MATHEMATICAL AND STATISTICAL ANALYSIS OF NON-STATIONARY TIME SERIES DATA

HANG DU

A DISSERTATION SUBMITTED TO
THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

GRADUATE PROGRAM IN MATHEMATICS AND STATISTICS
YORK UNIVERSITY
TORONTO, ONTARIO

Sep 2023

Abstract

Non-stationary time series, with intrinsic properties constantly changing over time, present significant challenges for analysis in various scientific fields, particularly in biomedical signal analysis. This dissertation presents novel methodologies for analyzing and classifying highly noisy and non-stationary signals with applications to electroencephalograms (EEGs) and electrocardiograms (ECGs).

The first part of the dissertation focuses on a framework integrating pseudo-differential operators with convolutional neural networks (CNNs). We present their synergistic potential for signal classification from an innovative perspective.

Building on the fundamental concept of pseudo-differential operators, the dissertation further proposes a novel methodology that addresses the challenges of applying time-variant filters or transforms to non-stationary signals. This approach enables the neural network to learn a convolution kernel that changes over time or location, providing a refined strategy to effectively handle these dynamic signals.

This dissertation also introduces a hybrid convolutional neural network that in-

tegrates both complex-valued and real-valued components with the discrete Fourier transform (DFT) for EEG signal classification. This fusion of techniques significantly enhances the neural network's ability to utilize the phase information contained in the DFT, resulting in substantial accuracy improvements for EEG signal classification.

In the final part of this dissertation, we apply a conventional machine learning approach for the detection and localization of myocardial infarctions (MIs) in electrocardiograms (ECGs) and vectorcardiograms (VCGs), using the innovative features extracted from the geometrical and kinematic properties within VCGs. This boosts the accuracy and sensitivity of traditional MI detection.

Acknowledgements

Throughout this journey, I've been fortunate to have the support and guidance of many people. I'd like to take this opportunity to express my heartfelt gratitude towards them.

As I take a moment to express my deepest appreciation, the first person that comes to mind is my supervisor, Prof. Steven Wang. Throughout my PhD journey, his guidance has been instrumental in overcoming both academic hurdles and personal challenges. I still remember the time when he was brave enough to accompany me during my first freeway driving experience. His courage and trust in me not only helped boost my driving confidence but also showed his unwavering support in all aspects of my life. Besides, our enlightening discussions on the future of statistics and AI have been invaluable. Combined with his dedication to fostering my statistical thinking within my applied math program, they have played a significant role in broadening my academic horizons. In essence, without his support, completing my PhD would have been a formidable task.

Collaboration has been a cornerstone of my research, and in this light, I'd like to express my genuine appreciation to my research collaborators, Dr. Shahla Mollaha, Dr. Rebecca Pillai Riddell, Dr. He Jiang, and Dr. Xiaoxia Li. Working with such esteemed colleagues has provided invaluable insights and has significantly enriched

my research experience.

Next, I would like to extend my gratitude to Prof. Huaxiong Huang and Prof. Hongmei Zhu for their time and effort in serving on my dissertation committee. Their insights and feedback have been invaluable to my research.

I also wish to thank my wife, Qian Zhang, an exceptional researcher herself. Her unwavering encouragement and support throughout my studies will forever be etched in my memory.

Additionally, I owe a deep debt of gratitude to my parents, whose constant support and belief in me strengthened my determination to pursue a PhD.

Lastly, I want to thank all my friends who have accompanied me on this journey. You have added color to my PhD life, turning what could have been a monotonous journey into an enriching adventure. Thank you all for your companionship and unwavering support.

Table of Contents

List of Tables

List of Figures

1 Introduction

1.1 Non-stationary time series

A time series is a sequence of data points collected or recorded in a chronological order at regular intervals over a period of time [1]. It is a crucial type of structured data in a variety of fields, including economics, finance, weather forecasting, and engineering [2].

One of the key assumptions in many time series models is stationarity. A time series X_t is defined as strictly stationary if its joint probability distribution is invariant under time shifts. Formally, for any collection of time points $t_1, ..., t_n$ and any time delay h, the joint distribution of the vector $(X_{t_1}, ..., X_{t_n})$ is identical to that of the vector $(X_{t_1+h}, ..., X_{t_n+h})$ [2]. This means that the statistical properties of a process generating a time series do not change over time [3].

Often, a weaker form of stationarity, known as weak or wide-sense stationarity, is considered. A time series X_t is said to be weakly stationary [3] if:

- The mean $\mathbb{E}[X_t] = \mu$ is constant for all t.
- The variance $\text{Var}[X_t] = \gamma_0$ is constant for all t .
- The autocovariance $\text{Cov}[X_{t+h}, X_t] = \gamma_h$ depends only on the lag h and not on t itself.

However, many real-world time series data do not satisfy the stationarity assumption [4]. They may exhibit trends, seasonal patterns, or other changing structures over time, making them non-stationary. A non-stationary time series is one where the statistical properties do change over time. A common example is the random walk, defined as $X_t = X_{t-1} + \epsilon_t$, where ϵ_t is a white noise process. The mean, variance, and autocorrelation structure of this process all change over time, making it non-stationary [5]. Such non-stationary series pose unique challenges in time series analysis, often requiring specialized models and techniques [1].

1.2 Electroencephalograph (EEG) and Electrocardiograph (ECG)

An interesting area of application for non-stationary time series analysis is in the field of medical data analysis. In particular, electroencephalograph (EEG) and electrocardiograph (ECG) signals, which record the electrical activity of the brain and heart respectively, are prime examples of non-stationary time series. These signals often exhibit complex temporal dynamics due to the inherent physiological processes they capture, making their analysis both challenging and essential. Using the appropriate time series models and techniques for these non-stationary signals is vital in enhancing our understanding of neurological and cardiovascular conditions, leading to more accurate diagnoses and more effective treatments [6, 7].

1.2.1 EEG

Electroencephalography (EEG) is a non-invasive method for measuring the electrical activity of the brain, which provides valuable information about brain func-

tions, cognitive processes, and neural dynamics. The technique was first introduced by Hans Berger in 1929 when he recorded the first human EEG [8]. Since then, EEG has become a widely used tool in various fields, including clinical diagnosis, cognitive neuroscience, neuropsychology, and brain-computer interfaces [9, 10, 11].

EEG measures the voltage fluctuations that result from the synchronous activity of cortical neurons near the scalp surface. The electrical signals are recorded using electrodes placed on the scalp, which are connected to an amplifier and a recording device. The recorded signals, known as EEG traces, show the brain's electrical activity over time and can be analyzed to investigate the temporal and spatial dynamics of the underlying neural processes [12].

One of the key features of EEG is its high temporal resolution, which allows researchers to study the dynamics of cognitive processes on a millisecond timescale. This is particularly useful in examining the time course of event-related potentials (ERPs), which are specific patterns of neural activity associated with particular cognitive events, such as perception, attention, and memory [11].

Despite the excellent temporal resolution, EEG suffers from limited spatial resolution due to the nature of volume conduction, which causes the electrical activity of neurons to be spread out and mixed on the scalp surface. This makes it challenging to localize the sources of the recorded activity precisely. However, several advanced analysis techniques, such as source localization and connectivity analysis, have been developed to address this limitation [13, 14]. For example, inverse source localization methods, such as minimum norm estimates and beam forming, have been employed to estimate the neural generators of the EEG signals [15].

In addition to ERPs, researchers also investigate the oscillatory activity of the brain, which is considered a fundamental aspect of neural communication and processing [16]. EEG signals can be decomposed into different frequency bands, such

as delta (0.5-4 Hz), theta (4-8 Hz), alpha (8-12 Hz), beta (12-30 Hz), and gamma (30-100 Hz) bands, each of which is associated with specific cognitive functions and states of consciousness [17]. The analysis of these oscillatory activities allows for a deeper understanding of the complex interactions between different brain regions and their contributions to various cognitive processes.

EEG has also been employed in the development of brain-computer interfaces (BCIs), which are communication systems that allow users to control external devices using their brain activity [18]. BCIs have been applied in various areas, including assistive technology for people with severe motor disabilities, neurofeedback for the treatment of neurological and psychiatric disorders, and human-computer interaction for gaming and virtual reality [19, 20].

Moreover, the combination of EEG with other neuroimaging techniques, such as functional magnetic resonance imaging (fMRI), magnetoencephalography (MEG), and near-infrared spectroscopy (NIRS), has led to multimodal approaches that capitalize on the strengths of each method to obtain a more comprehensive understanding of brain function [21, 22]. For example, the high temporal resolution of EEG can be combined with the superior spatial resolution of fMRI to provide a more accurate picture of the neural processes underlying various cognitive functions.

Recent advancements in machine learning and artificial intelligence have greatly enhanced EEG research capabilities. Sophisticated algorithms, such as deep learning and support vector machines, have been applied to the analysis of EEG data, enabling the detection of more subtle patterns and improving the accuracy of classification tasks [23, 24]. This has had a significant impact on various applications, including BCIs, neurofeedback, and the early detection of neurological disorders, such as Alzheimer's disease and epilepsy [25].

1.2.2 ECG

Electrocardiography (ECG) is a fundamental tool in cardiovascular medicine that records the heart's electrical activity via electrodes placed on the skin, offering a valuable way to diagnose diverse heart conditions such as arrhythmias, myocardial infarctions, and structural heart diseases [26]. This method's genesis is found in the late 19th century when the first human electrocardiogram was documented by British physiologist, Augustus Waller. Notably, the field was significantly advanced by Dutch physiologist Willem Einthoven's invention of the string galvanometer and the successful recording of a human electrocardiogram [27, 28].

Vectorcardiography (VCG) is a technique closely related to ECG that records the heart's electrical activity using a three-dimensional approach. It represents the electrical forces generated by the heart as vectors, offering a more comprehensive view of the cardiac electrical activity. Although VCG has been largely replaced by modern 12-lead ECG systems in routine clinical practice, it still holds value in specific diagnostic scenarios, such as pinpointing the precise location of myocardial infarctions and assessing the cardiac axis [29].

In clinical practice, ECG and VCG have been essential tools for monitoring the efficacy of treatments and interventions in patients with cardiovascular diseases. They have also contributed to the optimization of personalized treatment plans and the identification of patients who may benefit from more aggressive or targeted therapies [30].

ECG recordings are obtained using a set of electrodes arranged in specific configurations, known as leads. A standard 12-lead ECG system comprises 10 electrodes placed on the limbs and chest, providing a comprehensive view of the heart's electrical activity from various angles.

In contrast, VCG employs a set of three orthogonal leads (X, Y, and Z) to obtain a three-dimensional representation of the heart's electrical activity. This technique enables a more detailed analysis of the spatial aspects of cardiac electrical forces, particularly when evaluating the cardiac axis and identifying the location of myocardial infarctions [29].

Over the years, both ECG and VCG techniques have progressed, with the development of new lead systems [31], recording methods, and analysis techniques [32]. These advances have improved the diagnostic capabilities of electrocardiography and vectorcardiography, supplying valuable information for clinical decision-making and management of cardiac conditions [33].

As technology continues to advance, new frontiers in electrocardiography and vectorcardiography are emerging, such as machine learning and artificial intelligence-based ECG interpretation [34]. These innovations have the potential to further improve the diagnostic accuracy, clinical utility, and accessibility of electrocardiography and vectorcardiography in the coming years.

1.3 Main Contributions and Chapter Preview

Chapter 2 explores the theory of complex-valued neural networks (CVNNs) and pseudo differential operators. We detail CVNN's general form and backpropagation, and introduce a unique architecture that uses varied kernels inspired by pseudo-differential operators, in contrast to traditional CNNs.

Chapter 3 addresses applying time-variant filters in biomedical EEG signal processing. Diverging from conventional methods that use Linear Time-Invariant (LTI) filters, we propose an approach using time-varying filters governed by a second-order ODE, where the EEG signal is the driving function. This approach promises en-

hanced EEG denoising and potentially improves classification accuracy.

In Chapter 4, we present an advanced EEG classification algorithm that utilizes both amplitude and phase data using complex-valued CNNs. By integrating real and complex-valued neural structures, our method achieves higher accuracy with fewer parameters, outperforming traditional models on benchmark datasets. This chapter includes published work with Dr. Rebecca Pillai Riddell and Dr. Xiaogang Wang [35].

Chapter 5 focuses on improving myocardial infarction (MI) detection using the Frank 3-lead VCG instead of the typical 12-lead ECG. By introducing new features based on VCG's kinematic and geometric properties and combining them with prior research, we enhance MI classification and localization. Our results show significant improvements in accuracy and sensitivity, demonstrating our method's adaptability on VCG signals from 12-lead ECG transformations.

2 A New Neural Network Structre based on Pseudo-Differential Operator

2.1 Introduction

Convolutional neural networks (CNNs) have played a significant role in the advancement of various fields, including image classification [36, 37, 38], object detection [39, 40, 41], semantic segmentation [42, 43, 44], text classification [45, 46], and medical image analysis [47, 48]. With the evolution of these networks, complex-valued neural networks (CVNNs) have emerged, further broadening the scope of applications and adding a new dimension to the field [49, 50, 51, 52].

In the context of signal processing and imaging, Fourier transform and pseudo-differential operators are widely employed [53, 54, 55], leading to the need for handling complex numbers for inputs and parameters. Simultaneously, the challenge imposed by the uncertainty principle [56], which constrains achieving a high frequency resolution and high time resolution concurrently, is also addressed by CVNNs.

The traditional approach to CVNNs treats the real and imaginary parts of network parameters separately, drawing on the phase and amplitude information of the data [57]. However, the evolving need for finer granularity of information, especially concerning the locations of high and low frequencies, has necessitated the development of more sophisticated methods.

Building on this background, our study introduces a novel neural network architecture that shares similarities with CNNs but incorporates a significant deviation: Instead of applying the same kernel across all locations, as is the case in CNNs, our proposed framework employs different kernels at different locations, based on pseudo-differential operators. This framework, rooted in the concept of embedding and cropping operators, brings a new level of adaptability to the network design. Under our framework, we extend the input and output layers, convolution layer, activation function, and pooling layer to operate within the complex domain. Furthermore, we propose a complex backpropagation algorithm, derived using Wirtinger calculus [58, 52], that applies an adaptive gradient descent method tailored for our model architecture.

The rest of this chapter is structured as follows: In Section 2, we present the methodology and introduce the concepts of embedding and cropping operators, which are essential for subsequent analysis. Section 3 explores the Wirtinger derivative and provides a detailed description of its specific form for operators. Section 4 focuses on the introduction of the pooling layer along with definitions for head and tail mappings. Following this, in Section 5, we derive the adaptive gradient descent method within our proposed framework. Section 6 explores the pseudo-differential operator and its relationship with the convolution operator. Results of testing using our approach with one-dimensional and two-dimensional datasets, is presented in Section 7. Finally, in Section 8, we present an extensive discussion of our findings and their implications.

2.2 Methodology

Let $\mathbb{Z}_N = \{0, 1, \ldots, N-1\}$ be the additive group, where N is a positive integer greater than or equal to 2. and the group law is addition modulo N. Let $L^2(\mathbb{Z}_N)$ be the set of all functions $z : \mathbb{Z}_N \to \mathbb{C}$. A function $z \in L^2(\mathbb{Z}_N)$ can be completely specified by its N components $z(0), z(1), \ldots, z(N-1)$. Let $z, w \in L^2(\mathbb{Z}_N)$. In fact, $L^2(\mathbb{Z}_N)$ is a Hilbert space where the inner product is defined

$$(z, w)_{L^2(\mathbb{Z}_N)} = \sum_{n \in \mathbb{Z}_N} z(n)\overline{w(n)}, \quad z, w \in L^2(\mathbb{Z}_N).$$

Hence, for any $z, \in L^2(\mathbb{Z}_N)$,

$$\|z\|_{L^2(\mathbb{Z}_N)} = \left\{ \sum_{n \in \mathbb{Z}_N} |z(n)|^2 \right\}^{1/2}.$$

Similarly, we can define $L^2(\mathbb{Z}_n \times \mathbb{Z}_l)$, be the set of all functions $z : \mathbb{Z}_n \times \mathbb{Z}_l \to \mathbb{C}$.

Remark 2.2.1 *We can represent $L^2(\mathbb{Z}_n)$ by the set of all n-tuples with complex entries, i.e., for every $z \in L^2(\mathbb{Z}_n)$, we have*

$$z = \begin{bmatrix} z(0) \\ z(1) \\ \vdots \\ z(n-1) \end{bmatrix} \in \mathbb{C}^n$$

and we can represent $L^2(\mathbb{Z}_n \times \mathbb{Z}_l)$, by the set of all $n \times l$ matrices with complex entries,

i.e., every $z \in L^2(\mathbb{Z}_n \times \mathbb{Z}_l)$ *can be represented by*

$$z = \begin{bmatrix} z(0,0) & z(0,1) & \cdots & z(0,l-1) \\ z(1,0) & z(1,1) & \cdots & z(1,l-1) \\ \vdots & \vdots & \vdots & \vdots \\ z(n-1,0) & z(n-1,1) & \cdots & z(n-1,l-1) \end{bmatrix} \in \mathbb{C}^{n \times l}$$

and its norm given by

$$\|z\|_{L^2(\mathbb{Z}_n \times \mathbb{Z}_l)} = \left\{ \sum_{s \in \mathbb{Z}_n} \sum_{t \in \mathbb{Z}_l} z(s,t) \overline{z(s,t)} \right\}^{1/2}$$

which is also known as Frobenious norm and can be written as

$$\|z\|_{L^2(\mathbb{Z}_n \times \mathbb{Z}_l)} = \left\{ \mathrm{Tr}(zz^*) \right\}^{1/2},$$

where z^* *is the conjugate transpose of the matrix* z*.*

Throughout the chapter, sometimes we will use the matrix representations of these spaces. Then for all $z \in L^2(\mathbb{Z}_n \times \mathbb{Z}_l)$, z^T is the transpose of the matrix z and therefore z^T can be considered as a function in $L^2(\mathbb{Z}_l \times \mathbb{Z}_n)$.

Definition 2.2.1 *Let* n, p, l, q *be positive integers such that* $n \gg p$ *and* $l \gg q$*. For any* $k \in \mathbb{Z}_{n-p+1}$ *and* $m \in Z_{l-q+1}$*,we define the cropping operator* $K_{k,m} : L^2(\mathbb{Z}_n \times \mathbb{Z}_l) \to L^2(\mathbb{Z}_p \times \mathbb{Z}_q)$ *by*

$$(K_{k,m} z)(s,t) = z(s+k, t+m), \quad (s,t) \in \mathbb{Z}_p \times \mathbb{Z}_q$$

where $z \in L^2(\mathbb{Z}_n \times \mathbb{Z}_l)$*.*

In fact the cropping operator $K_{k,m}$, takes an $n \times l$ matrix z and returns a sub-matrix of z of dimension $p \times q$.

Proposition 2.2.1 *Let n, p, l, q be positive integers such that $n \gg p$ and $l \gg q$. For any $k \in \mathbb{Z}_{n-p+1}$ and $m \in Z_{l-q+1}$, $K_{k,m} : L^2(\mathbb{Z}_n \times \mathbb{Z}_l) \to L^2(\mathbb{Z}_p \times \mathbb{Z}_q)$ is a bounded linear operator. Moreover,*

$$\|K_{k,m}\|_{B(L^2(\mathbb{Z}_n \times \mathbb{Z}_l), L^2(\mathbb{Z}_p \times \mathbb{Z}_q))} \leq 1,$$

where $B\left(L^2(\mathbb{Z}_n \times \mathbb{Z}_l), L^2(\mathbb{Z}_p \times \mathbb{Z}_q)\right)$ is the space of all bounded linear operators from $L^2(\mathbb{Z}_n \times \mathbb{Z}_l)$ to $L^2(\mathbb{Z}_p \times \mathbb{Z}_q)$.

Proof It is easy to check that $K_{k,m}$ is a linear operator. Let $z \in L^2(\mathbb{Z}_n \times \mathbb{Z}_l)$. Then

$$\begin{aligned} \|K_{k,m} z\|^2_{L^2(\mathbb{Z}_p \times \mathbb{Z}_q)} &= \sum_{s \in \mathbb{Z}_p} \sum_{t \in \mathbb{Z}_q} \left|\left(K_{k,m} z\right)(s,t)\right|^2 \\ &= \sum_{s \in \mathbb{Z}_p} \sum_{t \in \mathbb{Z}_q} \left|z(s+k, t+m)\right|^2 . \end{aligned}$$

By the change of variables from s to $\tilde{s} = s + k$ and from t to $\tilde{t} = t + m$, we get

$$\begin{aligned} &\|K_{k,m} z\|^2_{L^2(\mathbb{Z}_p \times \mathbb{Z}_q)} \\ &= \sum_{\tilde{s}=k}^{p-1+k} \sum_{\tilde{t}=m}^{q-1+m} \left|z(\tilde{s}, \tilde{t})\right|^2 \\ &\leq \|z\|_{L^2(\mathbb{Z}_n \times \mathbb{Z}_l)}. \end{aligned}$$

□

Now, we need to define the embedding operator, in order to find the adjoint of the cropping operators.

Definition 2.2.2 *Let n, p, l, q be positive integers such that $n \gg p$ and $l \gg q$. For any $k \in \mathbb{Z}_{n-p+1}$ and $m \in Z_{l-q+1}$, we define the embedding operator $E_{k,m} : L^2(\mathbb{Z}_p \times \mathbb{Z}_q) \to L^2(\mathbb{Z}_n \times \mathbb{Z}_l)$ by*

$$(E_{k,m}w)(s,t)$$
$$= \begin{cases} w(s-k,t-m), & \text{if } k \le s \le k+p-1 \text{ and } m \le t \le m+q-1, \\ 0, & \text{if } s \in \mathbb{Z}_n \setminus \{k \le s \le k+p-1\} \text{ or } t \in \mathbb{Z}_l \setminus \{m \le t \le m+q-1\} \end{cases}$$

We can easily see that when we apply the embedding operator to a matrix w, it returns a matrix which has w as a sub-matrix and the other entries are zero.

The following theorem, gives us the adjoint of the cropping operators.

Theorem 2.2.3 *Let n, p, l, q be positive integers such that $n \gg p$ and $l \gg q$. For any $k \in \mathbb{Z}_{n-p+1}$ and $m \in Z_{l-q+1}$, the adjoint of the cropping operator $K_{k,m} : L^2(\mathbb{Z}_n \times \mathbb{Z}_l) \to L^2(\mathbb{Z}_p \times \mathbb{Z}_q)$ is the embedding operator $E_{k,m}$.*

Proof Let $z \in L^2(\mathbb{Z}_n \times \mathbb{Z}_l)$ and $w \in L^2(\mathbb{Z}_p \times \mathbb{Z}_q)$. The

$$\begin{aligned} (K_{k,m}z, w)_{L^2(\mathbb{Z}_p \times \mathbb{Z}_q)} &= \sum_{s \in \mathbb{Z}_p} \sum_{t \in \mathbb{Z}_q} (K_{k,m}z)(s,t)\overline{w(s,t)} \\ &= \sum_{s \in \mathbb{Z}_p} \sum_{t \in \mathbb{Z}_q} z(s+k, t+m)\overline{w(s,t)} \end{aligned}$$

By the change of variables from (s,t) to $(\tilde{s}, \tilde{t})$ where

$$\begin{cases} \tilde{s} = s + k, \\ \tilde{t} = t + m, \end{cases}$$

we get

$$\begin{aligned} (K_{k,m}z, w)_{L^2(\mathbb{Z}_p \times \mathbb{Z}_q)} &= \sum_{\tilde{s}=k}^{k+p-1} \sum_{\tilde{t}=m}^{q+m-1} z(\tilde{s}, \tilde{t})\overline{w(\tilde{s}-k, \tilde{t}-m)} \\ &= \sum_{\tilde{s}=0}^{n-1} \sum_{\tilde{t}=0}^{l-1} z(\tilde{s}, \tilde{t})\overline{(E_{k,m}w)(\tilde{s}, \tilde{t})} \\ &= (z, E_{k.m}w)_{L^2(\mathbb{Z}_n \times \mathbb{Z}_l)}. \end{aligned}$$

□

We first define the convolutional operator with stride $\mathfrak{s} = 1$.

