph classification tasks in graph neural networks, which requires additional operations before the inference [85]. Furthermore, the superpixel graph based learning is proposed for graph classification. The feature engineering followed by the manifold SLIC (MSLIC) generate the mean feature vector, weight feature vector and centroidal location for each superpixel and then fit in the downstream SVM for classification [86].

A brief introduction to the SLIC algorithm is listed in Algorithm **??**. The boundary plotting and region adjacency graph (RAG) [87] plotting of the crack and no crack surface after the SLIC algorithms are demonstrated in Fig. 5.2 and Fig. 5.1. The images are processed with 256 segments and each segment is the corresponding superpixel. For each superpixel, the statistics (mean, std, max, min, median) value of the superpixel are extracted for each channel of the images, and the feature matrix $\mathbf{X} \in \mathbb{R}^{N \times F}$ is generated where $N = 256$ is the total number of the superpixels and F is $5 \times 3 = 15$ is the feature dimension. The corresponding adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$ is created using the segraph library [1], which is based on the idea of conditional random field. The feature matrix and adjacency matrix serve as the input to the VGAE anomaly detection model, which is demonstrated in details in the next section.

[1]The segraph library provides modules for creating graphs from SLIC segments, the original source link is https://github.com/alivcor/segraph

Algorithm 4 SLIC superpixel segmentation

/* initialization */

initialize cluster centers $C_k = [l_k, a_k, b_k, x_k, y_k]^T$ by sampling pixels at regular grid steps S.

Move cluster centers to the lowest gradient position in a 3×3 neighborhood.

Set label $l(i) = -1$ for each pixel i.

Set distance $d(i) = \infty$ for each pixel i

repeat

/* assignment */

for *each cluster center* C_k **do**

for *each pixel* i *in a* $2S \times 2S$ *region around* C_k **do**

Compute the distance D between C_k and i.

if $D < d(i)$ **then**

set $d(i) = D$

set $l(i) = k$

end

end

end

/* Update */

Compute new cluster centers

Compute residual error E **until** $E \leq$ threshold

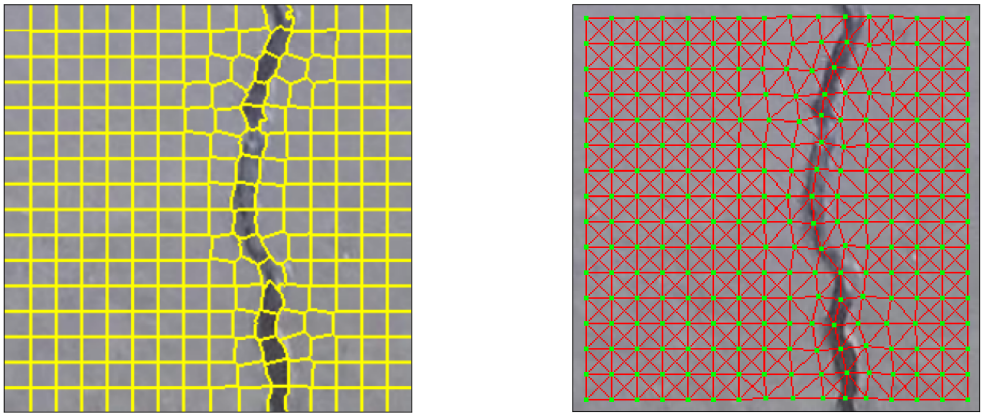
5.3 Generative model for anomaly detection

5.3.1 Variational autoencoder

There will be the bottleneck when performing efficient approximate inference and learning with directed probabilistic models which have intractable posterior dis-

tributions that cannot be solved by traditional model-based models. Consider the data generating process which involves unobserved latent variables $\mathbf{z}$ and and observation $\mathbf{x}$. $\mathbf{z}$ is generated from some prior distribution $p_{\boldsymbol{\theta}}(\mathbf{z})$ and $\mathbf{x}$ is generated from conditional distribution $p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z})$. The objective of the VAE is tried to solve the intractable marginal likelihood $p_{\boldsymbol{\theta}}(\mathbf{x}) = \int p_{\boldsymbol{\theta}}(\mathbf{z}) p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z}) d\mathbf{z}$. The posterior term is approximated by the recognition model $p_{\boldsymbol{\theta}}(\mathbf{z} \mid \mathbf{x})$. Taking the logarithm of the marginal likelihood $\log p_{\boldsymbol{\theta}}(\mathbf{x})$, it can be represented as:

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}) = D_{KL}\left(q_{\phi}(\mathbf{z} \mid \mathbf{x}) \| p_{\boldsymbol{\theta}}(\mathbf{z} \mid \mathbf{x})\right) + \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}). \tag{5.2}$$



(a) (b)

Figure 5.2. The boundary plot and RAG plot of the **crack** images of road surface with SLIC algorithm imposing on.

Since $D_{KL}\left(q_{\phi}(\mathbf{z} \mid \mathbf{x}) \| p_{\boldsymbol{\theta}}(\mathbf{z} \mid \mathbf{x})\right) \geq 0$, we have the following,

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}) \geq \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}) = \mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x})}\left[-\log q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x}) + \log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})\right]. \tag{5.3}$$

This term is also called the variational lower bound on the marginal likelihood of data point and it is expected to differentiate and optimize over it. It is treated

as the optimization function and to be differentiated and optimized over generative parameter $\boldsymbol{\theta}$ and variational parameter $\boldsymbol{\phi}$.

Let's consider the special case of where latent variables comes from a standard Gaussian distribution $p_{\boldsymbol{\theta}}(\mathbf{z}) = \mathcal{N}(\mathbf{z}; \mathbf{0}, \mathbf{I})$ and variational approximate posterior to be a multivariate Gaussian with a diagonal covariance:

$$\log q_{\phi}\left(\mathbf{z} \mid \mathbf{x}\right) = \log \mathcal{N}\left(\mathbf{z}; \boldsymbol{\mu}, \boldsymbol{\sigma}\mathbf{I}\right), \tag{5.4}$$

where it serves as the encoder of the VAE model. Then the latent variables is sampled from the posterior distribution and sent to the decoder model $p_{\boldsymbol{\theta}}(\mathbf{z} \mid \mathbf{x})$. The reparameterization trick $\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ is adopted for representation of the latent variable. The CNN modules and convolutional transposed layers are adopted to represent both the encoder and decoder model. The latent variables are generated as the last output of the dense layer in the encoder model. The whole model is demonstrated in Fig. 5.3. The Loss function contains both the reconstruction loss and KL-distance loss [88]:

$$\mathcal{L}\left(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}\right) \simeq \frac{1}{2} \sum_{j=1}^{J} \left(1 + \log\left(\left(\sigma_j\right)^2\right) - \left(\mu_j\right)^2 - \left(\sigma_j\right)^2\right) + \frac{1}{L} \sum_{l=1}^{L} \log p_{\boldsymbol{\theta}}\left(\mathbf{x} \mid \mathbf{z}\right) \tag{5.5}$$