Definition 2.2.4 *Let n, p, l, q be positive integers such that $n \gg p$ and $l \gg q$. We define the convolutional operator $C : L^2(\mathbb{Z}_n \times \mathbb{Z}_l) \times L^2(\mathbb{Z}_p \times \mathbb{Z}_q) \to L^2(\mathbb{Z}_{n-p+1} \times \mathbb{Z}_{l-q+1})$ by*

$$C(z, w)(k, m) = \left(w, \overline{K_{k,m} z}\right)_{L^2(\mathbb{Z}_p \times \mathbb{Z}_q)}, \quad k \in \mathbb{Z}_{n-p+1},\ m \in Z_{l-q+1},$$

for any $w \in L^2(\mathbb{Z}_p \times \mathbb{Z}_q)$ and $z \in L^2(\mathbb{Z}_n \times \mathbb{Z}_l)$. In fact,

$$C(z, w)(k, m) = \sum_{s \in \mathbb{Z}_p} \sum_{t \in \mathbb{Z}_q} w(s, t) z(s + k, t + m).$$

So, by Theorem 2.2.3,

$$C(z, w)(k, m) = (z, E_{k,m} \overline{w})_{L^2(\mathbb{Z}_n \times \mathbb{Z}_l)} \tag{2.1}$$

Definition 2.2.5 *Let n, p, l, q be positive integers such that $n \gg p$ and $l \gg q$. Let $\mathfrak{s}$ be a positive integer. Let $\hat{n} = \lfloor \frac{n-p}{\mathfrak{s}} \rfloor + 1$ and $\hat{l} = \lfloor \frac{l-q}{\mathfrak{s}} \rfloor + 1$. For any $k \in \mathbb{Z}_{\hat{n}}$ and $m \in Z_{\hat{l}}$, we define the cropping operator with stride $\mathfrak{s}$ by $K^{\mathfrak{s}}_{k,m} : L^2(\mathbb{Z}_n \times \mathbb{Z}_l) \to L^2(\mathbb{Z}_p \times \mathbb{Z}_q)$ by*

$$(K^{\mathfrak{s}}_{k,m} z)(s, t) = z(s + k\mathfrak{s}, t + m\mathfrak{s}), \quad (s, t) \in \mathbb{Z}_p \times \mathbb{Z}_q$$

where $z \in L^2(\mathbb{Z}_n \times \mathbb{Z}_l)$.

Note that if we let $\mathfrak{s} = 1$, we get back the cropping operator $K_{k,m}$ in Definition 2.2.1.

Similar to Proposition 2.2.1, we have the following proposition about the boundedness of $K^{\mathfrak{s}}_{k,m}$.

Proposition 2.2.2 *Let $\mathfrak{s}, n, p, l, q$ be positive integers such that $n \gg p$ and $l \gg q$. Let $\hat{n} = \lfloor \frac{n-p}{\mathfrak{s}} \rfloor + 1$ and $\hat{l} = \lfloor \frac{l-q}{\mathfrak{s}} \rfloor + 1$. For any $k \in \mathbb{Z}_{\hat{n}}$ and $m \in Z_{\hat{l}}$, $K^{\mathfrak{s}}_{k,m} : L^2(\mathbb{Z}_n \times \mathbb{Z}_l) \to L^2(\mathbb{Z}_p \times \mathbb{Z}_q)$ is a bounded linear operator. Moreover,*

$$\left\| K^{\mathfrak{s}}_{k,m} \right\|_{B(L^2(\mathbb{Z}_n \times \mathbb{Z}_l), L^2(\mathbb{Z}_p \times \mathbb{Z}_q))} \leq 1,$$

where $B\left(L^2(\mathbb{Z}_n \times \mathbb{Z}_l), L^2(\mathbb{Z}_p \times \mathbb{Z}_q)\right)$ is the space of all bounded linear operators from $L^2(\mathbb{Z}_n \times \mathbb{Z}_l)$ to $L^2(\mathbb{Z}_p \times \mathbb{Z}_q)$.

Proof It is easy to check that $K^{\mathfrak{s}}_{k,m}$ is a linear operator. Let $z \in L^2(\mathbb{Z}_n \times \mathbb{Z}_l)$. Then

$$\begin{aligned} \left\| K^{\mathfrak{s}}_{k,m} z \right\|^2_{L^2(\mathbb{Z}_p \times \mathbb{Z}_q)} &= \sum_{s \in \mathbb{Z}_p} \sum_{t \in \mathbb{Z}_q} \left| \left(K^{\mathfrak{s}}_{k,m} z \right)(s,t) \right|^2 \\ &= \sum_{s \in \mathbb{Z}_p} \sum_{t \in \mathbb{Z}_q} \left| z(s + k\mathfrak{s}, t + m\mathfrak{s}) \right|^2 . \end{aligned}$$

By the change of variables from s to $\tilde{s} = s + k\mathfrak{s}$ and from t to $\tilde{t} = t + m\mathfrak{s}$, we get

$$\begin{aligned} & \left\| K_{k,m} z \right\|^2_{L^2(\mathbb{Z}_p \times \mathbb{Z}_q)} \\ &= \sum_{\tilde{s} = k\mathfrak{s}}^{p-1+k\mathfrak{s}} \sum_{\tilde{t} = m\mathfrak{s}}^{q-1+m\mathfrak{s}} \left| z(\tilde{s}, \tilde{t}) \right|^2 \\ &\leq \left\| z \right\|_{L^2(\mathbb{Z}_n \times \mathbb{Z}_l)}. \end{aligned}$$

□

Now, we need to define the embedding operator, in order to find the adjoint of the cropping operators with stride.

Definition 2.2.6 *Let $\mathfrak{s}, n, p, l, q$ be positive integers such that $n \gg p$ and $l \gg q$. Let $\hat{n} = \lfloor \frac{n-p}{\mathfrak{s}} \rfloor + 1$ and $\hat{l} = \lfloor \frac{l-q}{\mathfrak{s}} \rfloor + 1$. For any $k \in \mathbb{Z}_{\hat{n}}$ and $m \in Z_{\hat{l}}$, we define the embedding operator $E^{\mathfrak{s}}_{k,m} : L^2(\mathbb{Z}_p \times \mathbb{Z}_q) \to L^2(\mathbb{Z}_n \times \mathbb{Z}_l)$ by*

$$\left(E^{\mathfrak{s}}_{k,m}w\right)(s,t) =$$

$$\begin{cases} w(s-k\mathfrak{s}, t-m\mathfrak{s}), & \text{if } k\mathfrak{s} \le s \le k\mathfrak{s}+p-1 \text{ and } m\mathfrak{s} \le t \le m\mathfrak{s}+q-1, \\ 0, & \text{if } s \in \mathbb{Z}_n \setminus \{k\mathfrak{s} \le s \le k\mathfrak{s}+p-1\} \text{ or } t \in \mathbb{Z}_l \setminus \{m\mathfrak{s} \le t \le m\mathfrak{s}+q-1\}. \end{cases}$$

We can easily see that when we apply the embedding operator to a matrix w, it returns a matrix which has w as a sub-matrix and the other entries are zero. It is easy to check that $E^{\mathfrak{s}}_{k,m} = E_{k\mathfrak{s},m\mathfrak{s}}$ and $K^{\mathfrak{s}}_{k,m} = K_{k\mathfrak{s},m\mathfrak{s}}$.

The following theorem, gives us the adjoint of the cropping operators.

Theorem 2.2.7 *Let $\mathfrak{s}, n, p, l, q$ be positive integers such that $n \gg p$ and $l \gg q$. Let $\hat{n} = \lfloor \frac{n-p}{\mathfrak{s}} \rfloor + 1$ and $\hat{l} = \lfloor \frac{l-q}{\mathfrak{s}} \rfloor + 1$. For any $k \in \mathbb{Z}_{\hat{n}}$ and $m \in Z_{\hat{l}}$, the adjoint of the cropping operator $K^{\mathfrak{s}}_{k,m} : L^2(\mathbb{Z}_n \times \mathbb{Z}_l) \to L^2(\mathbb{Z}_p \times \mathbb{Z}_q)$ is the embedding operator $E^{\mathfrak{s}}_{k,m}$.*

Proof Let $z \in L^2(\mathbb{Z}_n \times \mathbb{Z}_l)$ and $w \in L^2(\mathbb{Z}_p \times \mathbb{Z}_q)$. The

$$\begin{aligned} \left(K^{\mathfrak{s}}_{k,m}z, w\right)_{L^2(\mathbb{Z}_p \times \mathbb{Z}_q)} &= \sum_{s \in \mathbb{Z}_p} \sum_{t \in \mathbb{Z}_q} \left(K^{\mathfrak{s}}_{k,m}z\right)(s,t)\overline{w(s,t)} \\ &= \sum_{s \in \mathbb{Z}_p} \sum_{t \in \mathbb{Z}_q} z(s+k\mathfrak{s}, t+m\mathfrak{s})\overline{w(s,t)} \end{aligned}$$

By the change of variables from (s,t) to $(\tilde{s},\tilde{t})$ where

$$\begin{cases} \tilde{s} = s + k\mathfrak{s}, \\ \tilde{t} = t + m\mathfrak{s}, \end{cases}$$

we get

$$\begin{aligned}\left(K_{k,m}^{\mathfrak{s}}z,w\right)_{L^2(\mathbb{Z}_p\times\mathbb{Z}_q)} &= \sum_{\tilde{s}=k\mathfrak{s}}^{k\mathfrak{s}+p-1}\sum_{\tilde{t}=m\mathfrak{s}}^{q+m\mathfrak{s}-1} z(\tilde{s},\tilde{t})\overline{w(\tilde{s}-k\mathfrak{s},\tilde{t}-m\mathfrak{s})}\\ &= \sum_{\tilde{s}=0}^{n-1}\sum_{\tilde{t}=0}^{l-1} z(\tilde{s},\tilde{t})\overline{\left(E_{k,m}^{\mathfrak{s}}w\right)(\tilde{s},\tilde{t})}\\ &= \left(z,E_{k.m}^{\mathfrak{s}}w\right)_{L^2(\mathbb{Z}_n\times\mathbb{Z}_l)}.\end{aligned}$$

□

Definition 2.2.8 *Let $\mathfrak{s},n,p,l,q$ be positive integers such that $n\gg p$ and $l\gg q$. Let $\hat{n}=\lfloor\frac{n-p}{\mathfrak{s}}\rfloor+1$ and $\hat{l}=\lfloor\frac{l-q}{\mathfrak{s}}\rfloor+1$. We define the convolutional operator $C_{\mathfrak{s}}:L^2(\mathbb{Z}_n\times\mathbb{Z}_l)\times L^2(\mathbb{Z}_p\times\mathbb{Z}_q)\to L^2(\mathbb{Z}_{\hat{n}}\times\mathbb{Z}_{\hat{l}})$ by*

$$C_{\mathfrak{s}}(z,w)(k,m)=\left(w,\overline{K_{k,m}^{\mathfrak{s}}z}\right)_{L^2(\mathbb{Z}_p\times\mathbb{Z}_q)},\quad k\in\mathbb{Z}_{\hat{n}},\ m\in Z_{\hat{l}},$$

for any $z\in L^2(\mathbb{Z}_n\times\mathbb{Z}_l)$ and $w\in L^2(\mathbb{Z}_p\times\mathbb{Z}_q)$. In fact,

$$C_{\mathfrak{s}}(z,w)(k,m)=\sum_{s\in\mathbb{Z}_p}\sum_{t\in\mathbb{Z}_q} w(s,t)z(s+k\mathfrak{s},t+m\mathfrak{s}). \tag{2.2}$$

So, by Theorem 2.2.7,

$$C_{\mathfrak{s}}(z,w)(k,m)=\left(z,E_{k,m}^{\mathfrak{s}}\overline{w}\right)_{L^2(\mathbb{Z}_n\times\mathbb{Z}_l)}. \tag{2.3}$$

Theorem 2.2.9 *Let $\mathfrak{s},n,p,l,q$ be positive integers such that $n\gg p$ and $l\gg q$. Let $\hat{n}=\lfloor\frac{n-p}{\mathfrak{s}}\rfloor+1$ and $\hat{l}=\lfloor\frac{l-q}{\mathfrak{s}}\rfloor+1$. Let $z\in L^2(\mathbb{Z}_n\times\mathbb{Z}_l)$. Consider*

$$C_{\mathfrak{s}}(z,\cdot):L^2(\mathbb{Z}_p\times\mathbb{Z}_q)\to L^2(\mathbb{Z}_{\hat{n}}\times\mathbb{Z}_{\hat{l}})$$

defined by

$$\left(C_{\mathfrak{s}}(z,\cdot)\,w\right)(k,m)=C_{\mathfrak{s}}(w,z)(k,m).$$

Then the adjoint of $C_{\mathfrak{s}}(\cdot, z)$ *is*

$$(C_{\mathfrak{s}}(z,\cdot))^* : L^2(\mathbb{Z}_{\hat{n}} \times \mathbb{Z}_{\hat{l}}) \to L^2(\mathbb{Z}_p \times \mathbb{Z}_q),$$

where for all $\tilde{z} \in L^2(\mathbb{Z}_{\hat{n}} \times \mathbb{Z}_{\hat{l}})$

$$(C_{\mathfrak{s}}(z,\cdot))^* \tilde{z} = \sum_{k=0}^{\tilde{n}-1} \sum_{m=0}^{\tilde{l}-1} \tilde{z}(k,m) \overline{K_{k,m}^{\mathfrak{s}} z},$$

Proof Let $w \in L^2(\mathbb{Z}_p \times \mathbb{Z}_q)$ and $\tilde{z} \in L^2(\mathbb{Z}_{\hat{n}} \times \mathbb{Z}_{\hat{l}})$. Then

$$\begin{aligned}
(C_{\mathfrak{s}}(z,w), \tilde{z})_{L^2(\mathbb{Z}_{\hat{n}} \times \mathbb{Z}_{\hat{l}})} &= \sum_{k=0}^{\hat{n}-1} \sum_{m=0}^{\hat{l}-1} C_{\mathfrak{s}}(z,w)(k,m) \overline{\tilde{z}(k,m)} \\
&= \sum_{k=0}^{\hat{n}-1} \sum_{m=0}^{\hat{l}-1} \left(w, \overline{K_{k,m}^{\mathfrak{s}} z}\right)_{L^2(\mathbb{Z}_p \times \mathbb{Z}_q)} \overline{\tilde{z}(k,m)} \\
&= \left(w, \sum_{k=0}^{\hat{n}-1} \sum_{m=0}^{\hat{l}-1} \tilde{z}(k,m) \overline{K_{k,m}^{\mathfrak{s}} z} \right)_{L^2(\mathbb{Z}_p \times \mathbb{Z}_q)}.
\end{aligned}$$

Hence,

$$(C_{\mathfrak{s}}(z,\cdot))^* \tilde{z} = \sum_{k=0}^{n-1} \sum_{m=0}^{l-1} \tilde{z}(k,m) \overline{K_{k,m}^{\mathfrak{s}} z}.$$

□

Theorem 2.2.10 *Let* $\mathfrak{s}, n, p, l, q$ *be positive integers such that* $n \gg p$ *and* $l \gg q$. *Let* $\hat{n} = \lfloor \frac{n-p}{\mathfrak{s}} \rfloor + 1$ *and* $\hat{l} = \lfloor \frac{l-q}{\mathfrak{s}} \rfloor + 1$. *Let* $w \in L^2(\mathbb{Z}_p \times \mathbb{Z}_q)$. *Consider*

$$C_{\mathfrak{s}}(\cdot, w) : L^2(\mathbb{Z}_n \times \mathbb{Z}_l) \to L^2(\mathbb{Z}_{\hat{n}} \times \mathbb{Z}_{\hat{l}})$$

defined by

$$(C_{\mathfrak{s}}(\cdot, w)\, z)(k,m) = C_{\mathfrak{s}}(z,w)(k,m).$$

Then the adjoint of $C_{\mathfrak{s}}(w,\cdot)$ *is*

$$(C_{\mathfrak{s}}(\cdot, w))^* : L^2(\mathbb{Z}_{\hat{n}} \times \mathbb{Z}_{\hat{l}}) \to L^2(\mathbb{Z}_n \times \mathbb{Z}_l),$$

where for all $\tilde{z} \in L^2(\mathbb{Z}_{\hat{n}} \times \mathbb{Z}_{\hat{l}})$

$$\left(C_{\mathfrak{s}}(\cdot, w)\right)^* \tilde{z} = \sum_{k=0}^{\hat{n}-1} \sum_{m=0}^{\hat{l}-1} \tilde{z}(k,m) E_{k,m}^{\mathfrak{s}} \overline{w},$$

Proof Let $z \in L^2(\mathbb{Z}_n \times \mathbb{Z}_l)$ and $\tilde{z} \in L^2(\mathbb{Z}_{\hat{n}} \times \mathbb{Z}_{\hat{l}})$. Then by (2.3)

$$\begin{aligned}
\left(C_{\mathfrak{s}}(z,w), \tilde{z}\right)_{L^2(\mathbb{Z}_{\hat{n}} \times \mathbb{Z}_{\hat{l}})} &= \sum_{k=0}^{\hat{n}-1} \sum_{m=0}^{\hat{l}-1} \left(C_{\mathfrak{s}}\left(z,w\right)\right)(k,m) \overline{\tilde{z}(k,m)} \\
&= \sum_{k=0}^{\hat{n}-1} \sum_{m=0}^{\hat{l}-1} \left(z, E_{k,m}^{\mathfrak{s}} \overline{w}\right)_{L^2(\mathbb{Z}_n \times \mathbb{Z}_l)} \overline{\tilde{z}(k,m)} \\
&= \left(z, \sum_{k=0}^{\hat{n}-1} \sum_{m=0}^{\hat{l}-1} \tilde{z}(k,m) E_{k,m}^{\mathfrak{s}} \overline{w} \right)_{L^2(\mathbb{Z}_n \times \mathbb{Z}_l)}.
\end{aligned}$$

Hence,

$$\left(C_{\mathfrak{s}}(\cdot, w)\right)^* \tilde{z} = \sum_{k=0}^{\hat{n}-1} \sum_{m=0}^{\hat{l}-1} \tilde{z}(k,m) E_{k,m}^{\mathfrak{s}} \overline{w}.$$

□

2.3 Wirtinger Derivatives

In the complex case, the loss function is real valued with complex weights. Such a function is not differentiable everywhere, and its steepest descent direction cannot be calculated using the gradient. To that end we use the Wirtinger derivatives. Let $f : \mathbb{C} \to \mathbb{C}$ be a function defined by

$$f(z) = u(x,y) + iv(x,y)$$

where $i = \sqrt{-1}$, and $z = x + iy$, for $x, y \in \mathbb{R}$ and $u(x,y)$ and $v(x,y)$ are in $\mathbb{R}$. By using

$$x = \frac{z + \bar{z}}{2}, \quad y = \frac{z - \bar{z}}{2i}$$

the function f can be rewritten as a bivariate function of z and $\bar{z}$. If $u(x, y)$ and $v(x, y)$ are differentiable with respect to real-valued variables x and y, then we say that f is of class C^1 and we can apply Wirtinger calculus to compute $\frac{\partial f}{\partial z}$ and $\frac{\partial f}{\partial \bar{z}}$ by treating z and $\bar{z}$ as two independent variables, so

$$\frac{\partial z}{\partial \bar{z}} = 0, \quad \frac{\partial \bar{z}}{\partial z} = 0.$$

In this way, when taking the partial derivative with respect to z, we consider $\bar{z}$ as a constant and calculate $\frac{\partial f}{\partial z}$. Similarly, $\frac{\partial f}{\partial \bar{z}}$ is derived by considering z as a constant and taking the partial derivative with respect to $\bar{z}$. This makes it possible for $\frac{\partial f}{\partial z}$ and $\frac{\partial f}{\partial \bar{z}}$ to be derived in the same manner as for the real-valued case [59]. Hence by applying the chain rule, we obtain

$$\begin{aligned} \frac{\partial f}{\partial z} &= \frac{1}{2}\left[\frac{\partial f}{\partial x} - i\frac{\partial f}{\partial y}\right] \\ \frac{\partial f}{\partial \bar{z}} &= \frac{1}{2}\left[\frac{\partial f}{\partial x} + i\frac{\partial f}{\partial y}\right] \end{aligned} \tag{3.4}$$

The operators $\frac{\partial}{\partial z}$ and $\frac{\partial}{\partial \bar{z}}$ defined in (3.4) are called Cauchy-Riemann operators. We note that $\frac{\partial}{\partial z}$ and $\frac{\partial}{\partial \bar{z}}$ can be applied to arbitrary differentiable (not necessarily holomorphic) functions. We can see easiliy,

$$\overline{\frac{\partial f}{\partial z}} = \frac{\partial \bar{f}}{\partial \bar{z}}$$

Hence, if f is a real-valued function, then

$$\overline{\frac{\partial f}{\partial z}} = \frac{\partial f}{\partial \bar{z}}.$$

Moreover, $f : \mathbb{C} \to \mathbb{C}$ is a holomorphic function if and only if

$$\frac{\partial f}{\partial \bar{z}} = 0.$$

Let $\Omega \subset \mathbb{C}^n$. Let $C^1(\Omega)$ be the class of all functions $f : \Omega \to \mathbb{C}$ such that for all $1 \leq j \leq n$, $\frac{\partial f}{\partial x_j}$ and $\frac{\partial f}{\partial y_j}$ exist and are continuous in Ω.