5.3.2 Variational graph autoencoder

Similar to the idea of the VAE, we extend the generative model based approximation of the posterior distribution idea on the graph model. Consider the case of undirected graph $\mathcal{G}(V, E)$ with $N =\mid V \mid$ nodes. Given the feature matrix $\mathbf{X} \in \mathbb{R} \in N \times F$ and adjacency matrix $\mathbf{A} \in \mathbb{R} \in N \times N$, as proposed in the seminal work of the VGAE [89], the module can be divided in the following parts:

- Inference model (encoder part): $q(\mathbf{Z} \mid \mathbf{X}, \mathbf{A}) = \prod_{i=1}^{N} q\left(\mathbf{z}_i \mid \mathbf{X}, \mathbf{A}\right)$, where for each component (vector) of embedding matrix $\mathbf{Z}$, it is modeled using Gaussian distribution $q\left(\mathbf{z}_i \mid \mathbf{X}, \mathbf{A}\right) = \mathcal{N}\left(\mathbf{z}_i \mid \boldsymbol{\mu}_i, \operatorname{diag}\left(\boldsymbol{\sigma}_i^2\right)\right)$

- Generative model (decoder part):$p(\mathbf{A} \mid \mathbf{Z}) = \prod_{i=1}^{N} \prod_{j=1}^{N} p\left(A_{ij} \mid \mathbf{z}_i, \mathbf{z}_j\right)$, where $p\left(A_{ij} = 1 \mid \mathbf{z}_i, \mathbf{z}_j\right) = \sigma\left(\mathbf{z}_i^{\top} \mathbf{z}_j\right)$. The second term essentially is achieved by calculating the matrix multiplication of the $\mathbf{Z}^T\mathbf{Z}$, flatten it to a vector and impose the sigmoid function to get the probability
- Optimization (ELBO): $\mathcal{L} = \mathbb{E}_{q(\mathbf{Z}|\mathbf{X},\mathbf{A})}[\log p(\mathbf{A} \mid \mathbf{Z})] - \mathrm{KL}[q(\mathbf{Z} \mid \mathbf{X}, \mathbf{A}) \| p(\mathbf{Z})]$

The above framework is mainly for the node-level inference and lack the depth of the neural network. Apart from the decoding part in the original VGAE, the hierarchial model with coarsening and upsampling is introduced for the anomaly detection framework. Different from the idea of the pooling layer in CNN, there is no specified filter size in GNN due to the heterogeneous connection of the nodes and edges. Therefore we need to customize the pooling layer to mimic the pooling operation in GNN. We leverage the mincutpooling proposed in [90] to achieve this function. As mentioned earlier in Chapter 1, the GNN can be written as follows:

$$\bar{\mathbf{X}} = GNN(\mathbf{X}, \tilde{\mathbf{A}}; \Theta_{GNN}). \tag{5.6}$$

The cluster assignment matrix $\mathbf{S} = MLP(\bar{\mathbf{X}}; \Theta_{MLP})$ is designed as the sampling module for the $\bar{\mathbf{X}}$, where MLP is the multi-layer perceptron module and $s_[ij] \in [0, 1]$ due to the softmax activation function. The idea is to relax the original idea of the min cut optimization problem as:

$$\mathcal{L}_u = \mathcal{L}_c + \mathcal{L}_o = \underbrace{-\frac{\mathrm{Tr}\left(\mathbf{S}^T\tilde{\mathbf{A}}\mathbf{S}\right)}{\mathrm{Tr}\left(\mathbf{S}^T\tilde{\mathbf{D}}\mathbf{S}\right)}}_{\mathcal{L}_c} + \underbrace{\left\| \frac{\mathbf{S}^T\mathbf{S}}{\|\mathbf{S}^T\mathbf{S}\|_F} - \frac{\mathbf{I}_K}{\sqrt{K}} \right\|_E}_{\mathcal{L}_o}, \tag{5.7}$$

where minimizing the cut loss $\mathcal{L}_c$ will help the strongly connected nodes to be clustered together. The second orthogonality loss term $\mathcal{L}_o$ plays reversed way such that the cluster assignments to be orthogonal. The two adversarial terms contribute to constrain the degeneration of each other.

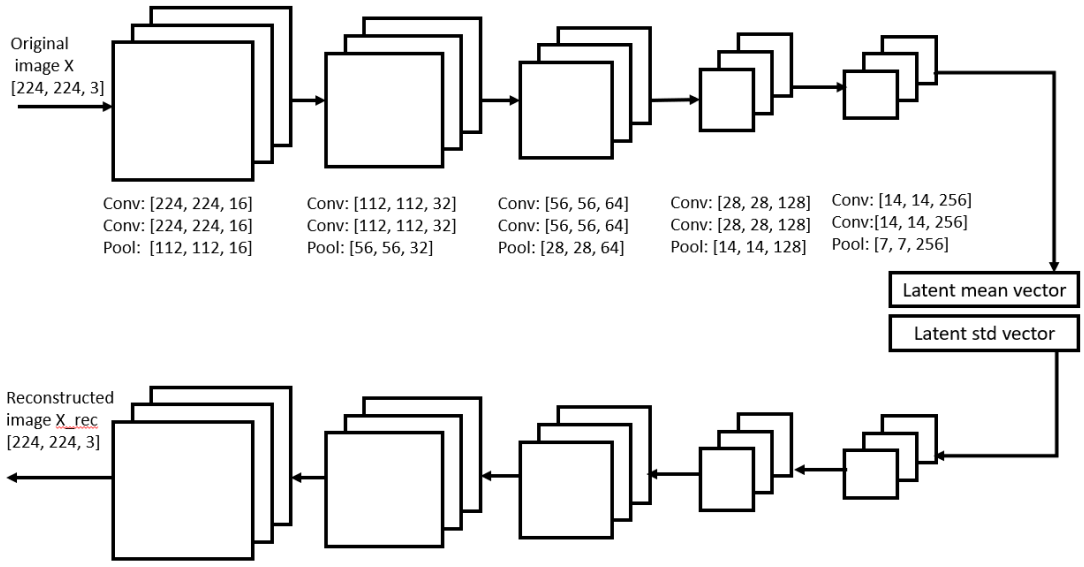


Figure 5.3. schematic diagram of the CVAE based anomaly detection.

After solving the optimization problem, the corresponding coarsened adjacency matrix and the pooled feature matrix can be represented as:

$$\mathbf{A}^{\text{pool}} = \mathbf{S}^T\mathbf{A}\mathbf{S}, \quad \mathbf{X}^{\text{pool}} = \mathbf{S}^T\mathbf{X} \tag{5.8}$$

And it fits in the encoder module of the VGAE anomaly detection module. For the reverse part in the decoder module, the upsampling operation can be defined as:

$$\mathbf{X}^{\text{rec}} = \mathbf{S}^{\text{pool}}, \quad \mathbf{A}^{\text{rec}} = \mathbf{S}\mathbf{A}^{\text{pool}}\mathbf{S}^T \tag{5.9}$$

The RMSE between the original and the reconstructed feature matrix $\|\mathbf{X} - \mathbf{X}^{\text{rec}}\|^2$ serve as the reconstruction loss and the corresponding KL divergence is similar to formula 5.5. The whole module of the VGAE based anomaly detection is demonstrated in Fig. 5.4.

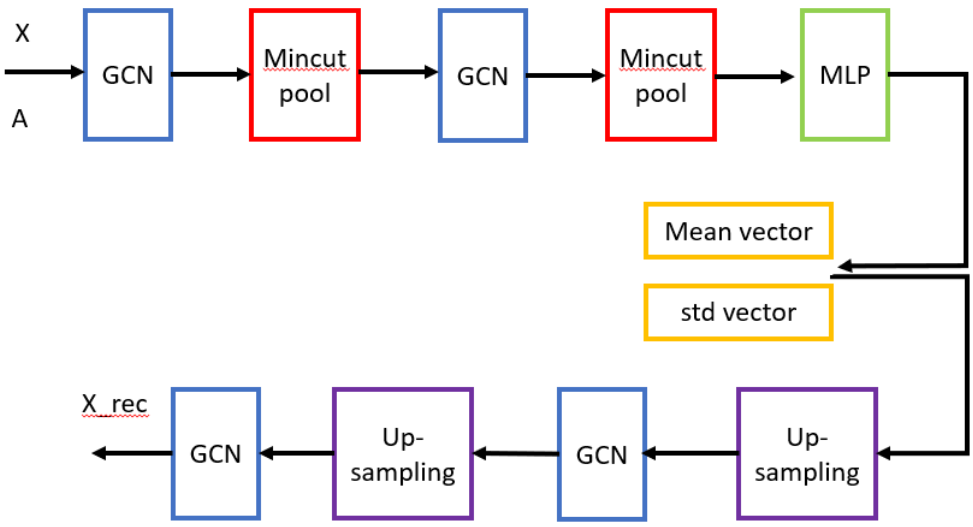


Figure 5.4. schematic diagram of the VGAE based anomaly detection.

5.4 Experiment setting and performance analysis

5.4.1 Dataset

In order to verify the feasibility of the proposed VGAE based anomaly detection framework, we tested on the road surface dataset and compare the performance with the VAE based method. As one of the typical applications in prognostic and health management, Crack detection serves a crucial role in the building inspection, finding the cracks and determining the building health [91]. The original dataset is generated from 458 high-resolution images (4032 pixels to 3024 pixels) taken approximately one meter away from walls and floors of several concrete buildings in METU Campus with the camera facing directly to the target [92]. The image data are divided into two as negative (without crack) and positive (with crack) in separate folder for image classification. Data augmentation in terms of random rotation or flipping or tilting

is not applied for the VAE setting in this study. Image samples of the dataset are shown in Fig. 5.5 and Fig. 5.6, for no crack and crack surface separately [1].

5.4.2 Experiment setting and performance analysis

we run the experiment on the kaggle kernel [1], which provides the NVIDIA TESLA P100 GPU, with CPU RAM up to 16GB and GPU RAM up to 16GB. Due to limit of the 5GB disk. We randomly choose 4096 images of normal images and 2048 images of the abnormal images. To setup the semi-supervised anomaly detection setting, at the training stage, the random choosing 2048 normal images are used for training only and rest of the normal and abnormal images are used for testing. During the training stages, the VAE and VGAE tries to learn the parameters related to the generating process of the normal images and tune the parameters in order to approximate it. During the test stage, we use the trained model for the inference of test dataset — mixture of the normal and abnormal images. The anomaly score is based on the pixel-wise root mean square error (PRMSE) of the three channels. If the threshold is above the median of the PRMSE of the training dataset, it is determined as the abnormal.

The Adam optimizer is utilized for the VAE and VGAE. The loss functions of the VAE and VGAE are the corresponding ones defined in the previous sections. The learning rate is 10^{-3} and the l2 regularization of the GCN module is 10^{-4}. We train on 500 epochs with batch size of 32. We run the standard F1 score, confusion matrix and AUC-ROC analysis on both the VAE and VGAE. The total number of parameters of the VAE and VGAE are around **4.8 million and 78 thousand** respectively so that the VGAE almost save 60 times in terms of network storage cost. We run 500 epochs

[1]https://www.kaggle.com/arunrk7/surface-crack-detection

[1]https://www.kaggle.com/docs/efficient-gpu-usage

models	total No. of network parameters	F1 score	AUC
VAE	4.8M	0.852	0.853
VGAE	80k	0.882	0.885

Table 5.1. Comparison of performance metrics for VAE and VGAE based anomaly detection model

during the training stages and evaluate the performance via the standard confusion matrix, F1-score and area under curve (AUC).

The details of the performance are also listed in table 5.1. We can see that VGAE achieve slightly better for both of the crack and no crack cases under the F1 score and AUC. From the Fig. 5.7 and Fig. 5.8, both of the two anomaly detection perform better on no crack cases. This may due to the training process and also sensitive to the threshold choosing.

To summary, the problem of the graph based neural network for anomaly detection is investigated. Superpixels induced by SLIC algorithms serve as the input to the graph model. The end-to-end variational graph autoencoder is built for semi-supervised anomaly detection. The training is only on the normal samples and achieve the inference on all the normal and potential abnormal samples. Anomaly score is calculated by the RMSE of the reconstructed samples compared to the normal samples and compare to the threshold. Future work will focus on the geometric descriptor for graph representation of the images for more complex anomaly settings and structured graph neural network design, such as pooling layers and upsampling layer.

Figure 5.5. sixteen different **no crack** images of the road surface.