We say that the operator $T : L^2(\mathbb{Z}_m) \to \mathbb{C}$ is of class C^1, if for all $1 \leq j \leq m$, $\frac{\partial T}{\partial x_j}$ and $\frac{\partial T}{\partial y_j}$ exists and are continuous. Similarly, we say the operator $T : L^2(\mathbb{Z}_m) \to L^2(\mathbb{Z}_n)$ is of class C^1, if for all $1 \leq j \leq m$ and $1 \leq k \leq n$, $\frac{\partial T_k}{\partial x_j}$ and $\frac{\partial T_k}{\partial y_j}$ exists and are continuous where

$$T_k(f) = (T(f))(k), \quad f \in L^2(\mathbb{Z}_m).$$

Let $z = (z_1, z_2, \ldots, z_n) \in \mathbb{C}^n$ where $z_j = x_j + iy_j$, $1 \leq j \leq n$, then we define the Cauchy-Riemann operators

$$\begin{aligned} D_j &= \frac{\partial}{\partial z_j} = \frac{1}{2}(\frac{\partial}{\partial x_j} - i\frac{\partial}{\partial y_j}) \\ \overline{D_j} &= \frac{\partial}{\partial \bar{z}_j} = \frac{1}{2}(\frac{\partial}{\partial x_j} + i\frac{\partial}{\partial y_j}) \end{aligned} \tag{3.5}$$

Let $T : L^2(\mathbb{Z}_n) \to \mathbb{C}$ be of class C^1. Then we define the gradient of T, $\nabla T : L^2(\mathbb{Z}_n) \to L^2(\mathbb{Z}_n)$ by

$$\begin{aligned} (\nabla T(z))(k) &= (D_k T)(z), \quad k \in \mathbb{Z}_n \\ \left(\overline{\nabla} T(z)\right)(k) &= \left(\overline{D_k} T\right)(z), \quad k \in \mathbb{Z}_n \end{aligned}$$

for all $z \in L^2(\mathbb{Z}_n)$, where $D_k T$, $k \in \mathbb{Z}_n$, are the Wirtinger derivatives of T with respect to $z_k = z(k)$ and $\bar{z}_k = \overline{z(k)}$ respectiely.

Similarly if $T : L^2(\mathbb{Z}_n) \to L^2(\mathbb{Z}_m)$ is of class C^1, we define $\nabla T : L^2(\mathbb{Z}_n) \to L^2(\mathbb{Z}_m) \times L^2(\mathbb{Z}_n)$ by

$$(\nabla T(z))(s,t) = (\nabla T_s(z))(t), \quad (s,t) \in \mathbb{Z}_m \times \mathbb{Z}_n,$$

for all $z \in L^2(\mathbb{Z}_n)$ where $T_s : L^2(\mathbb{Z}_n) \to \mathbb{C}$ is defined by

$$T_s(z) = (Tz)(s), \quad z \in L^2(\mathbb{Z}_n),$$

hence

$$(\nabla T(z))(s,t) = (D_t T_s)(z), \quad t \in \mathbb{Z}_n$$

Using the same idea, if $T : L^2(\mathbb{Z}_n \times \mathbb{Z}_l) \times L^2(\mathbb{Z}_p \times \mathbb{Z}_q)) \to L^2(\mathbb{Z}_{\tilde{n}} \times \mathbb{Z}_{\tilde{l}})$ Then we can define the gradient of T with respect to $z \in L^2(\mathbb{Z}_n \times \mathbb{Z}_l)$,

$$\nabla_z T : L^2(\mathbb{Z}_n \times \mathbb{Z}_l) \times L^2(\mathbb{Z}_p \times \mathbb{Z}_q) \to L^2(\mathbb{Z}_{\tilde{n}} \times \mathbb{Z}_{\tilde{l}}) \times L^2(\mathbb{Z}_n \times \mathbb{Z}_l) \tag{3.6}$$

by

$$(\nabla_z T(z,w))(\tilde{s},\tilde{t},s,t) = \left(\nabla_z T_{(\tilde{s},\tilde{t})}(z,w)\right)(s,t), \quad (\tilde{s},\tilde{t},s,t) \in \mathbb{Z}_{\tilde{n}} \times \mathbb{Z}_{\tilde{l}} \times \mathbb{Z}_n \times \mathbb{Z}_l$$

where $T_{(\tilde{s},\tilde{t})} : L^2(\mathbb{Z}_n \times \mathbb{Z}_l) \times L^2(\mathbb{Z}_p \times \mathbb{Z}_q)) \to \mathbb{C}$ is defined by

$$T_{(\tilde{s},\tilde{t})}(z,w) = (T(z,w))(\tilde{s},\tilde{t})$$

and hence $\nabla_z T_{(\tilde{s},\tilde{t})} : L^2(\mathbb{Z}_n \times \mathbb{Z}_l) \times L^2(\mathbb{Z}_p \times \mathbb{Z}_q)) \to L^2(\mathbb{Z}_n \times \mathbb{Z}_l)$ is defined by

$$\left(\nabla_z T_{(\tilde{s},\tilde{t})}(z,w)\right)(s,t) = \left(\frac{\partial T_{(\tilde{s},\tilde{t})}}{\partial z_{(s,t)}}\right)(z,w),$$

and $\frac{\partial T_{(\tilde{s},\tilde{t})}}{\partial z_{(s,t)}}$ is the Wirtinger derivative of $T_{(\tilde{s},\tilde{t})}$ with respect to $z_{(s,t)} = z(s,t)$. Similarly $\nabla_w T$, $\overline{\nabla}_w T$ and $\overline{\nabla}_z T$ are defined. In (3.6), we can think of $\nabla_z T(z,w)$ as a tensor of dimension $\tilde{n} \times \tilde{l} \times n \times l$.

The followiong definiton is needed for the chain rule of the composition on functions (see Theorem 2.4.3).

Definition 2.3.1 *If A and B are tensors of dimesions $n \times l \times \hat{n} \times \hat{l}$ and $\hat{n} \times \hat{l} \times p \times q$. Then we define AB to be a tensor of dimesion $n \times l \times p \times q$ and its (s,t,k,m)-th element is defined as folows*

$$AB(s,t,k,m) = \sum_{\hat{s}=0}^{\hat{n}} \sum_{\hat{t}=0}^{\hat{l}} A(s,t,\hat{s},\hat{k}) B(\hat{s},\hat{k},k,m). \tag{3.7}$$

where $(s,t,k,m) \in \mathbb{Z}_n \times \mathbb{Z}_l \times \mathbb{Z}_p \times \mathbb{Z}_q$.

For the sake of simplicity of notation, we use D for ∇_z and ∇ for ∇_w, where z is the input of a layer and w is a parameter(filter). Similarly, we define $\overline{\nabla}_z = \bar{D}$ and $\overline{\nabla}_w = \bar{\nabla}$.

We need the followoing chain rule theorem [60].

Theorem 2.3.2 *Let Ω be an open subset of $\mathbb{C}^m$ and $F_1, F_2, \ldots, F_n$ be in $C^1(\Omega)$. Define $F : \Omega \to \mathbb{C}^n$ by*

$$F(z) = (F_1(z), F_2(z), \ldots, F_n(z)), \quad z \in \Omega.$$

Let g be in $C^1(\Omega')$ where $\Omega' \subset \mathbb{C}^n$ is an open set containing the range of F. If we let

$$h = g \circ F.$$

Then for all $z \in \Omega$,

$$\nabla h(z) = \nabla g(w) \nabla F(z) + \overline{\nabla} g(w) \nabla \overline{F}(z)$$

and

$$\overline{\nabla} h(z) = \nabla g(w) \overline{\nabla} F(z) + \overline{\nabla} g(w) \overline{\nabla F}(z),$$

where $w = F(z)$.

Note that in the previous theorem, we consider $\nabla F(z)$ as a matrix and we use the matrix multiplication.

2.4 Pooling Layers and Mappings

In a pooling layer, the input is split into patches (using the cropping operator), and each patch is replaced by one value. In the popular max pooling, this value is the maximal value of the original patch $\max_{z \in \text{patch}} z$. Since the max operator is

not defined for complex numbers, it does not generalize trivially. One choice of the max-pooling operator suggested in [52] is

$$\text{argmax}_{z\in\text{patch}}|z|$$

where argmax gives the point in the patch where the $max_{z\in\text{patch}}|z|$ occurs.

Max-by-magnitude pooling generalizes the real valued max pooling only for non negative inputs, but in typical CNNs, a pooling layer follows a ReLU layer.

Proposition 2.4.1 *Let $\mathfrak{s}$ be a positive integer. Consider $P_{\mathfrak{s}} : L^2(\mathbb{Z}_{\mathfrak{s}} \times \mathbb{Z}_{\mathfrak{s}}) \to \mathbb{C}$, where for all $z \in L^2(Z_{\mathfrak{s}} \times \mathbb{Z}_{\mathfrak{s}})$*

$$P_{\mathfrak{s}}(z) = \text{argmax}\big\{\, |z(s,t)| : \ (s,t) \in \mathbb{Z}_{\mathfrak{s}} \times \mathbb{Z}_{\mathfrak{s}}\big\}.$$

Then the gradient of $P_{\mathfrak{s}}$, $DP_{\mathfrak{s}} : L^2(\mathbb{Z}_{\mathfrak{s}} \times \mathbb{Z}_{\mathfrak{s}}) \to L^2(\mathbb{Z}_{\mathfrak{s}} \times \mathbb{Z}_{\mathfrak{s}})$ is obtained by

$$DP_{\mathfrak{s}}(z) = \chi_{s^*,t^*}.$$

where $z \in L^2(\mathbb{Z}_{\mathfrak{s}} \times \mathbb{Z}_{\mathfrak{s}})$, (s^*, t^*) is the index where the maximum of $|z(s,t)|$ occurs and $\chi_{s*,t^*} \in L^2(\mathbb{Z}_{\mathfrak{s}} \times \mathbb{Z}_{\mathfrak{s}})$ is the characteristic function on $\{(s^*, t^*)\}$, i.e., $\chi_{(s^*,t^*)}(s^*, t^*) = 1$ and $\chi_{(s^*,t^*)}(s,t) = 0$ for $(s,t) \in \mathbb{Z}_{\mathfrak{s}} \times \mathbb{Z}_{\mathfrak{s}} \setminus \{(s^*, t^*)\}$.

Remark 2.4.1 *In Proposition 2.4.1, if the maximum occurs in more than one point, we choose the index randomly. or $(D\text{P}_{\mathfrak{s}}(\text{z}))\,(s,t) = 1$ for all indices (s,t) where the maximum of $|z(s,t)|$ occurs.*

Definition 2.4.1 *Let $\mathfrak{s}$ be a positive integer. In Definition 2.2.5, let $p = q = \mathfrak{s}$. Then $\hat{n} = \lfloor \frac{n}{\mathfrak{s}} \rfloor$ and $\hat{l} = \lfloor \frac{l}{\mathfrak{s}} \rfloor$. We define the max-pooling operator $\varphi_{\mathfrak{s}} : L^2(\mathbb{Z}_n \times \mathbb{Z}_l) \to L^2(\mathbb{Z}_{\hat{n}} \times \mathbb{Z}_{\hat{l}})$ by $(\varphi_{\mathfrak{s}}(z))\,(k,m) = \left(P_{\mathfrak{s}} \circ K^{\mathfrak{s}}_{k,m}\right)(z)$. Then*

$$(\varphi_{\mathfrak{s}}(z))\,(k,m) = \text{argmax}\left\{\left|\left(K^{\mathfrak{s}}_{k,m} z\right)(s,t)\right| : \ (s,t) \in \mathbb{Z}_{\mathfrak{s}} \times \mathbb{Z}_{\mathfrak{s}}\right\}.$$

In fact,

$$(\varphi_{\mathfrak{s}}(z))(k,m) = \operatorname{argmax}\big\{|z(s+k\mathfrak{s}, t+m\mathfrak{s})| : \ (s,t) \in \mathbb{Z}_{\mathfrak{s}} \times \mathbb{Z}_{\mathfrak{s}}\big\}.$$

Theorem 2.4.2 *Let $\mathfrak{s}$ be a positive integer. In Definition 2.2.5, let $p = q = \mathfrak{s}$ and $\hat{n} = \lfloor \frac{n}{\mathfrak{s}} \rfloor$ and $\hat{l} = \lfloor \frac{l}{\mathfrak{s}} \rfloor$. Consider the max-pooling operator $\varphi_{\mathfrak{s}} : L^2(\mathbb{Z}_n \times \mathbb{Z}_l) \to L^2(\mathbb{Z}_{\hat{n}} \times \mathbb{Z}_{\hat{l}})$. Then*

$$D\varphi_{\mathfrak{s}} : L^2(\mathbb{Z}_n \times \mathbb{Z}_l) \to L^2(\mathbb{Z}_{\tilde{n}} \times \mathbb{Z}_{\tilde{l}}) \times L^2(\mathbb{Z}_n \times \mathbb{Z}_l)$$

is given by

$$\begin{aligned}
&(D\varphi_{\mathfrak{s}}(z))(k,m;s,t) \\
&= \chi_{(s^*+k\mathfrak{s}, t^*+m\mathfrak{s})}(s,t) \\
&= \begin{cases} 1, & \text{if } (s,t) = (s^*_{k,m} + k\mathfrak{s}, t^*_{k,m} + m\mathfrak{s}), \\ 0, & \text{otherwise.} \end{cases}
\end{aligned}$$

for all $z \in L^2(\mathbb{Z}_n \times \mathbb{Z}_l)$ and $(k,m;s,t) \in \mathbb{Z}_{\hat{n}} \times \mathbb{Z}_{\hat{l}} \times \mathbb{Z}_n \times \mathbb{Z}_l$ and $(s^*_{k,m}, t^*_{k,m})$ is the index where the maximum of $|\left(K^{\mathfrak{s}}_{k,m} z\right)(s,t)|$ occurs.

By using (2.2) and (2.3), we have the following proposition.

Proposition 2.4.2 *Let $n_1 \gg p_1$ and $l_1 \gg q_1$, $\hat{n}_1 = \lfloor \frac{n_1 - p_1}{\mathfrak{s}_1} \rfloor + 1$ and $\hat{l}_1 = \lfloor \frac{l_1 - q_1}{\mathfrak{s}_1} \rfloor + 1$, where $\mathfrak{s}_1$ is a positive integer. For any $(z,w) \in L^2(\mathbb{Z}_{n_1} \times \mathbb{Z}_{l_1}) \times L^2(\mathbb{Z}_{p_1} \times \mathbb{Z}_{q_1})$ and $(k,m) \in \mathbb{Z}_{\hat{n}_1} \times Z_{\hat{l}_1}$*

$$DC_{\mathfrak{s}_1}(z,w)(k,m,s,t) = \left(E^{\mathfrak{s}_1}_{k,m} w\right)(s,t) \quad (s,t) \in \mathbb{Z}_{n_1} \times \mathbb{Z}_{l_1}$$

and

$$\nabla C_{\mathfrak{s}_1}(z,w)(k,m,s,t) = \left(K^{\mathfrak{s}_1}_{k,m} z\right)(s,t), \quad (s,t) \in \mathbb{Z}_{p_1} \times \mathbb{Z}_{q_1}.$$

Theorem 2.4.3 *Let ψ be a suitable elementwise non-linearity function such as* ReLU *and $\varphi_{\mathfrak{s}}$ be the max-pooling operator defined above. Consider the following diagram*

$$\begin{array}{c} L^2(\mathbb{Z}_{n_1} \times \mathbb{Z}_{l_1}) \times L^2(\mathbb{Z}_{p_1} \times \mathbb{Z}_{q_1}) \\ \downarrow C_{\mathfrak{s}} \\ L^2(\mathbb{Z}_{\hat{n}_1} \times \mathbb{Z}_{\hat{l}_1}) \\ \downarrow \psi \\ L^2(\mathbb{Z}_{\hat{n}_1} \times \mathbb{Z}_{\hat{l}_1}) \\ \downarrow \varphi_{\mathfrak{s}} \\ L^2(\mathbb{Z}_{n_2} \times \mathbb{Z}_{l_2}) \end{array}$$

where $n_1 \gg p_1$ and $l_1 \gg q_1$. Let $\hat{n}_1 = \lfloor \frac{n_1 - p_1}{\mathfrak{s}} \rfloor + 1$ and $\hat{l}_1 = \lfloor \frac{l_1 - q_1}{\mathfrak{s}} \rfloor + 1$, $n_2 = \lfloor \frac{\hat{n}_1}{\mathfrak{s}} \rfloor$ and $l_2 = \lfloor \frac{\hat{l}_1}{\mathfrak{s}} \rfloor$ for a positive integer $\mathfrak{s}$. For any $(z, w) \in L^2(\mathbb{Z}_{n_1} \times \mathbb{Z}_{l_1}) \times L^2(\mathbb{Z}_{p_1} \times \mathbb{Z}_{q_1})$, let

$$f(z, w) = \varphi_{\mathfrak{s}} \circ \psi \circ C_{\mathfrak{s}}(z, w).$$

Then

$$\begin{aligned} Df(z, w) = &\left\{ D\varphi_{\mathfrak{s}}\left(\tilde{\tilde{z}}\right)) \, D\psi(\tilde{z}) + \bar{D}\varphi_{\mathfrak{s}}\left(\tilde{\tilde{z}}\right)) \, D\bar{\psi}(\tilde{z}) \right\} DC_{\mathfrak{s}_1}(z, w) + \\ &\left\{ D\varphi_{\mathfrak{s}}\left(\tilde{\tilde{z}}\right)) \, \bar{D}\psi(\tilde{z}) + \bar{D}\varphi_{\mathfrak{s}}\left(\tilde{\tilde{z}}\right) \overline{D\psi}(\tilde{z}) \right\} D\overline{C_{\mathfrak{s}_1}}(z, w). \end{aligned}$$

and

$$\begin{aligned} \nabla f(z, w) = &\left\{ D\varphi_{\mathfrak{s}}\left(\tilde{\tilde{z}}\right)) \, D\psi(\tilde{z}) + \bar{D}\varphi_{\mathfrak{s}}\left(\tilde{\tilde{z}}\right)) \, D\bar{\psi}(\tilde{z}) \right\} \nabla C_{\mathfrak{s}_1}(z, w) + \\ &\left\{ D\varphi_{\mathfrak{s}}\left(\tilde{\tilde{z}}\right)) \, \bar{D}\psi(\tilde{z}) + \bar{D}\varphi_{\mathfrak{s}}\left(\tilde{\tilde{z}}\right) \overline{D\psi}(\tilde{z}) \right\} \nabla\overline{C_{\mathfrak{s}_1}}(z, w). \end{aligned}$$

where $\tilde{z} = C_{\mathfrak{s}}(z, w)$ and $\tilde{\tilde{z}} = (\psi \circ C_{\mathfrak{s}})(z, w)$.

The above formulas are straightforward using the chain rule (Theorem 2.3.2) and (3.7). In layer i, Let ψ_i be the elementwise non-linearity function such as ReLU.

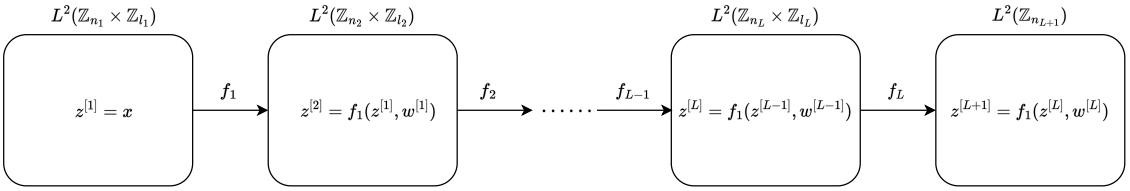


Figure 2.1: Neural Network Layers

Consider the following diagram for $1 \leq i \leq L-1$

$$\begin{array}{c} L^2(\mathbb{Z}_{n_i} \times \mathbb{Z}_{l_i}) \times L^2(\mathbb{Z}_{p_i} \times \mathbb{Z}_{q_i}) \\ \downarrow C_{\mathfrak{s}_i} \\ L^2(\mathbb{Z}_{\hat{n}_i} \times \mathbb{Z}_{\hat{l}_i}) \\ \downarrow \psi_i \\ L^2(\mathbb{Z}_{\hat{n}_i} \times \mathbb{Z}_{\hat{l}_i}) \\ \downarrow \varphi_{\mathfrak{s}_i} \\ L^2(\mathbb{Z}_{n_{i+1}} \times \mathbb{Z}_{l_{i+1}}) \end{array}$$

where $n_i \gg p_i$ and $l_i \gg q_i$. Let $\hat{n}_i = \lfloor \frac{n_i - p_i}{\mathfrak{s}_i} \rfloor + 1$ and $\hat{l}_i = \lfloor \frac{l_i - q_i}{\mathfrak{s}_i} \rfloor + 1$ $n_{i+1} = \lfloor \frac{\hat{n}_i}{\mathfrak{s}_i} \rfloor$ and $l_{i+1} = \lfloor \frac{\hat{l}_i}{\mathfrak{s}_i} \rfloor$ for a positive integer $\mathfrak{s}_i$ (note that the stride in the pooling layer does not have to be the same as the stride in the convolutional layer). Set

$$f_i(z^{[i]}, w^{[i]}) = \varphi_{\mathfrak{s}_i} \circ \psi_i \circ C_{\mathfrak{s}_i}(z^{[i]}, w^{[i]}), \tag{4.8}$$

for any $(z^{[i]}, w^{[i]}) \in L^2(\mathbb{Z}_{n_i} \times \mathbb{Z}_{l_i}) \times L^2(\mathbb{Z}_{p_i} \times \mathbb{Z}_{q_i})$.

For $i = L$, we define $f_L : L^2(\mathbb{Z}_{p_L} \times \mathbb{Z}_{q_L}) \times L^2(\mathbb{Z}_{n_L} \times \mathbb{Z}_{l_L}) \to L^2(\mathbb{Z}_{n_{L+1}})$??? so that it is fully connected.

Let $\{(z_k, y_k)\}_{k=1}^m$ be our m training samples where for all $k = 1, 2, \ldots, m$, $z_k \in L^2(\mathbb{Z}_{n_1} \times \mathbb{Z}_{l_1})$ and $y_k \in L^2(\mathbb{Z}_{n_{L+1}})$.