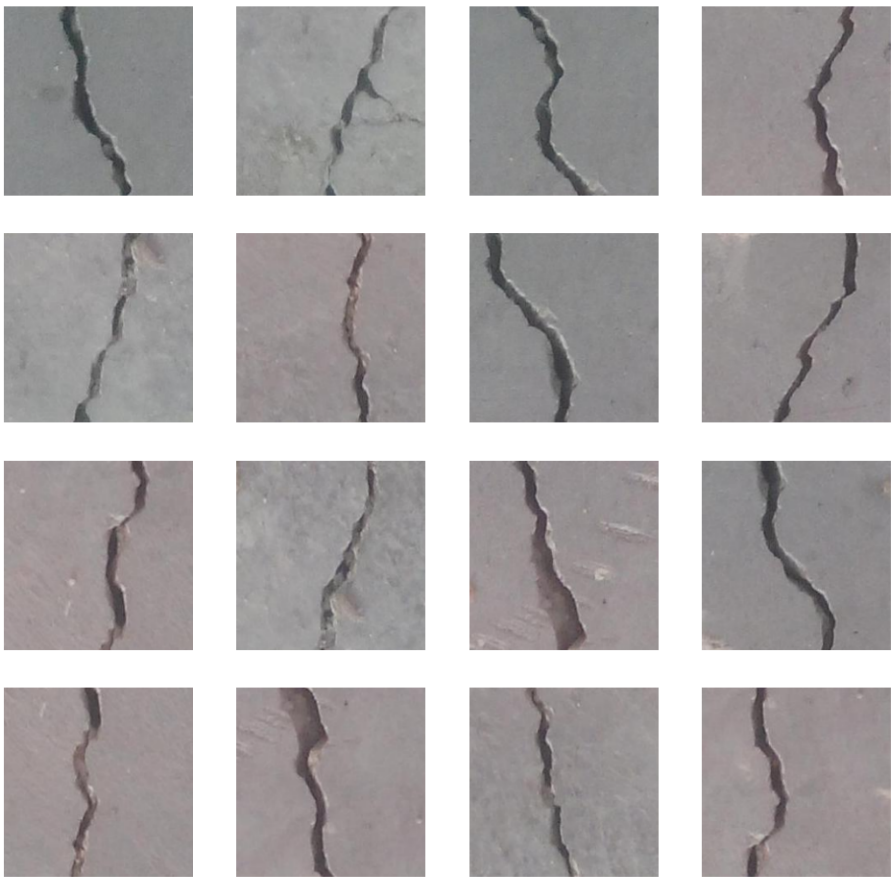
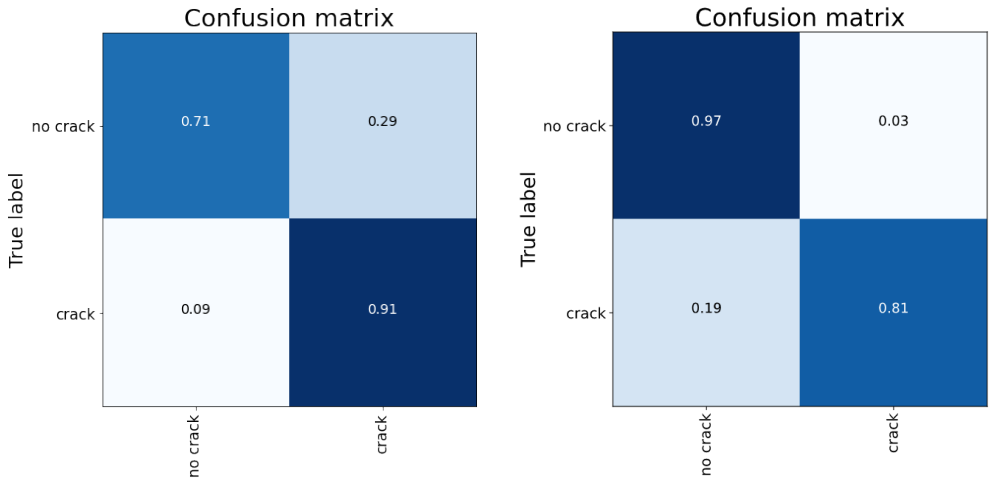


Figure 5.6. sixteen different **crack** images of the road surface.



(a) (b)

Figure 5.7. The boundary plot and RAG plot of the **crack** images of road surface with SLIC algorithm imposing on.

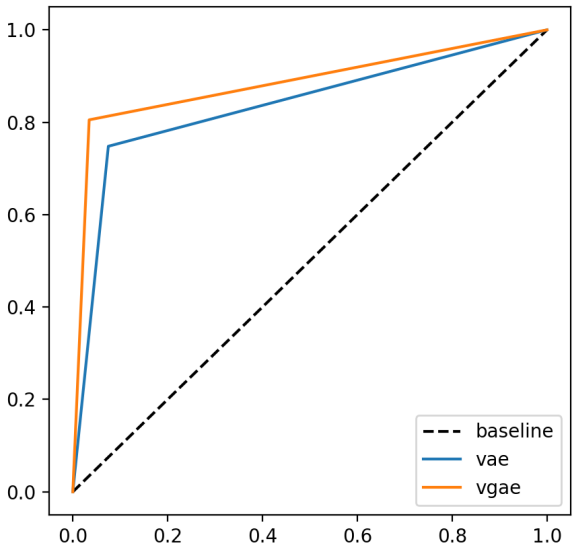


Figure 5.8. Comparison of the ROC curve and AUC score of the VAE and VGAE.

CHAPTER 6

Conclusion and Future Works

This chapter concludes the whole dissertation. It begins with a summary of the dissertation results and contributions, follows with a discussion of future research directions in further investigation of structured deep learning.

6.1 Summary

This dissertation has focused on structured deep learning and its applications, with particular interests on deep sequential modeling and graph neural networks. The contributions of this dissertation are:

1. *Structure recurrent neural network (SRNN) with orthogonal constraints*: Motivated by solving the vanishing gradient and exploding problems inside the recurrent neural network, the orthogonal constraints imposed on the recurrent weight matrix is investigated. By leveraging the two decomposition methods that lie on the Stiefel manifold, the weight matrix can be divided into several easy to implement parts with theoretic properties to control the vibration of the eigenvalues. The coordinate-descent like algorithms is adopted to design the corresponding training algorithms for the two decomposition mechanisms (orRNN and pdRNN). With the proposed methods, it is able to verify the memory keep abilities on synthetic datasets with the visualization of the pseudospectrum on how well the evolvement of the orthogonality is keeping. Moreover, the faster convergence as well as lower computation cost in terms of total number of network parameters are the corresponding benefits that the methods bring about.

2. *Application of SRNN for polys detection*: Consider the real scenario of deep sequential modeling of polyps detection based on vocal signal data. An end-to-end framework is proposed by adoption of the SRNN. The problem is formulated as the binary classification for anomaly detection purpose. The time-series data is expanded to time-frequency feature map by the construction of spectrogram with calculation of Mel frequency cepstral coefficients. This two dimensional feature map is fed into the network model row by row (treat as the batch). With the comparison of the four RNNs, the networks with the orthogonal constraints can achieve comparative accuracy (better sometimes) with faster convergence as well as lower network complexity overhead.
3. *Plug and play SRNN module for EEG sleep staging analysis*: Build the orthogonal constraints as the Tensorflow layer for plug and play purpose and apply it on the EEG sleep staging analysis. The EEG sleep staging analysis is formulated as the multiple classification problems. Compared to the physical based model for time-frequency expansion, the one-dimensional convolutional neural network is adopted for generating the time frequency feature maps. The coarse and fine levels of the feature maps are generated separately and concatenated together to feed in the plug and play layer. The orthogonal constrained recurrent neural network has better performances in terms of accuracy, convergence speed as well as lower network overland. It also demonstrates its scalability to involve with other machine learning and deep learning modules.
4. *Graph neural network based anomaly detection for road surface crack detection*: Propose an end-to-end graph neural network based anomaly detection for images. The superpixels induced by the SLIC algorithms and conditional random field based methods are introduced to generate the inputs to the variational graph autoencoder (VGAE). The VGAE with mincutpooling and upsampling

layers mimic the similar operations in convolutional variational autoencoder (CVAE) but tailored for the specific fusion of the feature matrix X and adjacency matrix $\mathbf{A}$. The framework is verified on the road surface crack detection datasets and compared with CVAE. The results demonstrate the efficiency of the VGAE regarding saving the storage of the network load while achieve comparatively better performance on F1-score and area under curve score.

6.2 Future Direction

6.2.1 Dynamics of the Recurrent Neural Networks

It is believed that for the interpretation of the RNN models, the dynamics of the RNN is required to disclose so that the RNN model with specified functions can be generated. The relation between wave physics and RNN is disclosed through the comparative study of vowel classification. The interpretation of the RNN is achieved by measurements of the time integrated power at each probe and the automatic differentiation is demonstrated though the iteration of the material density [93]. The ordinary differential equation is combined with gated-control RNN (LSTM, GRU) to identity learning vectors and the curve dynamics and it is further verified through the human activity dataset [94]. Compared to the gated-control mechanism, the second order term in the orthogonal constrained RNN is a more natural choice due to the dynamic representation and it is believed that there is a connection between these two. From the signal processing point of view, the RNN can be regarded as the finite impulse response (FIR). Vanilla RNN and gated-control doesn't take the long-range dependence into consideration, which leads to short memory. By implement a memory filter, the memory augmented RNN is helpful for the long-memory issue

[95]. By incorporating the impulse response and dynamics into consideration, there will be a more robust design in structured RNN.

6.2.2 Structured representation of the GNN

There are two-folds regarding the structured representation of the GNN.

1. *Feature-wise representation of the graph*: Current model for the feature matrix and adjacency matrix are based on the statistics and conditional random field model. There are more potentials to explore the representation of the graph information. For example, characteristic functions based methods are utilized to form an elastic representation of multiscale representation of the feature distribution [96, 97]. Graph kernel is also a potential candidate for specific representation of the graph features [98]. The fusion of both the graph features and graph topology may serve as a more synthetic way for the image based anomaly detection setting [83]. Inspired by those methods, a automatic domain driven end-to-end graph multiscale feature generation and fusion method is worth for the further research purpose.
2. *Layer design for the GNN*: The mincutpool pooling layer is utilized in this work, however it is pointed out that mincutpool may not optimize its own objective function [99] and the corresponding new deep modularity network (DMN) is proposed. Also the upsampling layer doesn't play a consensus role like the convolutional transpose layer in CNN and it is worthwhile to look for the new design for transposed operation.

CHAPTER 7

Publication List

1. F. Zhu and Q. Liang, "Rethink of Orthographic Constraints On RNN and its Application in Sequential Modelling", Submitted to *IEEE Transactions on Neural Networks and Learning System*
2. F. Zhu and Q. Liang, "Orthographic Retraction Based RNN With Application to Vocal Data", Submitted to *IEEE Transactions on Neural Networks and Learning System*.
3. F. Zhu and Q. Liang, "OCRNN: An Orthogonal Constrained Recurrent Neural Network for Sleep Analysis Based on EEG Data." *Ad Hoc Networks pp: 102178, 2020*
4. F. Zhu and Q. Liang, "Sequential Modeling for Polyps Identification From the Vocal Data"*International Conference on Communications,Signal Processing, and Systems (CSPS)*, vol. 516, pp: 762-769, 2019.
5. T. Wang, F. Zhu and J. Liang, "Soil pH Classification Based on LSTM via UWB Radar Echoes", *International Conference on Communications,Signal Processing, and Systems (CSPS)*, vol. 516, pp: 945-954, 2019.
6. F. Zhu and J. Liang. "Soil Moisture Retrieval Based on Interval Type-2 Fuzzy Logic Systems", *IEEE Access*, Vol. 6, pp: 29846 - 29857, 2018.
7. J. J. Chen, F. Zhu and Q. Liang, " Information Theoretic Bounds for Sparse Reconstruction in Random Noise", *IEEE Access*, vol. 7, pp: 02304-102312, 2019.

8. K. Liao, J. Si, F. Zhu, and X. He, "Radar HRRP target recognition based on concatenated deep neural networks." *IEEE Access*, vol.6, pp: 29211-29218, 2018.
9. N. Wu, F. Zhu, and Q. Liang, "Evaluating spatial resolution and channel capacity of sparse cylindrical arrays for massive MIMO", *IEEE Access*, vol.5, pp: 23994-24003, 2017.
10. F. Zhu, N. Wu, and Q. Liang. "Channel estimation for massive MIMO with 2-D nested array deployment." *Physical Communication*, pp: 432-437, 2017.

REFERENCES

[1] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, "Learning structured sparsity in deep neural networks," in *Advances in neural information processing systems*, 2016, pp. 2074–2082.

[2] T.-J. Yang, Y.-H. Chen, and V. Sze, "Designing energy-efficient convolutional neural networks using energy-aware pruning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5687–5695.

[3] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[4] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *International Conference on Machine Learning*, 2013, pp. 1310–1318.

[5] R. Pascanu, C. Gulcehre, K. Cho, and Y. Bengio, "How to construct deep recurrent neural networks," in *Proceedings of the Second International Conference on Learning Representations (ICLR 2014)*, 2014.

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

[7] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[8] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[9] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[10] M. Arjovsky, A. Shah, and Y. Bengio, "Unitary evolution recurrent neural networks," in *International Conference on Machine Learning*. PMLR, 2016, pp. 1120–1128.