Let $z^{[1]} = z_1 \in L^2(\mathbb{Z}_{n_1} \times \mathbb{Z}_{l_1})$ and $y = y_1 \in L^2(\mathbb{Z}_{n_{L+1}})$. Choose filters $w^{[1]}, w^{[2]}, \ldots, w^{[L]}$ and the learning rate $\eta \geq 0$. Suppse that we have a regression problem (we can work

on the classification problem, later). Define

$$F : L^2(\mathbb{Z}_{n_1} \times \mathbb{Z}_{l_1}) \times \prod_{k=1}^{L} L^2(\mathbb{Z}_{p_k} \times \mathbb{Z}_{q_k}) \to L^2(\mathbb{Z}_{n_{L+1}})$$

by

$$\begin{aligned} &F(z; w^{[1]}, w^{[2]}, \ldots, w^{[L]}) \\ &= f_L\left(f_{L-1}\left(\ldots\left(f_2\left(f_1(z, w^{[1]}), w^{[2]}\right), \ldots\right), w^{[L-1]}\right), w^{[L]}\right). \end{aligned} \tag{4.9}$$

We define a cost function

$$J_R : \left(L^2(\mathbb{Z}_{n_1} \times \mathbb{Z}_{l_1})\right)^m \times \left(L^2(\mathbb{Z}_{n_{L+1}})\right)^m \times \prod_{k=1}^{L} L^2(\mathbb{Z}_{p_k} \times \mathbb{Z}_{q_k}) \to \mathbb{C}$$

by

$$\begin{aligned} &J_R(z_1, \ldots, z_m, y_1, \ldots, y_m; w^{[1]}, w^{[2]}, \ldots, w^{[L]}) \\ &= \frac{1}{2m} \sum_{k=1}^{m} \left\| -y_k + F(z_k; w^{[1]}, w^{[2]}, \ldots, w^{[L]}) \right\|^2_{L^2(\mathbb{Z}_{n_{L+1}})} \end{aligned} \tag{4.10}$$

where the network, predicts

$$\tilde{y}_k = F(z_k; w^{[1]}, w^{[2]}, \ldots, w^{[L]}), \quad k = 1, 2, \ldots, m.$$

Let $w = (w^{[1]}, w^{[2]}, \ldots, w^{[L]})$. We consider

$$\min_w J_R(x, y, w^{[1]}, w^{[2]}, \ldots, w^{[L]}),$$

and we want to estimate w for which the minimum of J_R occurs. Let

$$J : L^2(\mathbb{Z}_{n_1} \times \mathbb{Z}_{l_1}) \times L^2(\mathbb{Z}_{n_{L+1}}) \times \prod_{k=1}^{L} L^2(\mathbb{Z}_{p_k} \times \mathbb{Z}_{q_k}) \to \mathbb{C}$$

defined by

$$J(z,y;w^{[1]},w^{[2]},\ldots,w^{[L]}) = \frac{1}{2}\left\|-y+F(z;w^{[1]},w^{[2]},\ldots,w^{[L]})\right\|^2_{L^2(\mathbb{Z}_{n_{L+1}})}.$$

Then

$$\begin{aligned} &J_R(z_1,\ldots,z_m,y_1,\ldots,y_m;w^{[1]},w^{[2]},\ldots,w^{[L]}) \\ &= \frac{1}{m}\sum_{k=1}^{m} J(z_k,y_k;w^{[1]},w^{[2]},\ldots,w^{[L]}). \end{aligned} \tag{4.11}$$

Remark 2.4.2 *Note that for simplicty we have chosen one filter in each layer, but in reality, we may choose more filters for each layer. In this case, the depth of the layer will increase(we also have choosen to work with inputs of depth one, for simplicity of notations)*

Definition 2.4.4 *We define the head maps as follows. Let* $\alpha_0 = \mathrm{id}$*, where* id *is the identity operator. Define*

$$\alpha_1 : L^2(\mathbb{Z}_{n_1}\times\mathbb{Z}_{l_1})\times L^2(\mathbb{Z}_{p_1}\times\mathbb{Z}_{q_1}) \to L^2(\mathbb{Z}_{n_2}\times\mathbb{Z}_{l_2})$$

by

$$\alpha_1(z,w^{[1]}) = f_1(z,w^{[1]}), \quad (z,w^{[1]}) \in L^2(\mathbb{Z}_{n_1}\times\mathbb{Z}_{l_1})\times L^2(\mathbb{Z}_{p_1}\times\mathbb{Z}_{q_1})$$

and define

$$\alpha_2 : L^2(\mathbb{Z}_{n_1}\times\mathbb{Z}_{l_1})\times\prod_{k=1}^{2} L^2(\mathbb{Z}_{p_k}\times\mathbb{Z}_{q_k}) \to L^2(\mathbb{Z}_{n_3}\times\mathbb{Z}_{l_3})$$

by

$$\alpha_2(z,w^{[1]},w^{[2]}) = f_2(f_1(z,w^{[1]}),w^{[2]}),$$

where $(z, w^{[1]}, w^{[2]}) \in L^2(\mathbb{Z}_{n_1} \times \mathbb{Z}_{l_1}) \times \prod_{k=1}^{2} L^2(\mathbb{Z}_{p_k} \times \mathbb{Z}_{q_k})$ *and similarly, for all* $1 \leq i \leq L$, *we define*

$$\alpha_i : L^2(\mathbb{Z}_{n_1} \times \mathbb{Z}_{l_1}) \times \prod_{k=1}^{i} L^2(\mathbb{Z}_{p_k} \times \mathbb{Z}_{q_k}) \to L^2(\mathbb{Z}_{n_{i+1}} \times \mathbb{Z}_{l_{i+1}})$$

by

$$\alpha_i(z, w^{[1]}, w^{[2]}, \ldots, w^{[i]}) = f_i\left(f_{i-1}\left(\ldots\left(f_2\left(f_1(z, w^{[1]}), w^{[2]}\right)\right)\ldots, w^{[i-1]}\right), w^{[i]}\right),$$

where $(z, w^{[1]}, w^{[2]}, \ldots, w^{[i]}) \in L^2(\mathbb{Z}_{n_1} \times \mathbb{Z}_{l_1}) \times \prod_{k=1}^{i} L^2(\mathbb{Z}_{p_k} \times \mathbb{Z}_{q_k})$.

Definition 2.4.5 *We define the tail map*

$$\beta_1 : L^2(\mathbb{Z}_{n_1} \times \mathbb{Z}_{l_1}) \times \prod_{k=1}^{L} L^2(\mathbb{Z}_{p_k} \times \mathbb{Z}_{q_k}) \to L^2(\mathbb{Z}_{n_{L+1}})$$

by

$$\beta_1(z^{[1]}, w^{[1]}, w^{[2]}, \ldots, w^{[L]}) = f_L\left(f_{L-1}\left(\ldots\left(f_2\left(f_1(z^{[1]}, w^{[1]}), w^{[2]}\right)\right)\ldots, w^{[L-1]}\right), w^{[L]}\right),$$

where $(z^{[1]}, w^{[1]}, w^{[2]}, \ldots, w^{[L]}) \in L^2(\mathbb{Z}_{n_1} \times \mathbb{Z}_{l_1}) \times \prod_{k=1}^{L} L^2(\mathbb{Z}_{p_k} \times \mathbb{Z}_{q_k})$. *Moreover for* $1 \leq i \leq L$, *we define*

$$\beta_i : L^2(\mathbb{Z}_{n_i} \times \mathbb{Z}_{l_i}) \times \prod_{k=i}^{L} L^2(\mathbb{Z}_{p_k} \times \mathbb{Z}_{q_k}) \to L^2(\mathbb{Z}_{n_{L+1}})$$

by

$$\beta_i(z^{[i]}, w^{[i]}, w^{[i+1]}, \ldots, w^{[L]}) = f_L\left(f_{L-1}\left(\ldots\left(f_{i+1}\left(f_i(z^{[i]}, w^{[i]}), w^{[i+1]}\right)\right)\ldots, w^{[L-1]}\right), w^{[L]}\right),$$

where $(z^{[i]}, w^{[i]}, w^{[i+1]}, \ldots, w^{[L]}) \in L^2(\mathbb{Z}_{n_i} \times \mathbb{Z}_{l_i}) \times \prod_{k=i}^{L} L^2(\mathbb{Z}_{p_k} \times \mathbb{Z}_{q_k})$. *We define*

$$\beta_{L+1} = \mathrm{id} : \mathrm{L}^2(\mathbb{Z}_{\mathrm{n_{L+1}}}) \to \mathrm{L}^2(\mathbb{Z}_{\mathrm{n_{L+1}}}).$$

As we see above, in fact $\beta_1 = F$.

For the ease of computations, we suppress the dependency of the function f_i on the parameter $w^{[i]}$, and consider it as a function from $L^2(\mathbb{Z}_{n_i} \times \mathbb{Z}_{l_i})$ to $L^2(\mathbb{Z}_{n_{i+1}} \times \mathbb{Z}_{l_{i+1}})$. Using the same idea, we can write

$$\alpha_i = f_i \circ f_{i-1} \circ \cdots \circ f_1$$

and

$$\beta_i = f_L \circ f_{L-1} \circ \cdots \circ f_i$$

The following proposition is straightforward from the definitions.

Proposition 2.4.3 *We have*

1) $\beta_i = \beta_{i+1} \circ f_i, \quad i = 1, 2, \ldots, L.$

2) $F = \beta_{i+1} \circ \alpha_i, \quad i = 0, 1, \ldots, L.$

3) $\alpha_i = f_i \circ \alpha_{i-1}, \quad i = 1, 2, \ldots, L.$

4) $F = \beta_{i+1} \circ f_i \circ \alpha_{i-1}, \quad i = 1, 2, \ldots, L.$

It is very important to remember that β_{i+1} and α_{i-1} do not depend on the parameter $w^{[i]}$. To minimize J_R, we will use the gradient descent method.

2.5 Adaptive Gradient Descent Method

Using the Wirtinger derivate, we can define adaptive, gradient descent method as follows. We fix the learning rate η (can be a complex number) and we initialize $(w_0^{[1]}, w_0^{[2]}, \ldots, w_0^{[L]}) \in \prod_{k=1}^{L} L^2(\mathbb{Z}_{p_k} \times \mathbb{Z}_{q_k})$ Let N be the number of iterations. Then for $1 \leq i \leq L$ and for $k = 1, 2, \ldots, N$,

$$w_{k+1}^{[i]} = w_k^{[i]} - \eta \overline{\nabla}_{w_k^{[i]}} J_R(z_1, \ldots, z_m, y_1, \ldots, y_m; w_k^{[1]}, w_k^{[2]}, \ldots, w_k^{[L]})$$

Using (4.11),

$$w_{k+1}^{[i]} = w_k^{[i]} - \frac{\eta}{m} \sum_{k=1}^{m} \overline{\nabla}_{w_k^{[i]}} J(z_k, y_k; w_k^{[1]}, w_k^{[2]}, \ldots, w_k^{[L]})$$

So, we need to find $J(z, y; w^{[1]}, w^{[2]}, \ldots, w^{[L]})$. We give a set of results which will be used to obtain $\overline{\nabla} J(z, y; w^{[1]}, w^{[2]}, \ldots, w^{[L]})$.

Proposition 2.5.1 *Consider the function J defined in (4.11). Let $(x, y; w^{[1]}, w^{[2]}, \ldots, w^{[L]})$ be in $L^2(\mathbb{Z}_{n_1} \times \mathbb{Z}_{l_1}) \times L^2(\mathbb{Z}_{n_{L+1}}) \times \prod_{k=1}^{L} L^2(\mathbb{Z}_{p_k} \times \mathbb{Z}_{q_k})$ and*

$$\tilde{y} = F(x; w^{[1]}, w^{[2]}, \ldots, w^{[L]}). \tag{5.12}$$

where F is defined in (4.9) Then for $i = 1, 2, \ldots, L$,

$$\begin{aligned}
&\overline{\nabla}_{w^{[i]}} J(x, y; w^{[1]}, w^{[2]}, \ldots, w^{[L]}) : \\
&L^2(\mathbb{Z}_{n_1} \times \mathbb{Z}_{l_1}) \times L^2(\mathbb{Z}_{n_{L+1}}) \times \prod_{k=1}^{L} L^2(\mathbb{Z}_{p_k} \times \mathbb{Z}_{q_k}) \to L^2(\mathbb{Z}_{p_i} \times \mathbb{Z}_{q_i})
\end{aligned}$$

is obtained by

$$\begin{aligned}
&\overline{\nabla}_{w^{[i]}} J(x, y; w^{[1]}, w^{[2]}, \ldots, w^{[L]}) \\
&= \frac{1}{2} \sum_{k \in \mathbb{Z}_{n_{L+1}}} \left(\overline{-y_k + \tilde{y}_k}\right) \overline{\nabla}_{w^{[i]}} F(x; w^{[1]}, w^{[2]}, \ldots, w^{[L]})(k) \\
&+ \frac{1}{2} \sum_{k \in \mathbb{Z}_{n_{L+1}}} (-y_k + \tilde{y}_k)) \, \overline{\nabla_{w^{[i]}} F}(x; w^{[1]}, w^{[2]}, \ldots, w^{[L]})(k).
\end{aligned}$$

Proof Using the product rule,

$$\begin{aligned}
&\overline{\nabla}_{w^{[i]}} J(x, y; w^{[1]}, w^{[2]}, \ldots, w^{[L]}) \\
&= \frac{1}{2} \overline{\nabla}_{w^{[i]}} \left(-y + F(x; w^{[1]}, w^{[2]}, \ldots, w^{[L]}), -y + F(x; w^{[1]}, w^{[2]}, \ldots, w^{[L]})\right)_{L^2(\mathbb{Z}_{n_{L+1}})} \\
&= \frac{1}{2} \sum_{k \in \mathbb{Z}_{n_{L+1}}} \left(\overline{-y_k + F(x; w^{[1]}, w^{[2]}, \ldots, w^{[L]})(k)}\right) \overline{\nabla}_{w^{[i]}} F(x; w^{[1]}, w^{[2]}, \ldots, w^{[L]})(k) \\
&+ \frac{1}{2} \sum_{k \in \mathbb{Z}_{n_{L+1}}} \left(-y_k + F(x; w^{[1]}, w^{[2]}, \ldots, w^{[L]})(k)\right) \overline{\nabla_{w^{[i]}} F}(x; w^{[1]}, w^{[2]}, \ldots, w^{[L]})(k).
\end{aligned}$$

□

Note that in the above theorem, since

$$F : L^2(\mathbb{Z}_{n_1} \times \mathbb{Z}_{l_1}) \times \prod_{k=1}^{L} L^2(\mathbb{Z}_{p_k} \times \mathbb{Z}_{q_k}) \to L^2(\mathbb{Z}^{n_L+1})$$

Then

$$\nabla_{w^{[i]}} F(x; w^{[1]}, w^{[2]}, \ldots, w^{[L]}) : L^2(\mathbb{Z}_{n_1} \times \mathbb{Z}_{l_1}) \times \prod_{k=1}^{L} L^2(\mathbb{Z}_{p_k} \times \mathbb{Z}_{q_k}) \to L^2(\mathbb{Z}^{n_L+1}) \times L^2(\mathbb{Z}_{p_i} \times \mathbb{Z}_{q_i})$$

and hence for all $k \in \mathbb{Z}_{n_L+1}$, we can think of $\nabla_{w^{[i]}} F(x; w^{[1]}, w^{[2]}, \ldots, w^{[L]})(k)$ as a matrix of dimension $p_i \times q_i$ where its (s,t)-th element is defined by

$$\nabla_{w^{[i]}} F(x; w^{[1]}, w^{[2]}, \ldots, w^{[L]})(k, s, t).$$

Proposition 2.5.2 *For all $i = 1, \ldots, L$ and $(x; w^{[1]}, w^{[2]}, \ldots, w^{[L]})$ in $L^2(\mathbb{Z}_{n_1} \times \mathbb{Z}_{l_1}) \times \prod_{k=1}^{L} L^2(\mathbb{Z}_{p_k} \times \mathbb{Z}_{q_k})$.*

$$\begin{aligned}
&\overline{\nabla}_{w^{[i]}} F(x; w^{[1]}, w^{[2]}, \ldots, w^{[L]}) \\
&= D\beta_{i+1}(z^{[i+1]}, w^{[i+1]}, \ldots, w^{[L]}) \overline{\nabla}_{w^{[i]}} f_i(z^{[i]}, w^{[i]}) \\
&+ \bar{D}\beta_{i+1}(z^{[i+1]}, w^{[i+1]}, \ldots, w^{[L]}) \overline{\nabla_{w^{[i]}} f_i}(z^{[i]}, w^{[i]}).
\end{aligned}$$

and

$$\begin{aligned}
&\overline{\nabla_{w^{[i]}} F}(x; w^{[1]}, w^{[2]}, \ldots, w^{[L]}) \\
&= D\overline{\beta}_{i+1}(z^{[i+1]}, w^{[i+1]}, \ldots, w^{[L]}) \overline{\nabla}_{w^{[i]}} f_i(z^{[i]}, w^{[i]}) \\
&+ \overline{D\beta}_{i+1}(z^{[i+1]}, w^{[i+1]}, \ldots, w^{[L]}) \overline{\nabla_{w^{[i]}} f_i}(z^{[i]}, w^{[i]}).
\end{aligned}$$

where

- $z^{[1]} = x$,

- $z^{[2]} = \alpha_1(z^{[1]}, w^{[1]}) = f_1(z^{[1]}, w^{[1]})$,
- $z^{[3]} = \alpha_2(z^{[1]}, w^{[1]}, w^{[2]}) = f_2(z^{[2]}, w^{[2]})$
- *and for* $i = 1, 2, \ldots, L$,

$$z^{[i]} = \alpha_{i-1}(z^{[1]}, , w^{[1]}, w^{[2]}, \ldots, w^{[i-1]}) = f_{i-1}(z^{[i-1]}, w^{[i-1]})$$

Proof By Proposition 2.4.3,

$$F = \beta_{i+1} \circ f_i \circ \alpha_{i-1}, \quad i = 1, 2, \ldots, L.$$

By the chain rule (Theorem 2.3.2) and the fact that β_{i+1} and α_{i-1} do not depend on $w^{[i]}$.

$$\begin{aligned}
&\overline{\nabla}_{w^{[i]}} F(x; w^{[1]}, w^{[2]}, \ldots, w^{[L]}) \\
&= D\beta_{i+1}(f_i(\alpha_{i-1}(z^{[1]}, w^{[1]}, \ldots, w^{[i-1]}), w^{[i]}), w^{[i+1]}, \ldots, w^{[L]}) \\
&\times \overline{\nabla}_{w^{[i]}} f_i(\alpha_{i-1}(z^{[1]}, w^{[1]}, \ldots, w^{[i-1]}), w^{[i]}) \\
&+ \bar{D}\beta_{i+1}(f_i(\alpha_{i-1}(z^{[1]}, w^{[1]}, \ldots, w^{[i-1]}), w^{[i]}), w^{[i+1]}, \ldots, w^{[L]}) \\
&\times \overline{\nabla_{w^{[i]}} f_i}(\alpha_{i-1}(z^{[1]}, w^{[1]}, \ldots, w^{[i-1]}), w^{[i]}) \\
&= D\beta_{i+1}(z^{[i+1]}, w^{[i+1]}, \ldots, w^{[L]}) \overline{\nabla}_{w^{[i]}} f_i(z^{[i]}, w^{[i]}) \\
&+ \bar{D}\beta_{i+1}(z^{[i+1]}, w^{[i+1]}, \ldots, w^{[L]}) \overline{\nabla_{w^{[i]}} f_i}(z^{[i]}, w^{[i]}).
\end{aligned}$$

Similarly since

$$\overline{F} = \overline{\beta}_{i+1} \circ f_i \circ \alpha_{i-1}, \quad i = 1, 2, \ldots, L,$$

and $\bar{\nabla}\bar{F} = \overline{\nabla F}$, we can get the second part.

□

Note that in Proposition 2.5.2, we have used the product definition in (3.7) for

$$D\overline{\beta}_{i+1}(z^{[i+1]}, w^{[i+1]}, \ldots, w^{[L]}) \overline{\nabla}_{w^{[i]}} f_i(z^{[i]}, w^{[i]})$$

and

$$\overline{D\beta}_{i+1}(z^{[i+1]}, w^{[i+1]}, \ldots, w^{[L]})\overline{\nabla_{w^{[i]}} f_i}(z^{[i]}, w^{[i]}).$$

Theorem 2.5.1 *For all $i = 1, \ldots, L$ and $(x; w^{[1]}, w^{[2]}, \ldots, w^{[L]})$ in $L^2(\mathbb{Z}_{n_1} \times \mathbb{Z}_{l_1}) \times L^2(\mathbb{Z}_{p_1} \times \mathbb{Z}_{q_1}) \times L^2(\mathbb{Z}_{p_2} \times \mathbb{Z}_{q_2}) \times L^2(\mathbb{Z}_{p_L} \times \mathbb{Z}_{q_L})$, we have*

$$\begin{aligned}
&D\beta_i(z^{[i]}, w^{[i]}, w^{[i+1]}, \ldots, w^{[L]}) \\
&= D\beta_{i+1}\left(z^{[i+1]}, w^{[i+1]}, \ldots, w^{[L]}\right) Df_i(z^{[i]}, w^{[i]}) \\
&+\bar{D}\beta_{i+1}\left(z^{[i+1]}, w^{[i+1]}, \ldots, w^{[L]}\right) D\overline{f}_i(z^{[i]}, w^{[i]}).
\end{aligned}$$

and

$$\begin{aligned}
&\bar{D}\beta_i(z^{[i]}, w^{[i]}, w^{[i+1]}, \ldots, w^{[L]}) \\
&= D\beta_{i+1}\left(z^{[i+1]}, w^{[i+1]}, \ldots, w^{[L]}\right) \bar{D}f_i(z^{[i]}, w^{[i]}) \\
&+\bar{D}\beta_{i+1}\left(z^{[i+1]}, w^{[i+1]}, \ldots, w^{[L]}\right) \overline{Df}_i(z^{[i]}, w^{[i]}).
\end{aligned}$$

where $z^{[1]}, z^{[2]}, \ldots, z^{[L]}$ is defined in Proposition 2.5.2.