[11] E. Vorontsov, C. Trabelsi, S. Kadoury, and C. Pal, "On orthogonality and learning recurrent networks with long term dependencies," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, JMLR. org. PMLR, 2017, pp. 3570–3578.

[12] S. V. N. Vishwanathan, N. N. Schraudolph, R. Kondor, and K. M. Borgwardt, "Graph kernels," *The Journal of Machine Learning Research*, vol. 11, pp. 1201–1242, 2010.

[13] S. E. Schaeffer, "Graph clustering," *Computer science review*, vol. 1, no. 1, pp. 27–64, 2007.

[14] N. M. Kriege, F. D. Johansson, and C. Morris, "A survey on graph kernels," *Applied Network Science*, vol. 5, no. 1, pp. 1–42, 2020.

[15] T. Hashimoto, Y. Sun, and T. Jaakkola, "From random walks to distances on unweighted graphs," in *Advances in neural information processing systems*, 2015, pp. 3429–3437.

[16] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances in neural information processing systems*, 2016, pp. 3844–3852.

[17] T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," in *Proceedings of the 5th International Conference*

on Learning Representations, ser. ICLR '17, 2017. [Online]. Available: https://openreview.net/forum?id=SJU4ayYgl

[18] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "Lstm: A search space odyssey," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 10, pp. 2222–2232, 2017.

[19] Y. Bengio, P. Frasconi, and P. Simard, "The problem of learning long-term dependencies in recurrent networks," in *IEEE international conference on neural networks*. IEEE, 1993, pp. 1183–1188.

[20] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.

[21] S. Wisdom, T. Powers, J. Hershey, J. Le Roux, and L. Atlas, "Full-capacity unitary recurrent neural networks," in *Advances in Neural Information Processing Systems*, 2016, pp. 4880–4888.

[22] Z. Mhammedi, A. Hellicar, A. Rahman, and J. Bailey, "Efficient orthogonal parametrisation of recurrent neural networks using householder reflections," in *Proceedings of the 34th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, D. Precup and Y. W. Teh, Eds., vol. 70. International Convention Centre, Sydney, Australia: PMLR, 06–11 Aug 2017, pp. 2401–2409. [Online]. Available: http://proceedings.mlr.press/v70/mhammedi17a.html

[23] L. Jing, Y. Shen, T. Dubcek, J. Peurifoy, S. Skirlo, Y. LeCun, M. Tegmark, and M. Soljačić, "Tunable efficient unitary neural networks (EUNN) and their application to RNNs," in *Proceedings of the 34th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 70, 2017, pp. 1733–1741.

[24] C. Jose, M. Cisse, and F. Fleuret, "Kronecker recurrent units," *arXiv preprint arXiv:1705.10142*, 2017.

[25] I. Shafran, T. Bagby, and R. J. Skerry-Ryan, "Complex evolution recurrent neural networks (cernns)," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, April 2018, pp. 5854–5858.

[26] E. Vorontsov, C. Trabelsi, S. Kadoury, and C. Pal, "On orthogonality and learning recurrent networks with long term dependencies," in *Proceedings of the 34th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, D. Precup and Y. W. Teh, Eds., vol. 70. International Convention Centre, Sydney, Australia: PMLR, 06–11 Aug 2017, pp. 3570–3578. [Online]. Available: http://proceedings.mlr.press/v70/vorontsov17a.html

[27] K. Helfrich, D. Willmott, and Q. Ye, "Orthogonal recurrent neural networks with scaled Cayley transform," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. Stockholmsmässan, Stockholm Sweden: PMLR, 10–15 Jul 2018, pp. 1969–1978. [Online]. Available: http://proceedings.mlr.press/v80/helfrich18a.html

[28] K. D. Maduranga, K. E. Helfrich, and Q. Ye, "Complex unitary recurrent neural networks using scaled cayley transform," *arXiv preprint arXiv:1811.04142*, 2018.

[29] P.-A. Absil, R. Mahony, and R. Sepulchre, *Optimization algorithms on matrix manifolds*. Princeton University Press, 2009.

[30] P.-A. Absil and J. Malick, "Projection-like retractions on matrix manifolds," *SIAM Journal on Optimization*, vol. 22, no. 1, pp. 135–158, 2012.

[31] T. Kaneko, S. Fiori, and T. Tanaka, "Empirical arithmetic averaging over the compact stiefel manifold," *IEEE Transactions on Signal Processing*, vol. 61, no. 4, pp. 883–894, 2013.

[32] K. B. Petersen and M. S. Pedersen, "The matrix cookbook (version: November 15, 2012)," 2012.

[33] L. N. Trefethen and M. Embree, *Spectra and pseudospectra: the behavior of nonnormal matrices and operators*. Princeton University Press, 2005.

[34] B. Sengupta and K. J. Friston, "How robust are deep neural networks?" *arXiv preprint arXiv:1804.11313*, 2018.

[35] G. André. Pseudopy. [Online]. Available: https://github.com/andrenarchy/pseudopy

[36] T. J. Wang, J. M. Massaro, D. Levy, R. S. Vasan, P. A. Wolf, R. B. D'Agostino, M. G. Larson, W. B. Kannel, and E. J. Benjamin, "A risk score for predicting stroke or death in individuals with new-onset atrial fibrillation in the community: the framingham heart study," *Jama*, vol. 290, no. 8, pp. 1049–1056, 2003.

[37] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[38] A. Roebroeck, A. K. Seth, and P. Valdes-Sosa, "Causal time series analysis of functional magnetic resonance imaging data." in *NIPS mini-symposium on causality in time series*, 2011, pp. 65–94.

[39] Z. Zhang, "Mechanics of human voice production and control," *The Journal of the Acoustical Society of America*, vol. 140, no. 4, pp. 2614–2635, 2016.

[40] ——, "Cause-effect relationship between vocal fold physiology and voice production in a three-dimensional phonation model," *The Journal of the Acoustical Society of America*, vol. 139, no. 4, pp. 1493–1507, 2016.

[41] J. Liang and F. Zhu, "Soil moisture retrieval from uwb sensor data by leveraging fuzzy logic," *IEEE Access*, vol. 6, pp. 29 846–29 857, 2018.

[42] Z. Zhong, T. Jiang, W. Zhang, H. Yao, and S. Xiao, "Analyzing speech of patients with vocal polyps based on channel parameters and fuzzy logic systems," *Computers & Mathematics with Applications*, vol. 62, no. 7, pp. 2834–2842, 2011.

[43] A. Tsanas, M. A. Little, P. E. McSharry, J. Spielman, and L. O. Ramig, "Novel speech signal processing algorithms for high-accuracy classification of parkinson's disease," *IEEE Transactions on Biomedical Engineering*, vol. 59, no. 5, pp. 1264–1271, May 2012.