Proof By Proposition 2.4.3,

$$\beta_i(z^{[i]}, w^{[i]}, w^{[i+1]}, \ldots, w^{[L]}) = \beta_{i+1}\left(f_i(z^{[i]}, w^{[i]}), w^{[i+1]}, \ldots, w^{[L]}\right).$$

Apply the chain rule,

$$\begin{aligned}
&D\beta_i(z^{[i]}, w^{[i]}, w^{[i+1]}, \ldots, w^{[L]}) \\
&= D\beta_{i+1}\left(f_i(z^{[i]}, w^{[i]}), w^{[i+1]}, \ldots, w^{[L]}\right) Df_i(z^{[i]}, w^{[i]}) \\
&+\bar{D}\beta_{i+1}\left(f_i(z^{[i]}, w^{[i]}), w^{[i+1]}, \ldots, w^{[L]}\right) D\overline{f}_i(z^{[i]}, w^{[i]}) \\
&= D\beta_{i+1}\left(z^{[i+1]}, w^{[i+1]}, \ldots, w^{[L]}\right) Df_i(z^{[i]}, w^{[i]}) \\
&+\bar{D}\beta_{i+1}\left(z^{[i+1]}, w^{[i+1]}, \ldots, w^{[L]}\right) D\overline{f}_i(z^{[i]}, w^{[i]}).
\end{aligned}$$

□

Proposition 2.5.3 *Let*

$$e_L^1 = \tilde{y}_R - y$$

and

$$e_L^2 = \overline{\tilde{y}_R - y}$$

and for $i = 0, 1, \ldots, L$, let

$$\begin{aligned} e_i^1 &= \overline{(\tilde{y}_R - y)} \cdot D\beta_{i+1}(z^{[i+1]}, w^{[i+1]}, \ldots, w^{[L]}), \\ e_i^2 &= (\tilde{y}_R - y) \cdot D\overline{\beta_{i+1}}(z^{[i+1]}, w^{[i+1]}, \ldots, w^{[L]}). \end{aligned}$$

Then for all $i = 1, 2, \ldots, L$

$$\begin{aligned} e_{i-1}^1 &= e_i^1 Df_i(z^{[i]}, w^{[i]}) + \overline{e_i^2} D\overline{f_i}(z^{[i]}, w^{[i]}) \\ e_{i-1}^2 &= e_i^2 Df_i(z^{[i]}, w^{[i]}) + \overline{e_i^1} D\overline{f_i}(z^{[i]}, w^{[i]}) \end{aligned}$$

and

$$\begin{aligned} &\overline{\nabla_{w^{[i]}}} J(x, y; w^{[1]}, w^{[2]}, \ldots, w^{[L]}) \\ &= \frac{1}{2}(e_i^1 + e_i^2)\overline{\nabla}_{w^{[i]}} f_i(z^{[i]}, w^{[i]}) + \frac{1}{2}(\overline{e_i^1 + e_i^2})\overline{\nabla_{w^{[i]}} f_i}(z^{[i]}, w^{[i]}). \end{aligned}$$

Proof By Theorem 2.5.1,

$$\begin{aligned} e_{i-1}^1 &= \left(\overline{\tilde{y}_R - y}\right) \cdot D\beta_i(z^{[i]}, w^{[i]}, \ldots, w^{[L]}) \\ &= \left(\overline{\tilde{y}_R - y}\right) \cdot \left\{ D\beta_{i+1}\left(z^{[i+1]}, w^{[i+1]}, \ldots, w^{[L]}\right) Df_i(z^{[i]}, w^{[i]}) \right. \\ &\left. + \bar{D}\beta_{i+1}\left(z^{[i+1]}, w^{[i+1]}, \ldots, w^{[L]}\right) D\overline{f_i}(z^{[i]}, w^{[i]}) \right\} \\ &= e_i^1 Df_i(z^{[i]}, w^{[i]}) + \overline{e_i^2} D\overline{f_i}(z^{[i]}, w^{[i]}). \end{aligned}$$

Similarly, we can find e_{i-1}^2. To prove the second part, we use Proposition 2.5.1 and

Proposition 2.5.2,

$$
\begin{aligned}
&\overline{\nabla}_{w^{[i]}} J(x,y;w^{[1]},w^{[2]},\ldots,w^{[L]}) \\
&= \frac{1}{2}\sum_{k\in\mathbb{Z}_{n_{L+1}}} \left(\overline{-y_k+\tilde{y}_k}\right)\overline{\nabla}_{w^{[i]}} F(x;w^{[1]},w^{[2]},\ldots,w^{[L]})(k) \\
&+\frac{1}{2}\sum_{k\in\mathbb{Z}_{n_{L+1}}} (-y_k+\tilde{y}_k))\,\overline{\overline{\nabla}_{w^{[i]}} F}(x;w^{[1]},w^{[2]},\ldots,w^{[L]})(k). \\
&= \frac{1}{2}\sum_{k\in\mathbb{Z}_{n_{L+1}}} \left(\overline{-y_k+\tilde{y}_k}\right)\left\{D\beta_{i+1}(z^{[i+1]},w^{[i+1]},\ldots,w^{[L]})(k)\overline{\nabla}_{w^{[i]}} f_i(z^{[i]},w^{[i]})\right. \\
&\left.+\bar{D}\beta_{i+1}(z^{[i+1]},w^{[i+1]},\ldots,w^{[L]})(k)\overline{\nabla_{w^{[i]}} f_i}(z^{[i]},w^{[i]})\right\} \\
&+\frac{1}{2}\sum_{k\in\mathbb{Z}_{n_{L+1}}} (-y_k+\tilde{y}_k))\left\{D\overline{\beta}_{i+1}(z^{[i+1]},w^{[i+1]},\ldots,w^{[L]})(k)\overline{\nabla}_{w^{[i]}} f_i(z^{[i]},w^{[i]})\right. \\
&\left.+\overline{D\beta}_{i+1}(z^{[i+1]},w^{[i+1]},\ldots,w^{[L]})(k)\overline{\nabla_{w^{[i]}} f_i}(z^{[i]},w^{[i]})\right\}.
\end{aligned}
$$

Using the definition of e_i^1 and e_i^2,

$$
\begin{aligned}
&\overline{\nabla}_{w^{[i]}} J(x,y;w^{[1]},w^{[2]},\ldots,w^{[L]}) \\
&= \frac{1}{2}(\overline{-y+\tilde{y}})\cdot\left\{D\beta_{i+1}(z^{[i+1]},w^{[i+1]},\ldots,w^{[L]})\overline{\nabla}_{w^{[i]}} f_i(z^{[i]},w^{[i]})\right. \\
&\left.+\bar{D}\beta_{i+1}(z^{[i+1]},w^{[i+1]},\ldots,w^{[L]})\overline{\nabla_{w^{[i]}} f_i}(z^{[i]},w^{[i]})\right\} \\
&+\frac{1}{2}(-y+\tilde{y}))\cdot\left\{D\overline{\beta}_{i+1}(z^{[i+1]},w^{[i+1]},\ldots,w^{[L]})(k)\overline{\nabla}_{w^{[i]}} f_i(z^{[i]},w^{[i]})\right. \\
&\left.+\overline{D\beta}_{i+1}(z^{[i+1]},w^{[i+1]},\ldots,w^{[L]})(k)\overline{\nabla_{w^{[i]}} f_i}(z^{[i]},w^{[i]})\right\}. \\
&= \frac{1}{2}(e_i^1+e_i^2)\overline{\nabla}_{w^{[i]}} f_i(z^{[i]},w^{[i]}) + \frac{1}{2}(\overline{e_i^1+e_i^2})\overline{\nabla_{w^{[i]}} f_i}(z^{[i]},w^{[i]})
\end{aligned}
\tag{5.13}
$$

□

We are ready to give the backward and forward propagation algorithm in the convolutional neural networks.

Inputs: $z_1,z_2,\ldots,z_m\in L^2(\mathbb{Z}_{n_1}\times\mathbb{Z}_{l_1})$ and $y_1,y_2,\ldots,y_m\in L^2(\mathbb{Z}_{n_{L+1}})$. Filters $w^{[1]},w^{[2]},\ldots,w^{[L]}$,

suitable positive integers $\mathfrak{s}_1, \mathfrak{s}_2, \ldots, \mathfrak{s}_L$ and the learning rate $\eta \geq 0$. number of layers L, stopping criteria

Outputs: Modified $w^{[1]}, w^{[2]}, \ldots, w^{[L]}$ to estimate and calculate the output of a new $z \in L^2(\mathbb{Z}_{n_1} \times \mathbb{Z}_{l_1})$ by using the function $F(z; w^{[1]}, w^{[2]}, \ldots, w^{[L]})$.

Algorithm:

Step 1: For $j = 1, 2, \ldots, m$

Step 1-1 Set $x = z_j$ and $y = y_j$ and $z^{[1]} = z_j$.

Step 1-2 For $i = 1, 2, \ldots, L$, set $z^{[i+1]} = f_i(z^{[i]}, w^{[i]})$ (forward propagation)

Step 1-4 Let $\tilde{y} = z^{[L+1]}$.

Step 1-5 For $i = L, L-1, \ldots, 1$ (backward propagation)

Step 1-5-1 If $i = L$, then set

$$e_L^1 = \tilde{y} - y, \quad e_L^2 = \overline{\tilde{y} - y},$$

else set

$$e_i^1 = e_{i+1}^1 Df_{i+1}(z^{[i+1]}, w^{[i+1]}) + \overline{e_{i+1}^2} D\overline{f_{i+1}}(z^{[i+1]}, w^{[i+1]})$$
$$e_i^2 = e_{i+1}^2 Df_{i+1}(z^{[i+1]}, w^{[i+1]}) + \overline{e_{i+1}^1} D\overline{f_{i+1}}(z^{[i+1]}, w^{[i+1]})$$

Step 1-5-2 Set

$$\nabla_j^i J = \overline{\nabla_{w^{[i]}}} J(x, y; w^{[1]}, w^{[2]}, \ldots, w^{[L]})$$

where

$$\overline{\nabla_{w^{[i]}}} J(x, y; w^{[1]}, w^{[2]}, \ldots, w^{[L]})$$
$$= \frac{1}{2}(e_i^1 + e_i^2)\overline{\nabla}_{w^{[i]}} f_i(z^{[i]}, w^{[i]}) + \frac{1}{2}(\overline{e_i^1 + e_i^2})\overline{\nabla_{w^{[i]}} f_i}(z^{[i]}, w^{[i]}).$$

Step 2: For $i = 1, 2, \ldots, L$, set

$$\tilde{w}^{[i]} = w^{[i]}$$

and

$$w^{[i]} = \tilde{w}^{[i]} - \frac{\eta}{m} \sum_{j=1}^{m} \nabla J_i^j.$$

Step 3: Repeat Step 1 to Step 2 until stopping criteria are achieved (using the modified $w^{[1]}, w^{[2]}, \ldots, w^{[L]}$).

Note that in the above algorithm $f_i(z^{[i]}, w^{[i]})$, $D\overline{f_{i+1}}(z^{[i+1]}, w^{[i+1]})$ and $\overline{\nabla_{w^{[i]}}} f_i(z^{[i]}, w^{[i]})$ are obtained by (4.8), Theorem 2.4.3 and Proposition 2.4.2.

2.6 Pseudo-Differential Operators in Neural Networks

Let $\mathfrak{s}, n, p, l, q$ be positive integers such that $n \gg p$ and $l \gg q$. Let $\hat{n} = \lfloor \frac{n-p}{\mathfrak{s}} \rfloor + 1$ and $\hat{l} = \lfloor \frac{l-q}{\mathfrak{s}} \rfloor + 1$. Let $\sigma : L^2(\mathbb{Z}_{\hat{n}} \times \mathbb{Z}_{\hat{l}}) \times L^2(\mathbb{Z}_p \times \mathbb{Z}_q) \to \mathbb{C}$. Then we define the pseudo-differential operator $T_\sigma : L^2(\mathbb{Z}_n \times \mathbb{Z}_l) \to L^2(\mathbb{Z}_{\hat{n}} \times \mathbb{Z}_{\hat{l}})$ corresponding to the symbol σ by

$$(T_\sigma z)(k, m) = \frac{1}{\sqrt{nl}} \sum_{s \in \mathbb{Z}_n} \sum_{t \in \mathbb{Z}_l} \mathrm{e}^{2\pi i \mathfrak{s}(\frac{ks}{n} + \frac{mt}{l})} \sigma\left(k, m; \frac{p}{n}s, \frac{q}{l}t\right) \hat{z}(s, t)$$

for any $z \in L^2(\mathbb{Z}_n \times \mathbb{Z}_l)$ and $(k, m) \in \mathbb{Z}_n \times \mathbb{Z}_l$ where $\hat{z}$ is the Fourier transform of z sometimes is denoted by $\mathcal{F}z$ defined by

$$(\mathcal{F}z)(s, t) = \hat{z}(s, t) = \frac{1}{\sqrt{nl}} \sum_{k \in \mathbb{Z}_n} \sum_{m \in \mathbb{Z}_l} \mathrm{e}^{-2\pi i(\frac{ks}{n} + \frac{mt}{l})} z(k, m), \quad (s, t) \in \mathbb{Z}_n \times \mathbb{Z}_l.$$

Lemma 2.6.1 *Let $w \in L^2(\mathbb{Z}_p \times \mathbb{Z}_q)$. Then*

$$\mathcal{F}^{-1}\left(E^{\mathfrak{s}}_{k,m} w\right)(s,t) = \sqrt{\frac{pq}{nl}} e^{2\pi i \mathfrak{s}\left(\frac{ks}{n}+\frac{tm}{l}\right)} \left(\mathcal{F}^{-1} w\right)\left(\frac{p}{n}s, \frac{q}{l}t\right)$$

Proof

$$\begin{aligned}
&\mathcal{F}^{-1}\left(E^{\mathfrak{s}}_{k,m} w\right)(s,t) = \\
&= \frac{1}{\sqrt{nl}} \sum_{\tilde{s}=0}^{n-1} \sum_{\tilde{t}=0}^{l-1} e^{2\pi i\left(\frac{\tilde{s}s}{n}+\frac{\tilde{t}t}{l}\right)} \left(E^{\mathfrak{s}}_{k,m} w\right)(\tilde{s},\tilde{t}) \\
&= \frac{1}{\sqrt{nl}} \sum_{\tilde{s}=k\mathfrak{s}}^{k\mathfrak{s}+p-1} \sum_{\tilde{t}=m\mathfrak{s}}^{m\mathfrak{s}+q-1} e^{2\pi i\left(\frac{\tilde{s}s}{n}+\frac{\tilde{t}t}{l}\right)} w(\tilde{s}-k\mathfrak{s}, \tilde{t}-m\mathfrak{s}).
\end{aligned}$$

By the change of variables using $\gamma = \tilde{s} - k\mathfrak{s}$ and $\eta = \tilde{t} - m\mathfrak{s}$, we get

$$\begin{aligned}
&\mathcal{F}^{-1}\left(E^{\mathfrak{s}}_{k,m} w\right)(s,t) = \\
&= \frac{1}{\sqrt{nl}} \sum_{\tilde{s}=0}^{p-1} \sum_{\tilde{t}=0}^{q-1} e^{2\pi i\left(\frac{\tilde{s}s}{n}+\frac{\tilde{t}t}{l}\right)} w(\gamma,\eta). \\
&= \frac{1}{\sqrt{nl}} e^{2\pi i \mathfrak{s}\left(\frac{ks}{n}+\frac{tm}{l}\right)} \sum_{\tilde{s}=0}^{p-1} \sum_{\tilde{t}=0}^{q-1} e^{2\pi i\left(\frac{s}{n}\gamma+\frac{t}{l}\eta\right)} w(\gamma,\eta). \\
&= \sqrt{\frac{pq}{nl}} e^{2\pi i \mathfrak{s}\left(\frac{ks}{n}+\frac{tm}{l}\right)} \left(\mathcal{F}^{-1} w\right)\left(\frac{p}{n}s, \frac{q}{l}t\right).
\end{aligned}$$

□

We have the following theoreom known as Parseval's theorem.

Theorem 2.6.1 *Let $z, w \in L^2(\mathbb{Z}_n)$. Then*

$$(z,w)_{L^2(\mathbb{Z}_n)} = \left(\hat{z},\hat{w}\right)_{L^2(\mathbb{Z}_n)}.$$

Note that the above theorem is true in general for any dimesions. Using Theorem

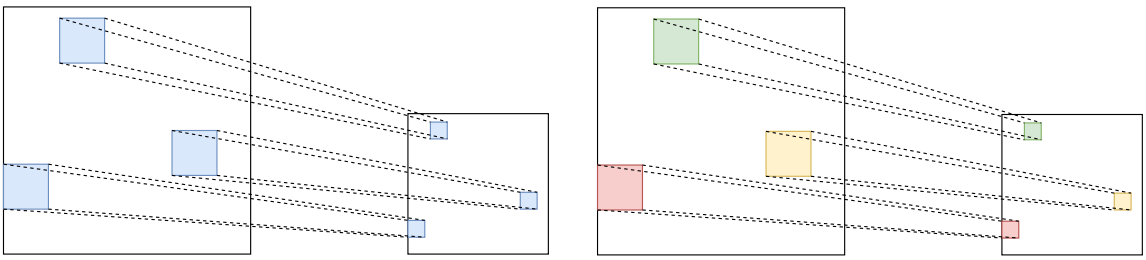
2.6.1 and Lemma 2.6.1, we have

$$
\begin{aligned}
\left(T_\sigma z\right)(k,m) &= \frac{1}{\sqrt{nl}} \sum_{s\in\mathbb{Z}_n} \sum_{t\in\mathbb{Z}_l} \mathrm{e}^{2\pi i\mathfrak{s}(\frac{ks}{n}+\frac{mt}{l})} \sigma\left(k,m;\frac{p}{n}s,\frac{q}{l}t\right)\hat{z}(s,t) \\
&= \frac{1}{\sqrt{pq}} \sum_{s\in\mathbb{Z}_n} \sum_{t\in\mathbb{Z}_l} \left(E^{\mathfrak{s}}_{k,m}\mathcal{F}_2\sigma\right)(k,m,\tilde{s},\tilde{t}) z(\tilde{s},\tilde{t}) \\
&= \frac{1}{\sqrt{pq}} \sum_{\tilde{s}=k\mathfrak{s}}^{k\mathfrak{s}+p-1} \sum_{\tilde{t}=m\mathfrak{s}}^{k\mathfrak{s}+q-1} \left(\mathcal{F}_2\sigma\right)(k,m,\tilde{s}-k\mathfrak{s},,\tilde{t}-m\mathfrak{s}) z(\tilde{s},\tilde{t})
\end{aligned}
\tag{6.14}
$$

where $\mathcal{F}_2\sigma$ is defined by

$$
\mathcal{F}_2\sigma(k,m;u,v) = \frac{1}{\sqrt{pq}} \sum_{s\in\mathbb{Z}_p} \sum_{t\in\mathbb{Z}_q} \mathrm{e}^{-2\pi\mathrm{i}(\frac{\mathrm{us}}{\mathrm{p}}+\frac{\mathrm{vt}}{\mathrm{q}})} \sigma(k,m,s,t)
$$

The following theorem states that every convolutional operator is a pseudo-differential operator where its symbol does not depend on time variables (k,m). Refer to Figure 2.2 for an intuitive illustration showing the differences between Convolution Operator and Pseudo-Differential Operator.



(a) Convolution operator **(b)** Pseudo-differential operator

Figure 2.2: Difference in applying convolution operator and pseudo-differential operator: convolution operator applies a uniform kernel across locations, while pseudo-differential operator employs different kernels at different locations

Theorem 2.6.2 *Let $\mathfrak{s}, n, p, l, q$ be positive integers such that $n \gg p$ and $l \gg q$. Let $\hat{n} = \lfloor \frac{n-p}{\mathfrak{s}} \rfloor + 1$ and $\hat{l} = \lfloor \frac{l-q}{\mathfrak{s}} \rfloor + 1$. Let $w \in L^2(\mathbb{Z}_p \times \mathbb{Z}_q)$. Then for all $z \in L^2(\mathbb{Z}_n \times \mathbb{Z}_l)$ and $(k, m) \in \mathbb{Z}_{\hat{n}} \times \mathbb{Z}_{\hat{l}}$*

$$\left(C_{\mathfrak{s}}(w, z)\right)(k, m) = \left(T_\sigma z\right)(k, m)$$

where for all $(k, m, s, t) \in \mathbb{Z}_n \times \mathbb{Z}_l \times \mathbb{Z}_n \times \mathbb{Z}_l$,

$$\sigma(k, m; s, t) = \sqrt{pq}\left(\mathcal{F}^{-1} w\right)(s, t),$$

and $\mathcal{F}^{-1} w$ is the inverse Fourier transform of w on $L^2(\mathbb{Z}_p \times \mathbb{Z}_q)$.

In the algorithm, we replace, $C_{\mathfrak{s}}(w, z)$ with $T_\sigma z$, and every thing works fine. For clarification let $T(\sigma, z) = T_\sigma z$ where we can look at the operator T as a function of z and symbol σ. The we consider our symbol σ as our filter. Then $DT(\sigma, z)$ is the the gradient with respect to z and $\nabla T(\sigma, z)$ is the gradient with respect to σ.

Proposition 2.6.1 *Let $\mathfrak{s}, n, p, l, q$ be positive integers such that $n \gg p$ and $l \gg q$. Let $\hat{n} = \lfloor \frac{n-p}{\mathfrak{s}} \rfloor + 1$ and $\hat{l} = \lfloor \frac{l-q}{\mathfrak{s}} \rfloor + 1$. Let $\sigma : L^2(\mathbb{Z}_{\hat{n}} \times \mathbb{Z}_{\hat{l}}) \times L^2(\mathbb{Z}_p \times \mathbb{Z}_q) \to \mathbb{C}$. and let $T(\sigma, z) = T_\sigma z$. We have*

$$DT(k, m; u, v) = \frac{1}{\sqrt{pq}}\left(\mathcal{F}_2 \sigma\right)(k, m, u - k\mathfrak{s}, v - m\mathfrak{s})$$

and

$$\nabla T(k, m, s, t) = \frac{1}{\sqrt{nl}} e^{2\pi i \mathfrak{s}(\frac{k}{p}s + \frac{m}{q}t)} \hat{z}(\frac{n}{p}s, \frac{l}{q}t)$$

Note that in fact ∇T is defined on $\mathbb{Z}_{\hat{n}} \times \mathbb{Z}_{\hat{l}} \times \mathbb{Z}_{\hat{n}} \times \mathbb{Z}_{\hat{l}} \times Z_p \times \mathbb{Z}_q$, but since for $(k, m) \neq (k', m')$ $\nabla(k, m, k', m', s, t) = 0$, we set $\nabla(k, m, s, t) := \nabla(k, m, k, m, s, t)$

2.7 Numerical Experiments

2.7.1 Overview

Tensorflow [61] was chosen as our platform for implementing the pseudo convolution layer. In this study, all layer parameters are represented as complex values. It should be noted, however, that real-valued weights can also be utilized in practical applications. Our focus on complex-valued weights in this instance is not due to necessity but serves to substantiate our theoretical discussion about their potential for convergence. Hence, the examples within this paper are intentionally designed to demonstrate this particular use case.

Our data sets include both one-dimensional (signals) and two-dimensional (images) cases. We selected the Ralph-Andrzejak EEG dataset [62], MNIST [63], and CIFAR-10 [64] as benchmark datasets. We do not compare the results with state-of-the-art findings, as pseudo convolution is a relatively new concept; rather, our experiments aim to demonstrate its feasibility. We employed the same training and testing protocols but used different hyperparameters for these three datasets.

For the pseudo convolution layer, we pre-defined the kernel size, as the image size and stride could affect the total number of kernels used. Initially, we retained 20% of the data in the training set as a validation set to determine the appropriate number of epochs, before training the model using the entire training set and the optimized epoch number. During the training process, we started with initial weights and a learning rate, optimizing the learning process with an exponential decay rate of 0.95. We trained the model using mini-batches of sizes 16 or 32. We applied the initialization for complex-valued weights mentioned in [51].

2.7.2 Datasets

The Ralph-Andrzejak EEG dataset[62] includes five signal categories named Z, O, F, N, S, with detailed descriptions for each category presented in Table 2.1. Each category contains 100 EEG recordings approximately 23.6 seconds long, sampled at a rate of 173.61 Hz, yielding 4097 values per record. These data were divided into 1-second long segments. Despite the dataset containing five classes, this study focuses solely on classifying S (Seizure) and NS (Non-Seizure). To demonstrate pseudo convolution for complex values, the Fourier transform of the signals was used as input. Outputs from the fully connected layer were processed by taking their absolute value and then applying a softmax classifier.

Category	Description
Z	Healthy group, recorded with eye open
O	Healthy group, recorded with eye closed
F	Interictal activity, recorded in the epileptogenic zone
N	Interictal activity, recorded at hippocampal location
S	Seizure activity

Table 2.1: Details of the 5 categories of EEG signals

The MNIST dataset[63] comprises 70,000 images of handwritten digits, with each image consisting of 28×28 pixels with values ranging from 0 to 255. The dataset is divided into a default training set with 60,000 images and a testing set of 10,000 images. Preprocessing only involved dividing all pixel values by 255 to normalize the input range to between 0 and 1.

Similarly, the CIFAR-10 dataset[64] includes 60,000 32×32 RGB images, distributed over 10 different classes with each class having 6,000 images. The dataset is split into a training set with 50,000 images and a testing set comprising 10,000 images. For preprocessing, each pixel value was divided by 255, ensuring that the input range lies between 0 and 1.

2.7.3 Results

We evaluated the performance of 1 and 5 pseudo differential operator kernels. In both cases, a ReLu non-linearity activation function and a fully connected layer followed. Table 2.2 shows that the accuracy increases with the number of pseudo convolution kernels. Figure 2.3 indicates that as the number of kernels increases, the average loss per epoch in the training set decreases faster and can reach a lower average loss. Figure 2.5 shows that as the number of epochs increases, the trend of test and training error rates is decreasing.

# of pseudo convolution kernel	1	5
EEG Seizure	0.9692	0.9807
MNIST	0.9699	0.9806
CIFAR-10	0.4725	0.5679

Table 2.2: Accuracy of EEG

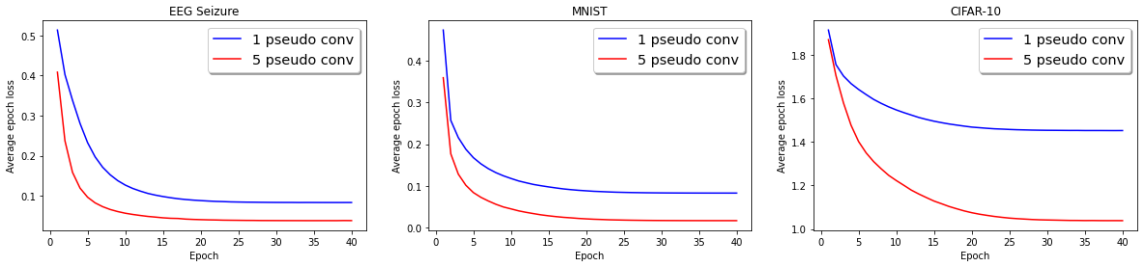


Figure 2.3: Average epoch loss of three different data sets

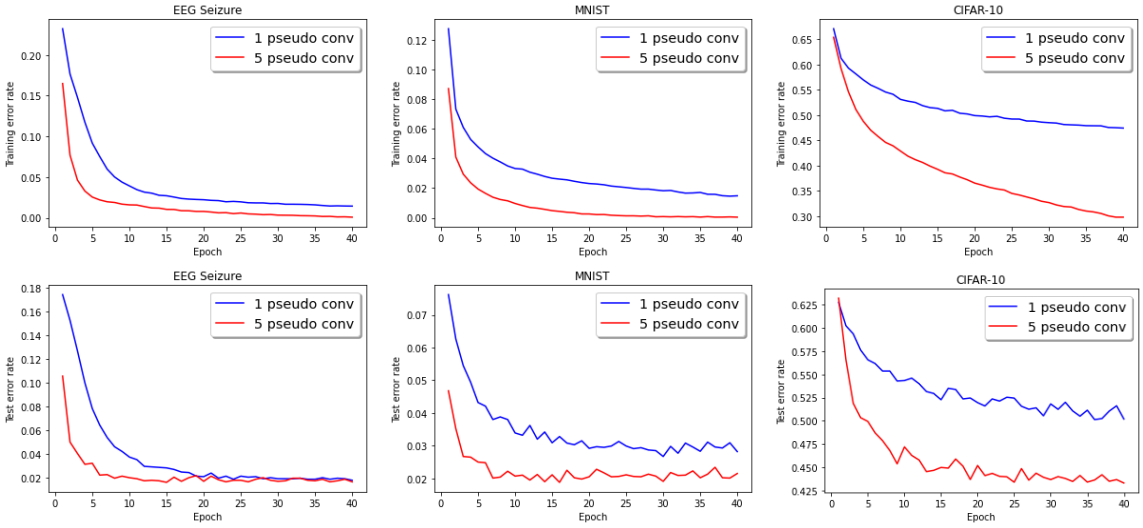


Figure 2.4: Training and test error rate per epoch: the upper 3 figures shows the training error rate and lower 3 figures shows the test error rate

2.8 Discussion

We introduced a new neural network structure based on the theory of pseudo-differential operator and discussed the specific backpropagation algorithm using Wirtinger derivative. Furthermore, we have elaborated on the activation functions and pooling layers necessary for implementing our neural network framework. Our proposed architecture is highly compatible with the pseudo-differential operator theory, and our numerical experiments on benchmark datasets demonstrate its effectiveness. In our future work, we plan to utilize this framework in the field of signal processing, with a particular focus on time-variant filtering.

3 Time-variant Transform

3.1 Introduction

In the realm of biomedical signal processing, Electroencephalogram (EEG) signals are notably challenging due to their non-stationary and time-varying nature. While traditional methods often employ Linear Time-Invariant (LTI) filters, these might not be optimal for EEG signals inherently characterized by time-varying properties. Addressing this, this chapter presents a groundbreaking approach: using neural networks to model and learn a time-varying filter. This innovative method not only offers an enhanced representation and denoising of EEG signals but also holds the potential to bolster the efficiency of subsequent EEG classification tasks.

Time-variant filter applications or transformations have historically been a formidable challenge. In our approach, we utilize parameterized pseudo-differential operators combined with neural networks to craft an apt time-variant parameter function, thus achieving the desired time-variant transformation. Existing literature [65] suggests that, when unconstrained, pseudo-differential operators in discrete scenarios can be matrix-represented, analogous to a fully connected layer in neural networks. Yet, representing a specific pseudo-differential operator using solely a fully connected layer becomes nearly unfeasible due to convergence challenges. A viable workaround is the adoption of ordinary differential equations (ODEs) to constrain the form of

the pseudo-differential operator. As illustrated in [65], a meta-learning technique was introduced to learn such operators via neural networks to address ODEs.

Ordinary Differential Equations (ODEs) are mathematical expressions involving functions of one independent variable and their derivatives. Their capacity to describe the behavior of numerous systems across disciplines, from physics to engineering, makes them indispensable. Specifically, in signal processing, ODEs play a pivotal role in modeling and designing filters. A widely-used Linear Time-Invariant (LTI) filter in such applications can be portrayed by an ODE of the form:

$$a_0 y(t) + a_1 \frac{dy(t)}{dt} + \ldots + a_n \frac{d^n y(t)}{dt^n} = b_0 x(t) + b_1 \frac{dx(t)}{dt} + \ldots + b_m \frac{d^m x(t)}{dt^m} \tag{1.1}$$

Here, $x(t)$ represents the input signal, $y(t)$ is the output, and a_i, b_j are the filter coefficients. This kind of ODE, known as a linear constant coefficient differential equation, encapsulates a wide spectrum of LTI filters.

However, the non-stationary nature of many signals and systems necessitates the extension of LTI filters to time-varying filters, which are aptly described by time-varying ODEs. A typical time-varying ODE for a first-order is given by:

$$\frac{dy(t)}{dt} = a(t)y(t) + b(t)x(t) \tag{1.2}$$

In this equation, the coefficients $a(t)$ and $b(t)$ are time-dependent, reflecting the ever-changing attributes of the filter. This time-dependence introduces a layer of complexity, making the analysis and resolution of time-varying ODEs notably intricate. Techniques like Euler's method or Runge-Kutta methods often come to the rescue, with advanced mathematical tactics needed to unpack their behavior.

While employing Ordinary Differential Equations (ODEs) offers a robust mathematical framework for signal processing, integrating them with neural networks introduces unique challenges. These complexities arise mainly due to the sensitive

nature of ODEs, where slight alterations in inputs can produce substantially different outcomes. In this chapter, we delve into strategies designed to overcome these challenges during the neural network's training phase. Additionally, the task of generating functions with a neural network that align with the ODE solver network is not without its hurdles. A potential mismatch in function distributions could compromise the system's effectiveness. Our solution involves utilizing actual EEG signals as the network's input, ensuring alignment while preserving the signals' intricate dynamics. Furthermore, we highlight the importance of refining our function generator based on classification results. Although this iterative approach promises optimized generator functions, it also raises concerns related to computational efficiency and loss function definition. We present a comprehensive strategy to navigate these challenges, emphasizing distinct training phases and safeguarding the ODE solver network during parameter updates.

The remainder of this section is organized as follows: Section 3.2 presents our methodology, the challenges we encountered, and the detailed solutions. In Section 3.3, we provide a comprehensive overview of the experiments conducted and their corresponding results. Finally, Section 3.4 delves into an in-depth discussion and interpretation of our findings.

3.2 Methodology, Challenges, and Solutions

3.2.1 General Framework and Detailed Insights

The general framework of our approach is illustrated in Figure 3.1. Our framework comprises three main parts: the first is the η generator network; the second, known as the PDO neural network or the ODE solving network in subsequent sec-

tions, is designed for solving the ODE; and the third is the classifier.

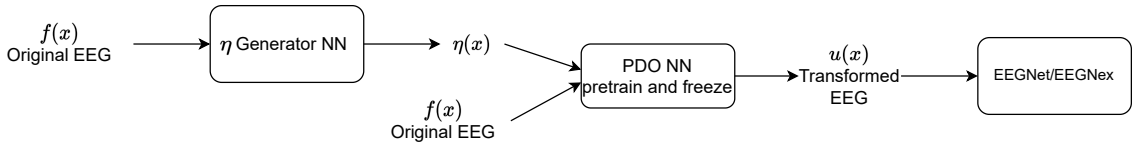


Figure 3.1: Comprehensive Layout of our Neural Network Structure. (NN: Neural Network)

PDO neural network (ODE solver Network): The ODE Solver Network we employed is based on the network structure detailed in [66]. This network can be conceptualized as one that seeks the inverse of an operator. In the discrete case, this translates to finding the inverse of a given matrix—a task challenging for standard neural networks. The neural network's operation can be represented by the following mapping:

$$\mathrm{M} : \eta \to G_\eta = L_\eta^{-1}$$

Given the above mapping, the solution can be derived as:

$$u = G_\eta f$$

An essential precondition for above process is that the form of L_η is already determined. using $\eta(x)$ and $f(x)$ as input, and output $u(x)$ as the solution of $L_\eta u(x) = f(x)$. The $L_\eta u(x)$ we used in this paper is:

$$L_\eta u(x) = \eta(x)u''(x) + u'(x) + u(x) \tag{2.3}$$

$\eta(t)$ generator Network: From examining Figure 3.1, we observe that in the discrete case, the input $f(x)$ and $\eta(x)$ must maintain the same length. To meet this requirement, we have devised a specialized neural network for generating η, which we refer to as the η generator neural network, depicted in Figure 3.2. The initial

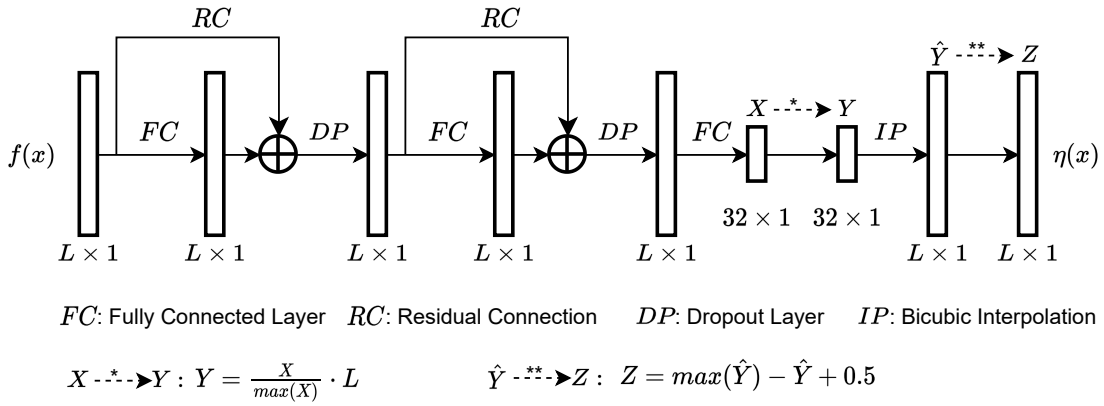


Figure 3.2: The layout of η generator NN

segment of this network structure comprises two residual blocks. Each block contains a fully connected layer, designed to facilitate the learning of complex patterns within the data. These blocks are then connected to another fully connected layer, serving a crucial role in information compression by distilling the essential features from the inputs.

Following this compression, the processed data is output as X. This output undergoes normalization, with each value divided by its maximum to ensure all values fall within the same scale. The normalized output is then scaled according to the length of the input $f(x)$, giving us Y. This Y then undergoes bicubic interpolation, a process that adjusts its length to match that of the original input $f(x)$, thus producing $\hat{Y}$. The final operation ensures the output values don't get too close to zero, by subtracting $\hat{Y}$ from its maximum value and adding 0.5. Across the network, we consistently employ the Rectified Linear Unit (ReLU) as the activation function and set a dropout rate of 0.2 to prevent overfitting.

3.2.2 Challenges and solutions

Learning to Solve the ODE: Learning to solve the ODE using a neural network poses its own unique set of challenges. The network is required to learn to approximate the solution of the differential equation over an extensive range of inputs. This task involves capturing the intricacies of complex, nonlinear dynamics. Moreover, the inherent nature of differential equations implies that even small changes in input or parameters can sometimes lead to substantial changes in the output. This can render the learning problem ill-conditioned, adding further complexity to the task [67].

In order to address these challenges, we employ a specific strategy during the training phase of the network. We carefully adjust the ranges of the randomly generated $\eta(t)$ and $f(x)$ to cover as wide a range of cases as possible. By doing so, we enable the network to learn from a diverse set of scenarios, thereby increasing its ability to generalize and respond to a wider range of inputs. This approach helps mitigate the difficulties presented by the nonlinear and potentially unstable nature of the problem at hand. The specifics of this method will be discussed in further detail in subsequent sections.

To train the PDO-Net, we also use central difference method with periodical boundary condition and let the integral of the output to be 1.

Generating $\eta(t)$ with Neural Network: Generating $\eta(t)$ with a Neural Network presents a unique set of challenges. The principal task requires the network to generate functions that can effectively serve as inputs for the ODE solver network. If the distribution of functions that the $\eta(t)$ generator network produces is not closely aligned with the distribution of functions that the ODE solver network has been trained on, the overall performance of the system could be significantly impaired

[68].

In addition to this, the selection of input values to this network is of importance. A straightforward approach, influenced by the idea of Generative Adversarial Networks (GAN), might involve using random values as inputs. However, such a methodology does not capture the time-dependent features intrinsic to specific EEG signals, creating a potential disparity in the generated $\eta(t)$.

To counteract these issues, we introduce an alternate strategy. In our proposed scheme, the EEG signals themselves, denoted as $f(x)$, serve as the input to our network. This tactic, while seemingly unconventional, allows the network to learn and reproduce the complex temporal dynamics embedded in the EEG signals. By using $f(x)$ as the input, we can ensure that the $\eta(t)$ generated by our network not only aligns more closely with the true distribution of functions, but also captures the necessary time-dependent features present in real-world EEG signals.

Updating the $\eta(t)$ Generator Using Classification Results: Updating the $\eta(t)$ Generator using classification results introduces a new set of challenges. The process necessitates backpropagation through the ODE solver network, a procedure that can be computationally expensive. Additionally, this step involves the intricate task of defining an effective loss function. This function needs to skillfully balance between the seemingly conflicting goals of accurate ODE solution and precise classification.

In response to these challenges, we propose a two-fold strategy. The first part of our solution places emphasis on an initial 'offline' training phase for the ODE solver network. This phase allows the network to be adequately trained prior to updating the parameters in the $\eta(t)$ generator. The second aspect of our solution concerns the parameter update process itself. During this process, the parameters within the ODE solver network are 'frozen,' effectively insulating them from the training of the $\eta(t)$

generator. This strategy helps to manage the backpropagation process, ultimately reducing the computational burden and enhancing the stability of the overall system.

3.3 Experiments

3.3.1 Simulation

In this section, we will demonstrate our approach through simulation, providing a more intuitive understanding of the transformed signals we generate. We pre-train our ODE solver network to solve equation 2.3 and then freeze it, then we simulate both η and EEG signals ($f(x)$), which are shown in Figure 3.3. Next, we use the generated η and $f(x)$ as input to the ODE solver network, we get the transformed EEG ($u(x)$). In this process, the operation in ODE solver network can be conceptualized as convolution in matrix form, which is a large diagonal matrix, where the kernel represents the non-zero values on each row of the diagonal, being applied (via left multiplication) to one-dimensional waveform data. In Figure 3.3, the brighter regions can be understood as the non-zero portions of the matrix. A narrower region indicates less smoothing. Consequently, from the narrower locations corresponding to the left multiplication, we can observe that the transformed signal retains more high-frequency components, implying that it preserves more noise.

3.3.2 Real-world dataset

We test our approach with two classifier EEGNet [69] and EEGNex [70] on two benchmark dataset EEG MNIST and Error-related Negativity (ERN).

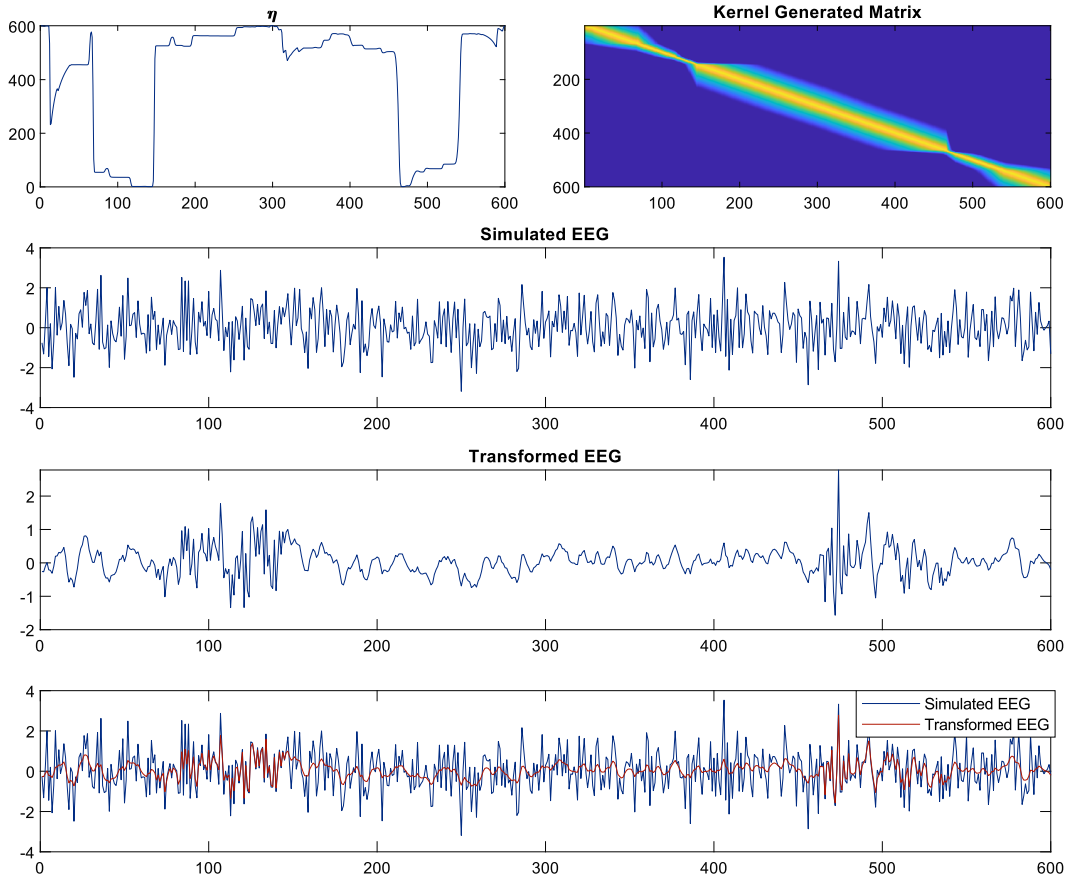


Figure 3.3: Simulated EEG and its transformation under a specific function η. The figure in the upper right corner shows the heatmap of the large diagonal matrix, where the non-zero values on each row are represented by the kernel.

3.3.2.1 EEG MNIST

The EEG MNIST dataset, assembled by David Vivancos, is an expansive open-source collection of over 1.2 million brain signals, each lasting two seconds. Captured using four commercial EEG devices, these signals are stimulated by the viewing of digit images (0-9), in a style akin to the traditional MNIST dataset, with any

unrelated or noise signals represented as -1. An intriguing characteristic of this dataset is the consistent phase synchrony exhibited across all channels for each class of digit stimuli.

Our study is primarily focused on the data derived from the Emotiv EPOC device, which records raw EEG signals at a sampling rate of 128Hz. Each digit image prompts around 6,500 trials, each encompassing 256 time steps across 14 channels. This device, along with the others used, captures signals from a range of 19 brain locations as per the 10/20 system, resulting in a comprehensive understanding of brain response patterns.

We've adopted common data preprocessing methodologies which involves removing noise events, applying a Butterworth lowpass filter with a cutoff frequency of 63Hz and a notch filter at 50Hz to all trials. We also trimmed the first 32 time steps from each trial to mitigate sensor power-on noise. The EEG MNIST dataset, therefore, provides an unaltered and detailed perspective of brain activity in response to digit stimuli, making it a critical asset for in-depth neural signal studies.

The accuracy outcomes using classifiers EEGNet and EEGNeX are depicted in Tables 3.1 and 3.2. Implementing our framework results in a noticeable boost in both accuracies.

Data	Classifier	Accuracy
EEG (bandpass)	EEGNet[69]	16.89%
Transformed EEG (our method)	EEGNet[69]	**19.81%**

Table 3.1: Accuracy comparison for EEG MNIST dataset using EEGNet as the classifier.

Data	Classifier	Accuracy
EEG (bandpass)	EEGNeX[70]	22.10%
Transformed EEG (our method)	EEGNeX[70]	**22.42%**

Table 3.2: Accuracy comparison for EEG MNIST dataset using EEGNeX as the classifier.

3.3.2.2 ERN dataset

The study at hand involves the "P300-Speller" paradigm, a renowned Brain-Computer Interface (BCI) application that utilizes Electroencephalography (EEG) data and leverages the P300 response, which is evoked by rare and attention-demanding stimuli on a computer screen. The paradigm operates by flashing screen items in groups and in a random sequence, where the item with the most likely recognizable typical target response is selected. This research focuses on identifying instances when the selected item is not the intended one, a determination made by analyzing the subject's brain signals post-feedback.

The dataset being used in this investigation is a 2-class Error-Related Negativity (ERN) data obtained from the BCI Challenge, hosted by Kaggle. This ERN feedback is crucial, as it contributes significantly to the enhancement of the P300 speller application's performance. The dataset, which consists of data from 26 healthy participants recorded via 56 passive Ag/AgCl EEG sensors, provides a rich source of information for this study. The data was initially recorded at 600Hz but was subsequently down-sampled to 128Hz and applied with a 1-40Hz notch filter for in-depth analysis.

In this competition, participants are tasked to develop and submit an Error Po-

tential detection algorithm. The proposed algorithm should be capable of detecting erroneous feedbacks in real-time and possess the ability to generalize across subjects, also known as transfer learning. The overarching aim of this task is to enhance the reliability and efficacy of BCIs by developing a sound error detection and correction strategy.

Tables 3.3 and 3.4 present a comparison of accuracy using the same classifiers (EEGNet and EEGNeX), both with and without the application of our framework. It is evident that the accuracies improve after implementing our framework.

Data	Classifier	Accuracy
EEG (bandpass)	EEGNet[69]	72.13%
Transformed EEG (our method)	EEGNet[69]	**76.65%**

Table 3.3: Accuracy comparison for ERN dataset using EEGNet as the classifier.

Data	Classifier	Accuracy
EEG (bandpass)	EEGNeX[70]	74.83%
Transformed EEG (our method)	EEGNeX[70]	**78.49%**

Table 3.4: Accuracy comparison for ERN dataset using EEGNeX as the classifier.