[44] F. Zhu and Q. Liang, "Sequential modeling for polypus identificationfrom the vocal data," in *International Conference in Communications, Signal Processing, and Systems*. Springer, 2019, pp. 945–954.

[45] T. Wang, F. Zhu, and J. Liang, "Soil ph classification based on lstm via uwb radar echoes," in *International Conference in Communications, Signal Processing, and Systems*. Springer, 2019, pp. 762–769.

[46] J. C. Vasquez-Correa, T. Arias-Vergara, J. Orozco-Arroyave, B. M. Eskofier, J. Klucken, and E. Noth, "Multimodal assessment of parkinson's disease: a deep learning approach," *IEEE journal of biomedical and health informatics*, 2018.

[47] G. Gelly and J.-L. Gauvain, "Optimization of rnn-based speech activity detection," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 3, pp. 646–656, 2018.

[48] L. Gabrielli, S. Tomassetti, C. Zinato, and F. Piazza, "End-to-end learning for physics-based acoustic modeling," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 2, pp. 160–170, 2018.

[49] J.-X. Zhang, Z.-H. Ling, L.-J. Liu, Y. Jiang, and L.-R. Dai, "Sequence-to-sequence acoustic modeling for voice conversion," *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 27, no. 3, pp. 631–644, 2019.

[50] G. K. Anumanchipalli, J. Chartier, and E. F. Chang, "Speech synthesis from neural decoding of spoken sentences," *Nature*, vol. 568, no. 7753, p. 493, 2019.

[51] A. K. Halberstadt, "Heterogeneous acoustic measurements and multiple classifiers for speech recognition," Ph.D. dissertation, Massachusetts Institute of Technology, 1999.

[52] S. K. Mitra and Y. Kuo, *Digital signal processing: a computer-based approach*. McGraw-Hill Higher Education New York, 2006, vol. 2.

[53] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in python," 2015.

[54] F. Lotte, L. Bougrain, A. Cichocki, M. Clerc, M. Congedo, A. Rakotomamonjy, and F. Yger, "A review of classification algorithms for eeg-based brain–computer interfaces: a 10 year update," *Journal of neural engineering*, vol. 15, no. 3, p. 031005, 2018.

[55] P. L. Nunez, R. Srinivasan, *et al.*, *Electric fields of the brain: the neurophysics of EEG*. Oxford University Press, USA, 2006.

[56] A. Subasi and M. I. Gursoy, "Eeg signal classification using pca, ica, lda and support vector machines," *Expert systems with applications*, vol. 37, no. 12, pp. 8659–8666, 2010.

[57] B. Richhariya and M. Tanveer, "Eeg signal classification using universum support vector machine," *Expert Systems with Applications*, vol. 106, pp. 169–182, 2018.

[58] F. Artoni, A. Delorme, and S. Makeig, "Applying dimension reduction to eeg data by principal component analysis reduces the quality of its subsequent independent component decomposition," *NeuroImage*, vol. 175, pp. 176–187, 2018.

[59] X. Li, X. Chen, Y. Yan, W. Wei, and Z. J. Wang, "Classification of eeg signals using a multiple kernel learning support vector machine," *Sensors*, vol. 14, no. 7, pp. 12 784–12 802, 2014.

[60] S. Raghu and N. Sriraam, "Classification of focal and non-focal eeg signals using neighborhood component analysis and machine learning algorithms," *Expert Systems with Applications*, vol. 113, pp. 18–32, 2018.

[61] U. Orhan, M. Hekim, and M. Ozer, "Eeg signals classification using the k-means clustering and a multilayer perceptron neural network model," *Expert Systems with Applications*, vol. 38, no. 10, pp. 13 475–13 481, 2011.

[62] G. Ruffini, D. Ibañez, M. Castellano, S. Dunne, and A. Soria-Frisch, "Eeg-driven rnn classification for prognosis of neurodegeneration in at-risk patients," in *International Conference on Artificial Neural Networks*. Springer, 2016, pp. 306–313.

[63] W. Chen, S. Wang, X. Zhang, L. Yao, L. Yue, B. Qian, and X. Li, "Eeg-based motion intention recognition via multi-task rnns," in *Proceedings of the 2018 SIAM International Conference on Data Mining*. SIAM, 2018, pp. 279–287.

[64] M.-A. Moinnereau, S. Karimian-Azari, T. Sakuma, H. Boutani, L. Gheorghe, and T. H. Falk, "Eeg artifact removal for improved automated lane change detection while driving," in *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2018, pp. 1076–1080.

[65] S. Crea, M. Nann, E. Trigili, F. Cordella, A. Baldoni, F. J. Badesa, J. M. Catalán, L. Zollo, N. Vitiello, N. G. Aracil, *et al.*, "Feasibility and safety of shared eeg/eog and vision-guided autonomous whole-arm exoskeleton control to perform activities of daily living," *Scientific reports*, vol. 8, no. 1, p. 10823, 2018.

[66] Z. Tayeb, J. Fedjaev, N. Ghaboosi, C. Richter, L. Everding, X. Qu, Y. Wu, G. Cheng, and J. Conradt, "Validating deep neural networks for online decoding of motor imagery movements from eeg signals," *Sensors*, vol. 19, no. 1, p. 210, 2019.

[67] Y. Roy, H. Banville, I. Albuquerque, A. Gramfort, T. H. Falk, and J. Faubert, "Deep learning-based electroencephalography analysis: a systematic review," *Journal of neural engineering*, 2019.

[68] T. Wang, F. Zhu, and J. Liang, "Soil ph classification based on lstm via uwb radar echoes," in *International Conference in Communications, Signal Processing, and Systems*. Springer, 2018, pp. 762–769.

[69] F. Zhu, Q. Liang, and Z. Zhong, "Sequential modeling for polyps identification from the vocal data," in *International Conference in Communications, Signal Processing, and Systems*. Springer, 2018, pp. 945–954.

[70] Z. Duan and J. Liang, "Non-contact detection of vital signs using a uwb radar sensor," *IEEE Access*, vol. 7, pp. 36 888–36 895, 2018.

[71] T. Wang, J. Liang, and X. Liu, "Soil moisture retrieval algorithm based on tfa and cnn," *IEEE Access*, vol. 7, pp. 597–604, 2018.

[72] P. H. Hoang, "Deepsleepnet datasets," 2017, a Model for Automatic Sleep Stage Scoring based on Raw Single-Channel EEG, https://www.kaggle.com/phhasian0710/eeg-fpz-cz.

[73] A. Supratak, H. Dong, C. Wu, and Y. Guo, "Deepsleepnet: a model for automatic sleep stage scoring based on raw single-channel eeg," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 25, no. 11, pp. 1998–2008, 2017.