3.4 Discussion

In our research, inspired by the principles of meta-learning pseudo-differential operators using deep neural networks [66], we leverage a neural network as an Or-

dinary Differential Equations (ODE) solver. Central to our methodology is the pseudo-random generation of the time-dependent coefficient $\eta(t)$ and the forcing function f(t), together creating diverse ODE instances. Employing the finite difference method, we determine the solution y(t) for each respective pair. After training our neural network on these datasets, it effectively learns the interrelationship between $\eta(t)$, f(t), and y(t). Consequently, the network can predict y(t) for any $\eta(t)$ and f(t) pair. This neural-based approach to ODE solving not only handles the complexities within the ODE efficiently but also benefits from backpropagation for error refinement. This process, in turn, refines our $\eta(t)$ generator network, enhancing the signal-to-noise ratio crucial for EEG signal processing and classification. Based on our experiments conducted on real-world datasets, we observed a significant enhancement in accuracy.

4 A Hybrid Neural Network Structure

4.1 Introduction

EEGs are used in many health care institutions to monitor and record electrical activity in the brain using electrodes placed on the scalp. Analysis of EEG recordings is a crucial first step to making a clinical diagnosis of epilepsy, severity of anesthesia, coma, encephalopathy, and brain death [71]. In addition, EEG can be used as input for a brain-computer interface (BCI) system to control a device such as a cursor or robotic limb [72]. Accurate and fast detection of EEG signals is crucial for not only disease diagnosis but also ensuring the optimal performance of BCIs. Thus automatic detection systems based on machine learning algorithms have been developed to evaluate and classify EEG signals [73, 74, 75].

Due to the specific properties of EEG signals, such a classification task can be more broadly described as non-stationary signal classification. Traditionally, in non-stationary signal classification, the first step is to extract the useful features and then use classifiers like Support Vector Machine(SVM), K-Nearest Neighbors(KNN), Regression, and so on to differentiate classes [76, 77, 78, 79]. Many techniques have been proposed for EEG signal classification. Hassanpour *et al.* [80] use modified B-distribution to characterize low-frequency EEG patterns and apply singular value decomposition (SVD) on the time-frequency distribution matrix to detect seizures in

newborns and obtain encouraging results. In [81], Tzallas *et al.* extract features using Cohen's class Time-Frequency Representation (TFR) and power spectrum density, then use Artificial Neural Networks as the classifier to identify epileptic seizures in three benchmark EEG datasets. Boashas *et al.* [82] use quadratic time-frequency distributions (TFDs) with extended features and matrix decomposition with SVM as the binary classifier to detect newborn EEG abnormalities. In [83], the authors use wavelet energy and wavelet entropy as features and KNN as the classifier. In [84], the authors propose exponential energy as a new feature, and combine it with other commonly used entropy and energy features, then use SVM to classify epileptic EEG signals.

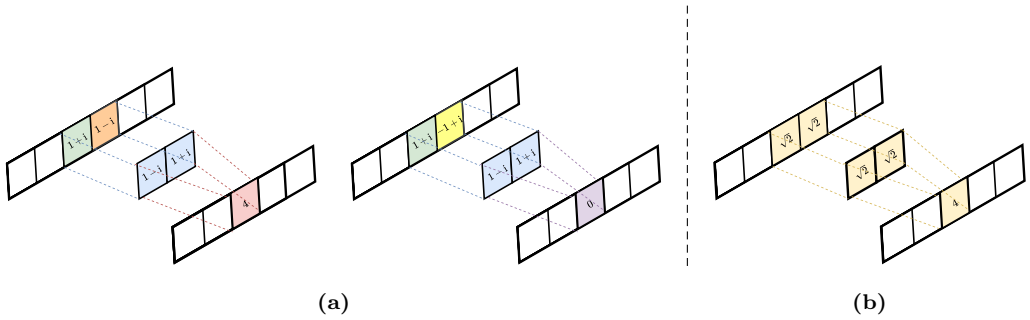


Figure 4.1: The convolution operation (a) on complex numbers, (b) on modulus (amplitude) only.

In the recent decade, it has been shown that methods based on deep learning can yield better performance than conventional methods [85, 86, 87, 88, 57]. Most of these methods utilize real-valued signals in the algorithms. In [85], the authors use the single-sided amplitude spectrum as input to a CNN. In [57], the authors concatenate real and imaginary parts of complex spectrum and use it as the input of a CNN. However, in frequency domain, the signals are represented as complex values. Such a complex representation contains the power of a specific wave relative to others (amplitude) and the alignment of different frequency components in

time (phase). Using only the amplitude information will lose the phase information, which might be critical for EEG signals. Figure 4.1 illustrate the core idea of our argument. In Figure 4.1a, we can see that these two convolution operations use the same convolution kernel, and all the complex numbers have the same modulus, which is $\sqrt{2}$; however, the convolution results are different. If we only use the modulus in this convolution operation (See Figure 4.1b), we can only get one result, which is 4. Suppose the phase is the only difference between two signals; using amplitude only as the feature can not differentiate them properly.

We, therefore, propose an algorithm that can utilize the features hidden in the phase information. To achieve this goal, we may train a real-valued convolutional neural network whose inputs are the amplitude and the phase as two channels or the real part and the imaginary part of the complex number as two channels. However, the traditional convolution operation contains a linear combination of the channels. It is unclear if such a linear combination is meaningful[52].

With these issues in mind, we believe that using the original complex values of DFT as input is a better alternative. In [89], the authors also mentioned that complex-valued neural networks might be suitable for extracting phase information. Therefore, we exploit a complex-valued neural network for our case, more specifically complex-valued convolutional neural network. In [90], a complex-valued convolution neural network was introduced and compared with a real-valued convolutional neural network on the classification tasks. In [91], complex-valued convolutional neural network was applied to analyze steady-state visual evoked potential (SSVEP) dataset and the results are very promising. Another notable paper is [92], where the authors applied deep complex networks to audio-related tasks and achieved promising results; this work also presented batch-normalization and complex weight initialization strategies for complex-valued neural networks. In one of our experiments,

we observe that the performance is not improved if all the layers in the network are complex-valued. The first possible reason is the complex-valued non-linear activation function is task-specific; an inadequate selection of activation function may lead to poor transmission of the information between layers[52, 93, 94]. The second possible reason is that in the loss function of a complex-valued neural network, one usually needs to calculate the "distance" between complex numbers and real numbers, which is not well-defined in mathematics.

Therefore, we develop an algorithm that integrates the real-valued and complex-valued neural networks to overcome the difficulties mentioned above. Our approaches builds on the network structures of San-Segundo *et al.* [85] and Ravi *et al.* [57]; both of these two structure contains two convolutional layers for feature extraction and fully connected layers for classification. We improve their neural network structures by adding a crucial complex-valued convolutional layer at the beginning or changing the very first real-valued convolutional layer into complex-valued, and immediately taking the modulus as non-linear activation (See Section 4.2.1 and Section 4.3.2.3 for details). There are four main advantages of our network architecture: 1) The complex-valued convolutional layer captures the features in the phase information of complex-valued input; 2) we only need one universal complex-valued activation function, which is taking the modulus; 3) The difficulty in distance calculation between complex numbers and real numbers in the loss function is avoided in our framework; 4) Our network uses about 50% fewer parameters compared with the structure in [85]. Moreover, our framework can achieve higher classification accuracies than the conventional feature selection method and the CNN in [85, 57] in both the experiments using a benchmark dataset.

We apply the proposed method to the EEG signal classification problem with discrete Fourier transform (DFT). We present qualitative results on several classifi-

cation tasks, including binary and multi-class classification on two simulated datasets and two real-world dataset. The simulated datasets consist of synthetic signals of our own design and signals generated based on well-known theories. The real-world datasets are the Ralph-Andrzejak EEG dataset [95] and SSVEP dataset [74].

The rest of this paper is structured as follows: Section 2 contains the methodology in this study, including a description of the discrete Fourier transform (DFT), the framework of our neural network, the backpropagation algorithm, and some training details. Section 3 describes the experiments and the results obtained in simulation and a real-world dataset. Section 4 summaries this paper and discusses the limitations of our method and our future works

The symbols and notations used in this paper are summarized in Table 4.1

$\mathbb{R},\ \mathbb{C}$	Sets of real, complex numbers
i	Imaginary unit
$z^\intercal,\ z^*,\ z^H$	Transpose, conjugate, conjugate transpose of z
$\mathfrak{Re}(z)$	Real part of z
$abs(z), \lvert z\rvert$	Absolute value, modulus of z

Table 4.1: SYMBOLS AND NOTATIONS

4.2 Methodology

In this section, we present an overview of our methodology, including the framework of our network, a description of discrete Fourier transform, the backpropagation

algorithm, and some training details.

4.2.1 Framework

The neural network frameworks analyzed in this study are based on adding a complex-valued layer or changing a real-valued layer into complex-valued. Figure 4.2 and Figure 4.13 show our frameworks. Both of these two structures use complex-valued spectrum as input and use complex-valued convolutional layer as first layer. We believe this first complex-valued convolutional layer can extract the features in the phase information in the spectrum.

Here, we introduce the details of the structure shows in Figure 4.2, because we use this structure on the analysis of the simulated dataset. Details about the structure in Figure 4.13 will be introduced in section 4.3.2.3. In the structure in Figure 4.2, the first layer we use is a complex-valued convolutional layer. Immediately after this layer, we take the modulus (see the part in the dotted line box in Figure 4.2), making all the outputs real-valued. After this, we add two real-valued convolutional layers and three fully connected layers with ReLu and max-pooling. Finally, we use softmax as the last activation function and cross-entropy as the loss function.

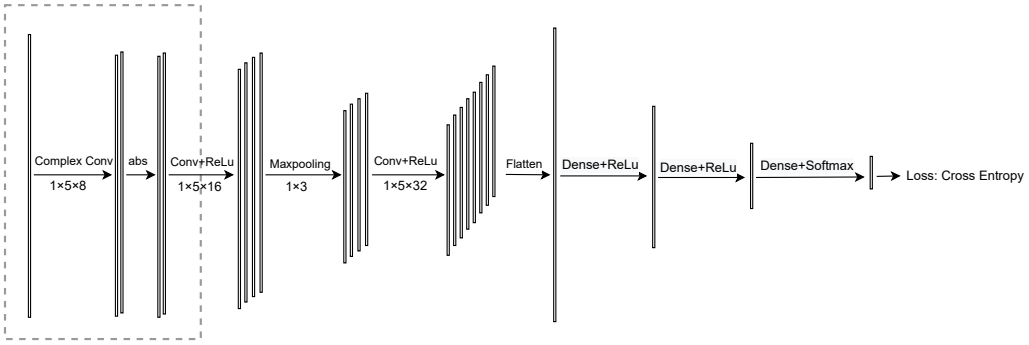


Figure 4.2: Our neural network structure.

4.2.2 Discrete Fourier Transform and Its Inverse Transform

We apply discrete Fourier transform (DFT) on the original EEG signals to obtain their representations on the frequency domain and apply inverse discrete Fourier transform (IDFT) to achieve the inverse transform. The formulas of DFT and IDFT are shown below:

$$DFT: \quad \hat{x}(k) = \sum_{j=1}^{n} x(j)\omega_n^{(j-1)(k-1)}, \ k = 1 \ ... \ n \tag{2.1}$$

$$IDFT: \quad x(j) = \frac{1}{n}\sum_{k=1}^{n} \hat{x}(k)\omega_n^{-(j-1)(k-1)}, \ j = 1 \ ... \ n \tag{2.2}$$

where x is the original signal of length n, $\hat{x}$ is the Fourier transform of x of length n, and $\omega_n = e^{-2\pi \mathrm{i}/n}$. This paper implements the DFT and IDFT using MATLAB command $fft()$ and $ifft()$. Because our original signals are all real-valued, their DFTs are conjugate symmetric. We only keep the first half part of the DFTs as the input to our neural network.

4.2.3 Backpropagation

As we can see from Figure 4.2, our framework contains a complex-valued convolutional layer taking modulus as activation, and after this, all the layers are real-valued. We use the Adam algorithm [96] with default settings to optimize the kernel and bias in real-valued layers. To optimize the parameters in the complex-valued convolutional layer, we also need the backpropagation algorithm in the complex value. The regular complex derivative only applies to the analytic functions [93]; however, to obtain the modulus need to evaluate the following the function:

$$f(Z) = |Z| = (Z^* Z)^{\frac{1}{2}} \tag{2.3}$$

which is not analytic. So in backpropagation, the derivative of $f(Z)$ with respect to Z can not be calculated with a regular complex derivative. In this case, we need to adopt the Wirtinger derivative.

4.2.3.1 Wirtinger derivative

The following defines the Wirtinger derivatives:

Definition 4.2.1 *Consider the complex plane* $\mathbb{C} \equiv \mathbb{R}^2 = \{(x, y) \mid x, y \in \mathbb{R}\}$. *The two operators* $\frac{\partial}{\partial z}$ *and* $\frac{\partial}{\partial z^*}$ *are defined by:*

$$\begin{aligned} \frac{\partial}{\partial z} :=& \frac{1}{2}\left[\frac{\partial}{\partial x} - \mathrm{i}\frac{\partial}{\partial y}\right], \\ \frac{\partial}{\partial z^*} :=& \frac{1}{2}\left[\frac{\partial}{\partial x} + \mathrm{i}\frac{\partial}{\partial y}\right]. \end{aligned} \tag{2.4}$$

are referred to as the Wirtinger derivatives [58].

Wirtinger derivative holds standard rules for differentiation known from real-valued analysis concerning the sum, product, and composition of two functions. Then from equation (2.4), we can derive another essential property of Wirtinger derivative:

$$\frac{\partial z}{\partial z^*} = 0 \ , \qquad \frac{\partial z^*}{\partial z} = 0,$$

which means we can treat z and z^* as independent variables. Then based on the first-order Taylor expansion for multivariable functions, we have:

$$df = \frac{\partial f}{\partial z} dz + \frac{\partial f}{\partial z^*} dz^* + \mathcal{O}(|dz|^2) \tag{2.5}$$

To further derive the backpropagation algorithm based on the Wirtinger derivative, we need the following Corollary.

Corollary 4.2.1 *Derivatives of the conjugate function $f^*(z)$ satisfy the following relationships:*

$$\frac{\partial f^*(z)}{\partial z} = \left(\frac{\partial f(z)}{\partial z^*}\right)^* \qquad \frac{\partial f^*(z)}{\partial z^*} = \left(\frac{\partial f(z)}{\partial z}\right)^*. \tag{2.6}$$

It is straightforward to derive equation (2.6) from Definition 4.2.1.

In our case, $f(z) : \mathbb{C} \to \mathbb{R}$, which means $f(z) = f^*(z)$, equation (2.6) can be further simplified to the following equations:

$$\frac{\partial f(z)}{\partial z} = \left(\frac{\partial f(z)}{\partial z^*}\right)^* \qquad \frac{\partial f(z)}{\partial z^*} = \left(\frac{\partial f(z)}{\partial z}\right)^*. \tag{2.7}$$

From equations (2.5) and (2.7), by omitting the lower order term, we then have:

$$\begin{aligned} df &= \frac{\partial f}{\partial z}dz + \frac{\partial f^*}{\partial z^*}dz^* \\ &= \frac{\partial f}{\partial z}dz + \left(\frac{\partial f}{\partial z}dz\right)^* \\ &= 2\mathfrak{Re}\left(\frac{\partial f}{\partial z}dz\right) \end{aligned} \tag{2.8}$$

Based on the principles of the gradient descent method, we need to find dz that maximize $\left|2\mathfrak{Re}\left(\frac{\partial f}{\partial z}dz\right)\right|$. From Cauchy–Schwarz inequality, we know that dz should have the same direction of $\left(\frac{\partial f}{\partial z}\right)^*$. So the direction of the steepest ascent is the direction of $(\frac{\partial f}{\partial z})^*$(if $f(z) : \mathbb{C} \to \mathbb{R}$, from Equation 2.7, we know that $\frac{\partial f}{\partial z^*} = (\frac{\partial f}{\partial z})^*$).

We can give the general form of the backpropagation algorithm for CVNN:

$$W^{(t+1)} = W^{(t)} - \eta \left(\frac{\partial Loss(W^{(t)})}{\partial W^*}\right)^\intercal \tag{2.9}$$

where $W^{(t)}$ is the set of all the parameters that need to be learned in the complex-valued layers at t'th iteration. η is the learning rate. We adopt automatic differentiation in Tensorflow [61] to calculate the gradients for complex-valued layers based on the Wirtinger derivative [61]. We use complex Adam [97] as the specific backpropagation algorithm to optimize the parameters.

4.2.3.2 An example of the gradients calculation in complex-valued convolution layer

Suppose we have an input sequence data Z and a complex convolution kernel k (See Figure 4.3). Z_i is an arbitrary part of Z, and $Z_i^\intercal k$ is the convolution result. Note that here $Z_i^\intercal k$ is not the Hermitian inner product of two complex-valued vectors. After adding the bias term b, Y_i can be defined as:

$$Y_i = abs(Z_i^\intercal k + b) = [(Z_i^\intercal k + b)^*(Z_i^\intercal k + b)]^{\frac{1}{2}} \tag{2.10}$$

From the backpropagation for the real-valued CNN, we can get $\partial Loss/\partial Y_i$. Then based on the Wirtinger derivative, we need to find $\partial Loss/\partial k^*$ and $\partial Loss/\partial b^*$. Based on the chain rule, we have:

$$\frac{\partial Loss}{\partial k^*} = \sum_i \frac{\partial Loss}{\partial Y_i}\frac{\partial Y_i}{\partial k^*} = \frac{1}{2}\sum_i \frac{\partial Loss}{\partial Y_i}(Z_i^\intercal k + b)Z_i^H \tag{2.11}$$

Similarly,

$$\frac{\partial Loss}{\partial b^*} = \sum_i \frac{\partial Loss}{\partial Y_i}\frac{\partial Y_i}{\partial b^*} = \frac{1}{2}\sum_i \frac{\partial Loss}{\partial Y_i}(Z_i^\intercal k + b) \tag{2.12}$$

Based on (2.11) and (2.12), we can finally find the proper gradients for k and b.

4.2.4 Other training details

For the real-valued weights and bias, we use Xaiver initialization [98]. For the complex-valued weights and bias, we use the Rayleigh distribution to generate the modulus of the complex number (r) and the Uniform distribution ($U_{[-\pi,\pi]}$) to generate the angle (θ). Then we can get the initialization for complex-valued parameters by using the formula $re^{\mathrm{i}\theta}$.

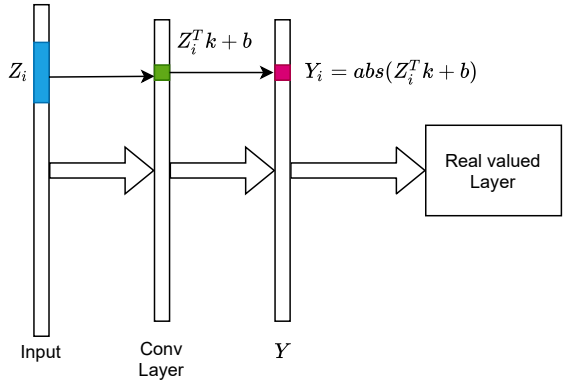


Figure 4.3: An example of the forward propagation in the complex-valued convolutional layer. Z_i: a part of the input, k: complex-valued convolution kernel, b: complex-valued bias, Y_i: the modulus of the output of the complex-valued convolutional layer, Y: the vector whose i'th entry is Y_i. Y is the input to the next real-valued layer.

4.3 Experiments

In this section, we compare our method against two existing frameworks on the classification task with two simulated datasets. We then apply our approach to a real-world dataset and compare it with several previous works.

4.3.1 Simulation Study

In this study, we simulate EEG signals with two different methods to compare the classification performance between our method and other methods. In the first simulation, we adopt the first-order autoregression (AR(1)) model to generate the amplitude and phase separately. We then use the inverse discrete Fourier transform (IDFT) to obtain the signals on the time domain. The main difference among signals in different classes in this simulation is the phase. We want to use this simulation to prove that our algorithm can efficiently utilize the features in phase. In the second

simulation, we adopt classical theory and phase resetting theory to generate event-related potential (ERP) with noise [99, 100], and then we design four classification task (See Table 4.3). There are two reasons we perform the second simulation: 1) the signals in the second simulation are closer to the real EEG signals, 2) ERP-signal classification is crucial in analyzing human EEG activities and can be a promising tool to study error detection and observational-learning mechanisms in performance monitoring [101].

We mainly compare our method with real-valued CNN and the conventional feature selection method in these two classification tasks. For real-valued CNN, the network structure we choose to compare with is the structure used in [85], which contains two convolutional layers and three fully connected layers. To compare with the feature selection method, we choose seven features, which are Shannon entropy [102], Renyi entropy [103], log-energy entropy [103], approximate entropy [104], sample entropy [105, 106], fuzzy entropy [107, 108, 109], and exponential energy [84](See Table 4.2 for detailed parameter settings.). In both simulations, we applied 6th order Butterworth low pass filter to remove the frequencies over 60 Hz before extracting the features. We try all 127 combinations of these seven features. For each combination of the features, we extract them from the pre-processed signal, the first and second order difference of the pre-processed signal. So the number of features we select is always a multiple of 3. The classifier we choose is the support vector machine(SVM) for the binary classification task and the error-correcting output codes (ECOC) model using SVM binary learners[110, 111] for the multi-class classification task.

In this simulation study, we use 5-fold cross-validation to obtain accuracy. To alleviate the accuracy variation, we repeat the 5-fold cross-validation ten times for each classification task to get the average and standard deviation of the accuracies.

Features	Parameters	Ref.
Renyi Entropy	$\alpha = 0.5$	–
Approximate Entropy	$m = 2, \tau = 1, r = 0.2 * sd$	[105]
Sample Entropy	$m = 2, \tau = 1, r = 0.2 * sd$	[105]
Fuzzy Entropy	$m = 3, \tau = 3, r = 0.15 * sd$	[112]

Table 4.2: Parameter details. α: order of Renyi entropy, m: embedding dimension, τ: time delay, r: threshold value to determine similarity, sd: the standard deviation of the input time-series data.

The accuracies presented in this simulation study are based on the results with the highest mean accuracy.

4.3.1.1 EEG signals generated with AR(1) model

In this section, we simulate the signals using AR(1) model because we assume that neighboring amplitude and phase are not independent. The AR(1) model assumes that the current value depends linearly on its immediately prior value and a stochastic term, which complies with our assumptions. The formula for AR(1) model is shown in Equation (3.13):

$$x_t = \beta_0 + \beta_1 x_{t-1} + \epsilon_t \tag{3.13}$$

where x_t is the present value, x_{t-1} is the immediately prior value, β_0 is a constant, β_1 is the model parameter, and ϵ_t is the white noise with zero mean and constant variance.