[74] B. Kemp, A. H. Zwinderman, B. Tuk, H. A. Kamphuisen, and J. J. Oberye, "Analysis of a sleep-dependent neuronal feedback loop: the slow-wave microcontinuity of the eeg," *IEEE Transactions on Biomedical Engineering*, vol. 47, no. 9, pp. 1185–1194, 2000.

[75] A. L. Goldberger, L. A. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley, "Physiobank,

physiotoolkit, and physionet: components of a new research resource for complex physiologic signals," *circulation*, vol. 101, no. 23, pp. e215–e220, 2000.

[76] R. Yao, C. Liu, L. Zhang, and P. Peng, "Unsupervised anomaly detection using variational auto-encoder based feature extraction," in *2019 IEEE International Conference on Prognostics and Health Management (ICPHM)*. IEEE, 2019, pp. 1–7.

[77] J. An and S. Cho, "Variational autoencoder based anomaly detection using reconstruction probability," *Special Lecture on IE*, vol. 2, no. 1, pp. 1–18, 2015.

[78] D. P. Kingma and M. Welling, "Auto-encoding variational bayes." in *ICLR*, Y. Bengio and Y. LeCun, Eds., 2014. [Online]. Available: http://dblp.uni-trier.de/db/conf/iclr/iclr2014.html#KingmaW13

[79] S. Akcay, A. Atapour-Abarghouei, and T. P. Breckon, "Ganomaly: Semi-supervised anomaly detection via adversarial training," in *Asian conference on computer vision*. Springer, 2018, pp. 622–637.

[80] B. Knyazev, G. W. Taylor, and M. Amer, "Understanding attention and generalization in graph neural networks," in *Advances in Neural Information Processing Systems*, 2019, pp. 4204–4214.

[81] K.-Y. Lee and J.-Y. Sim, "Warping residual based image stitching for large parallax," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 8198–8206.

[82] P. Bergmann, M. Fauser, D. Sattlegger, and C. Steger, "Mvtec ad–a comprehensive real-world dataset for unsupervised anomaly detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9592–9600.

[83] C. Deng, Z. Zhao, Y. Wang, Z. Zhang, and Z. Feng, "Graphzoom: A multi-level spectral approach for accurate and scalable graph embedding," in

International Conference on Learning Representations, 2020. [Online]. Available: https://openreview.net/forum?id=r1lGO0EKDH

[84] G. Cheung, E. Magli, Y. Tanaka, and M. K. Ng, "Graph spectral image processing," *Proceedings of the IEEE*, vol. 106, no. 5, pp. 907–930, 2018.

[85] G. Fracastoro, F. Verdoja, M. Grangetto, and E. Magli, "Superpixel-driven graph transform for image compression," in *2015 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2015, pp. 2631–2635.

[86] P. Sellars, A. I. Aviles-Rivero, and C.-B. Schönlieb, "Superpixel contracted graph-based learning for hyperspectral image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 58, no. 6, pp. 4180–4193, 2020.

[87] A. Trémeau and P. Colantoni, "Regions adjacency graph applied to color image segmentation," *IEEE Transactions on image processing*, vol. 9, no. 4, pp. 735–744, 2000.

[88] D. P. Kingma and M. Welling, "An introduction to variational autoencoders," *Foundations and Trends® in Machine Learning*, vol. 12, no. 4, pp. 307–392, 2019. [Online]. Available: http://dx.doi.org/10.1561/2200000056

[89] T. N. Kipf and M. Welling, "Variational graph auto-encoders," *NIPS Workshop on Bayesian Deep Learning*, 2016.

[90] F. M. Bianchi, D. Grattarola, and C. Alippi, "Spectral clustering with graph neural networks for graph pooling," in *Proceedings of the 37th international conference on Machine learning*. ACM, 2020, pp. 2729–2738.

[91] L. Zhang, F. Yang, Y. D. Zhang, and Y. J. Zhu, "Road crack detection using deep convolutional neural network," in *2016 IEEE international conference on image processing (ICIP)*. IEEE, 2016, pp. 3708–3712.

[92] Ç. F. Özgenel and A. G. Sorguç, "Performance comparison of pretrained convolutional neural networks on crack detection in buildings," in *ISARC. Proceedings*

of the International Symposium on Automation and Robotics in Construction, vol. 35. IAARC Publications, 2018, pp. 1–8.

[93] T. W. Hughes, I. A. Williamson, M. Minkov, and S. Fan, "Wave physics as an analog recurrent neural network," *Science advances*, vol. 5, no. 12, p. eaay6946, 2019.

[94] M. Habiba and B. A. Pearlmutter, "Neural ordinary differential equation based recurrent neural network model," *arXiv preprint arXiv:2005.09807*, 2020.

[95] J. Zhao, F. Huang, J. Lv, Y. Duan, Z. Qin, G. Li, and G. Tian, "Do rnn and lstm have long memory?" in *Proceedings of the 37th International Conference on Machine Learning*, vol. 119. PMLR, 2020.

[96] C. Donnat, M. Zitnik, D. Hallac, and J. Leskovec, "Learning structural node embeddings via diffusion wavelets," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 1320–1329.

[97] B. Rozemberczki and R. Sarkar, "Characteristic functions on graphs: Birds of a feather, from statistical descriptors to parametric models," 2020.

[98] G. Nikolentzos and M. Vazirgiannis, "Learning structural node representations using graph kernels," *IEEE Transactions on Knowledge and Data Engineering*, 2019.

[99] B. P. Anton Tsitsulin, John Palowitch and E. Müller, "Graph clustering with graph neural networks," 2020.

BIOGRAPHICAL STATEMENT

Fangqi Zhu was born in Chengdu, China in 1991. He received his B.S. and M.Sc degree from University of Electronic Science and Technology of China, China, in 2013 and 2016 respectively. He received his Ph.D. degree from The University of Texas at Arlington in 2020, all in Electrical Engineering. From 2014 to 2015, he was with the School of Communication and Information Engineering, University of Electronic Science and Technology of China as a Research Assistant in the Radar and Localization Lab. From July 2018 to August 2018, he was with the Lam Research as the full-time Data Science / Machine Learning Intern. From September 2019 to July 2020, he was with the Seagate US LLC as the full-time and part-time Data Science / Machine Learning Intern of the (Operations and Technology Advanced Analytics Group) OTAAG group. His current research interest is in the area of Deep Learning, Statistical Machine Learning, Signal Processing, Prognostic and Health Management in Semiconductor and Wireless Communication. He was the IEEE student member and served for IEEE Chengdu Section in 2014-2016 and served for IEEE Fort Worth Section in 2016-2018.