We first simulate the phase information. We know that the phase $\theta \in [-\pi, \pi]$, so we modify the Equation (3.13) to ensure the range of θ is limited. The modified

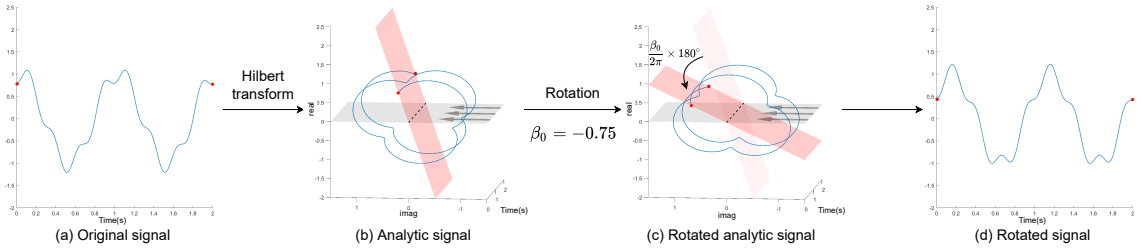


Figure 4.4: The effect of β_0: (a) the original signal, (b) the analytic signal obatined by applying Hilbert transform on the original signal. (If we observe the analytic signal (b) in the direction of the arrow, we can see the original signal in (a).) (c) the rotation of the analytic signal by β_0. (d) the rotated signal (If we observe the rotated analytic signal (c) in the direction of the arrow, we can see the signal in (d).)

formula is shown in Equation (3.14):

$$\theta_t = Rem(\beta_0 + \beta_1\theta_{t-1} + \epsilon_t, \pi) \tag{3.14}$$

where $Rem(\star, \pi)$ is a function used to obtain the remainder of $\star$ divided by π. Here, β_0 achieves an overall phase shift, and its effect can be understood as a rotation of a signal under the Hilbert transform (See Figure 4.4). Suppose we have the Hilbert transform of a real-valued signal. In that case, we can plot the analytic form of the signal in three-dimensional Cartesian coordinates with the time axis, the real part axis, and the imaginary part axis. Then we can rotate this analytic form of the signal about the time axis(dashed line in Figure 4.4 (b), (c)) to achieve the effect of β_0.

From equation (3.14), we can see that, to simulate the phase, we need to determine three parameters — β_0, β_1, and the variance of ϵ_t. Since β_1 determines the correlation between the θ_t and θ_{t-1} and the variance of ϵ_t determines the intensity of the noise, we use different β_0 as the baseline to generate signals for different classes (See the middle column in Figure 4.6). Because we want to compare our method with other methods in multi-class classification, we generate signals for five classes

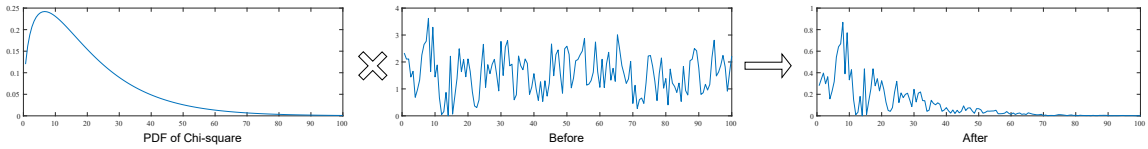


Figure 4.5: Simulated single-sided amplitude spectrum: the left graph shows the probability density function of the Chi-squared distribution, the middle graph is the single-sided amplitude spectrum generated with the unmodified AR(1) model, and the right graph shows the single-sided amplitude spectrum multiplied with the probability density function of the chi-squared distribution.

by setting β_0 as $0, \pm 0.5, \pm 1$ (corresponds to the different angle of rotation in Figure 4.4). To show the effect of β_1 and the variance of ϵ_t for different classes, we let β_1 and $Var(\epsilon_t)$ be ranged from 0.1 to 0.9 with an interval of 0.1. The accuracy table in Figure 4.7 contains nine by nine values, and each value corresponds to a specific pair of β_1 and $Var(\epsilon_t)$.

After we obtain the simulated phase, we use unmodified AR(1) model (3.13) with $\beta_0 = 0, \beta_1 = 0.5, Var(\epsilon_t) = 0.5$ to randomly generate the amplitude for different groups. We then multiplied the single-sided amplitude spectrum by the Chi-square distribution to make our simulated signals have the main bandwidth appears in the range of 0 to 70 Hz (see Figure 4.5), which is close to the bandwidth used by clinical analysis of EEG [113]. Then we adopt IDFT to obtain the simulated signals on the time domain using the simulated phase and amplitude (see Figure 4.6). Finally, the simulated signals last for 1.5 seconds with a 200 Hz sampling frequency.

Figure 4.7 shows the classification results using this simulated dataset. As shown in Figure 4.7a, it is not surprising that applying CNN on phase-only data can achieve the highest accuracy no matter the value of β_1 and $Var(\epsilon)$. Because based on our simulation, all the differences among different classes are reflected in the phase information, and the random variation in amplitude can be viewed as noise. We also apply

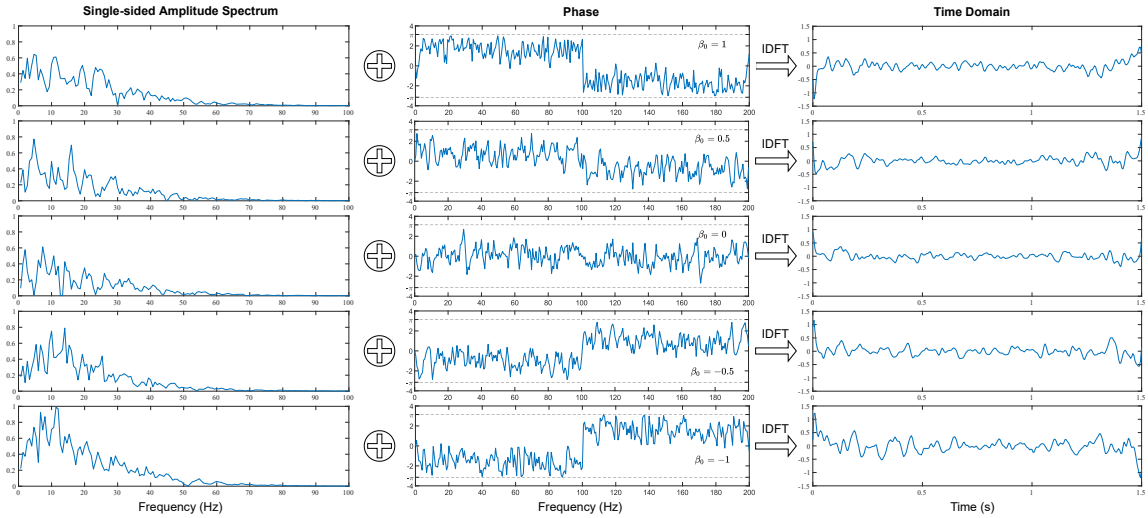


Figure 4.6: The first column shows the single-sided amplitude spectrum generated using AR(1) model with $\beta_0 = 0, \beta_1 = 0.5, Var(\epsilon_t) = 0.5$, the second column shows the simulated phase information generated using modified AR(1) formula with $\beta_0 = 0, \pm 0.5, \pm 1, \beta_1 = 0.5, Var(\epsilon_t) = 0.5$ and the last column shows the simulated signals generated from the IDFT of the amplitude spectrum and phase.

CNN and the feature selection method to the simulated signals in the time domain for comparison (see Figure 4.7b and 4.7c). Figure 4.7d shows the result obtained by using our algorithm. Figure 4.8 shows the accuracy differences between our method and the two methods for comparison. We can see our method outperforms other methods in most cases. We also show the confusion matrices in the lower part of Figure 4.8. These four confusion matrices are selected when the accuracy difference achieves the largest($Var(\epsilon) = 0.8,\ \beta_1 = 0.1$ and $Var(\epsilon) = 0.5,\ \beta_1 = 0.9$).

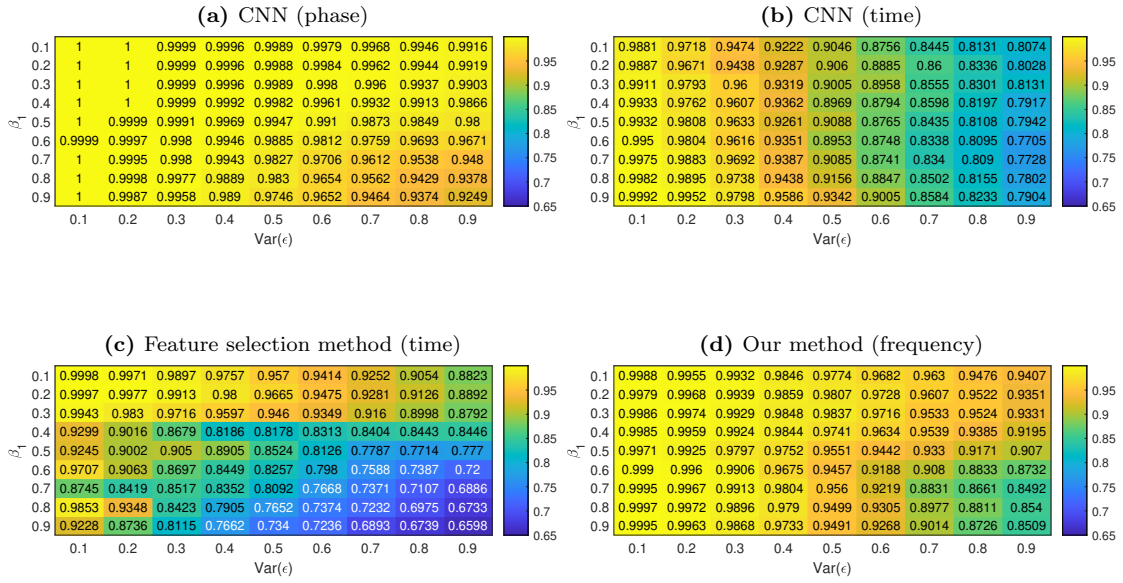


Figure 4.7: (a) the accuracies of using CNN with phase information only, (b) the accuracies of using the CNN with the input of simulated signals (time domain), (c) the accuracies of applying the feature selection method, (d) the accuracies of using our method on simulated signals (frequency domain).

4.3.1.2 EEG signals generated according to classical theory and phase resetting theory

In this simulation, we generate EEG signals for the classification task according to two main theories: classical theory and phase-resetting theory [99, 100]. The former theory assumes that the ERP signal is buried in the ongoing EEG noise, while the latter theory believes that the events can reset the phase of ongoing oscillations. With each theory, we design two binary classification tasks, referred to as the "Fixed location" task and the "Random location" task (see Table 4.3), to compare our method and other methods.

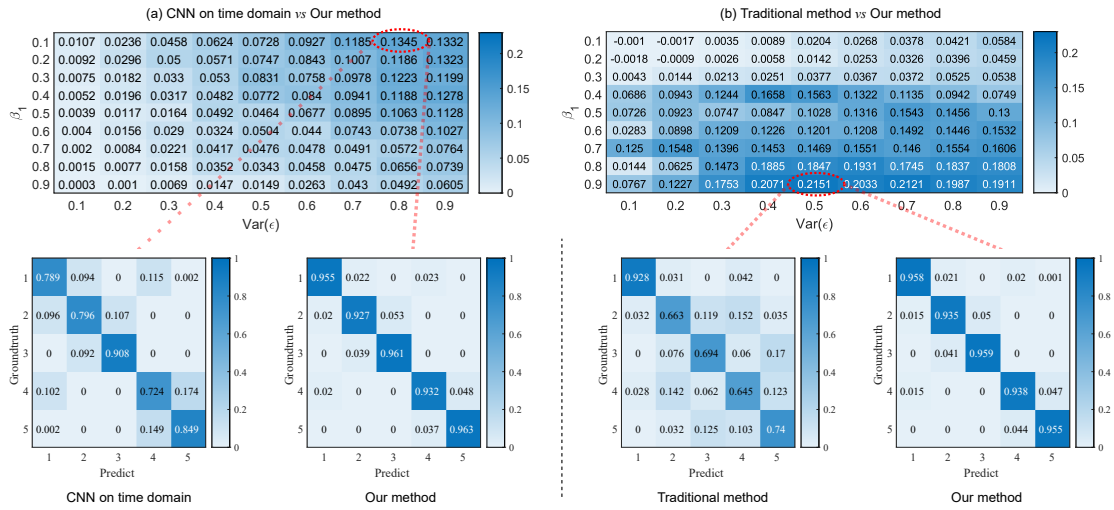


Figure 4.8: (a): the accuracy differences between our method and applying CNN on the time domain. (b): the accuracy differences between our method and the traditional method. From (a), we can see that when $Var(\epsilon) = 0.8$ and $\beta_1 = 0.1$, our method can outperform CNN on the time domain the most. As shown in (b), our method can outperform traditional method the most when $Var(\epsilon) = 0.5$ and $\beta_1 = 0.9$. The bottom figures are the confusion matrices when the accuracy difference achieves the largest (labels 1 to 5 correspond to $\beta_0 = 0,\ 0.5,\ 1,\ -0.5,\ -1$, respectively).

	classical theory	phase resetting theory
Fixed loc	Fixed peak loc + noise ***vs*** noise only	Fixed phase resetting loc + noise ***vs*** no phase resetting + noise
Random loc	Random peak loc + noise ***vs*** noise only	random phase resetting loc + noise ***vs*** no phase resetting + noise

Table 4.3: The four binary classification tasks we used to compare our approach and other methods.

In the simulation based on the classical theory, we first randomly generated 10,000 pieces of noise signals with the same power spectrum of human EEG signals using the MATLAB code downloaded from [114]. Each piece of noise signal lasts for 2 seconds with a sampling rate of 150 Hz. Then 5000 pieces of these noise signals were randomly selected to add a peak signal. In the experiment of the "Fixed location," we added

5 Hz peak signal centered at the middle of each piece of noise (see Figure 4.9), and in the experiment of the "Random location," we change the location of the center of the peak to be uniformly random distribute on the interval of 0 to 2 seconds.

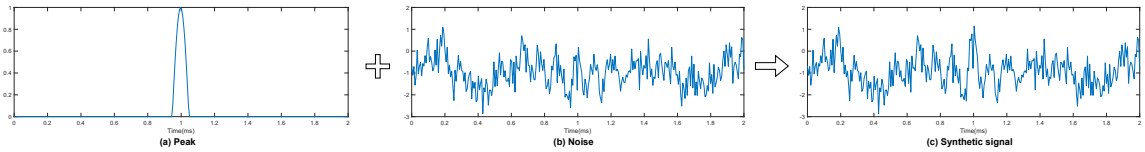


Figure 4.9: An example of the peak, noise, and synthetic signal generated according to classical theory. (a) Peak: the highest value of the peak signal shows exactly at 1s with the frequency of 5 Hz. (b) Noise: Randomly generated based on human EEG background signal spectrum. (c) Synthetic signal: the weighted summation of peak signal and noise signal.

In the simulation guided by phase resetting theory, we randomly generate 5000 signals with phase resetting and 5000 signals without phase resetting. For the phase resetting group, we generate simulated data by summing four sinusoids with frequencies chosen randomly from the range 4 to 16 Hz as in [115]. In the experiment of "Fixed position," each of these four sinusoids contains phase resetting at the center (see Figure 4.10). In the experiment of "Random position," each of these four sinusoids contains phase resetting at the same random position (uniformly distributed on the interval of 0 to 2 seconds). Then we add randomly generated human EEG background noise to all the signals.

We mainly compare our method with the approach that applies CNN on phase information only, amplitude information only, and the original signal. We also separated the real and imaginary parts of the FFT and applied the exact structure of CNN to each of them. Furthermore, we also apply CNN with the input of real and imaginary parts as two channels. The result shows in Table 4.4. We can see that our method can outperform others by comparing the accuracy.

		Method	classical theory	Phase resetting(with noise)
Fixed	Traditional	Features + classifier	0.6464±0.0010	0.6403±0.0012
	CNN	Phase only	0.9758±0.0018	0.9410±0.0042
		Amplitude only	0.9422±0.0023	0.8640±0.0113
		Original signal	0.9222±0.0022	0.9025±0.0021
		Real part only	0.9801±0.0013	0.9113±0.0028
		Imaginary part only	0.6349±0.0096	0.8553±0.0033
		Real+Imaginary(two channels)	0.9593±0.0049	0.9305±0.0017
		Our method	**0.9953±0.0009**	**0.9679±0.0103**
Random	Traditional	Features + classifier	0.6504±0.0010	0.6807±0.0010
	CNN	Phase only	0.9273±0.0029	0.9043±0.0022
		Amplitude only	0.8923±0.0056	0.7629±0.0086
		Original signal	0.8727±0.0071	0.5371±0.0049
		Real part only	0.9488±0.0020	0.8840±0.0070
		Imaginary part only	0.9431±0.0027	0.8690±0.0067
		Real+Imaginary(two channels)	0.9162±0.0113	0.7926±0.0067
		Our method	**0.9930±0.0012**	**0.9369±0.0144**

Table 4.4: Accuracy comparison.

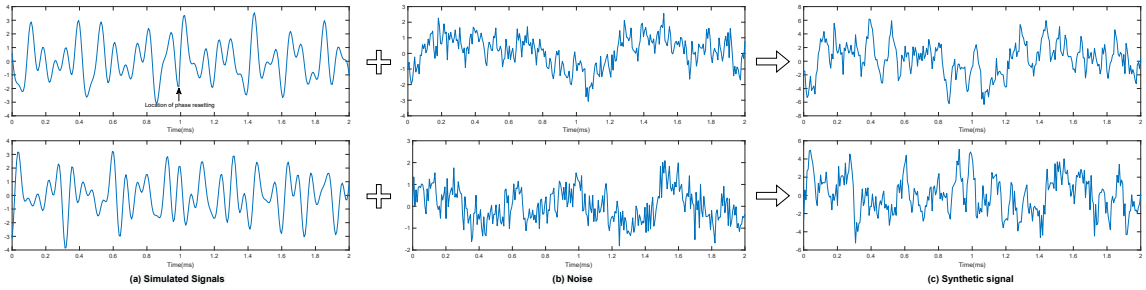


Figure 4.10: An example of the signals generated according to phase resetting theory. (a) an example of simulated signals with (top) and without (bottom) phase resetting. (b) Randomly generated noise based on human EEG background signal spectrum. (c) the synthetic signals, which are the weighted summation of signals in (a) and (b).

4.3.2 Real-World Data

In this section, we analyze two real-world datasets; the first one is Ralph-Andrzejak EEG dataset [95] and the second one is a public SSVEP dataset [74].

4.3.2.1 Ralph-Andrzejak EEG dataset

The first real-world dataset analyzed here is the Ralph-Andrzejak EEG dataset [95], which contains five categories of signals named Z, O, F, N and S, respectively (see Table 4.5). Each category contains 100 EEG records of about 23.6 seconds with a sampling rate of 173.61 Hz (4097 values per record). Although there are 5 classes in this dataset, the four most common classification tasks for this dataset are Z vs S [84], S vs NS (NS=Z+O+F+N) [116, 117], Z vs N vs S [118] and Z vs F vs S [119].

To obtain a comparable result, we use the pre-processed dataset in [120, 121]. In this dataset, the original signals of 23.6 seconds were divided into 23 non-overlapping segments. Each segment contains 178 values(about 1 second). Totally, there are $5 \times 100 \times 23 = 11,500$ pieces of segmental signals. Figure 4.11 shows five example

Category	Description
Z	Healthy group, recorded with eye open
O	Healthy group, recorded with eye closed
F	Interictal activity, recorded in the epileptogenic zone
N	Interictal activity, recorded at hippocampal location
S	Seizure activity

Table 4.5: Detailed description of the 5 categories of EEG signals in the Ralph-Andrzejak EEG dataset

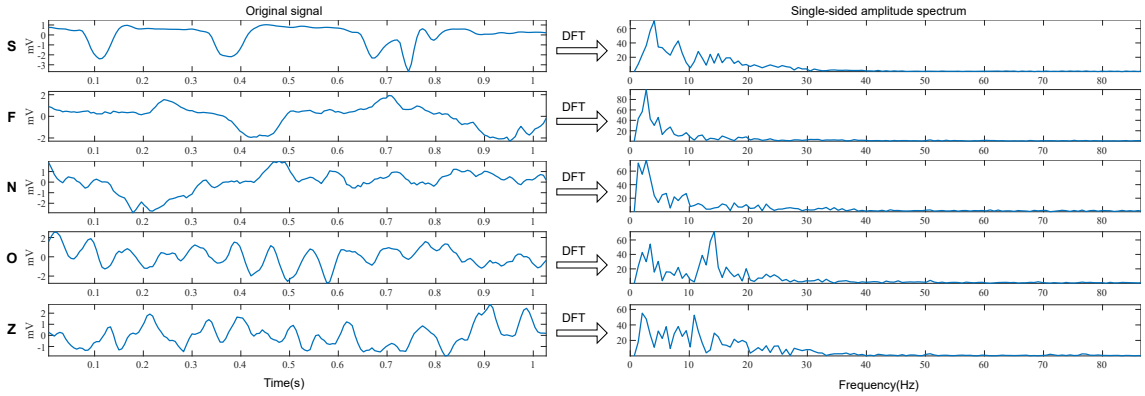


Figure 4.11: Five example signals in Ralph-Andrzejak EEG dataset (left) and their single-sided amplitude spectrum (right).

signals from each category and their corresponding single-sided amplitude spectrum.

To obtain the input to our neural network, we apply DFT on the original signal and only keep the first half. The right row in Figure 4.11 shows the corresponding single-sided amplitude spectrum. We also follow the training steps of the 5-fold

cross-validation mentioned in [85]. We randomly shuffle the data and divide them into five groups equally. We use one out of five as the test set, and in the other four groups, we choose one as the validation set and the other three as the training set(See Figure 4.12). We then train our neural network with these three training sets and use the validation set to choose the number of epochs yielding the highest accuracy. After that, we retrain the neural network on the training sets and the validation set together and use the number of epochs chosen in the previous step. We then apply this final model to the test set to get the classification results. For one round of the above steps, we can get five accuracies and we keep the average of them. Then we repeat this process ten times to get the mean and standard deviation of the accuracies for each classification task. The last row in table 4.6 shows the results obtained from our method.

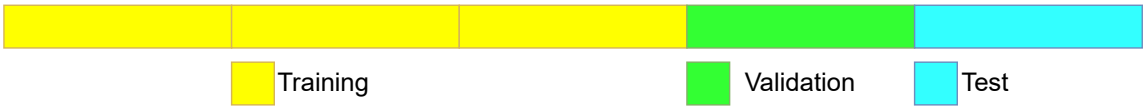


Figure 4.12: Evaluation grouping detail. In the 5-fold cross validation, each group is at least once used as validation set.

4.3.2.2 Results and comparison

We can see from Table 4.6 that our method can reach the same accuracy (99.8%±0.13%) on Z vs S and higher accuracy on Z vs N vs S (97.05±0.37%) and Z vs F vs S (96.36±0.63%) compare with the previous results. Another main concern about our method is the number of parameters and memory use. We summarize our neural network framework's parameters and memory use and the one used in [85] in Table 4.7. Our model uses 52% fewer parameters and about 15% more memory than

the model used in [85]. The additional memory cost happens in the layer of taking the modulus since this layer needs to store the modulus for the use of the next layer.

Methods	Z vs S	S vs NS	Z vs N vs S	Z vs F vs S
Neural Fuzzy Network[119]	–	–	–	86.0±0.82%
Ensemble Classifier[118]	–	–	90.0±0.71%	–
Local Binary Pattern[117]	–	98.3±0.24%	–	–
Multi-Domain Feature Extraction[116]	–	99.0±0.15%	–	–
Exponential energy + SVM[84]	99.5±0.20%	–	91.7±0.65%	89.0±0.74%
Raw data + DNN [85]	99.0±0.29%	98.8±0.20%	89.2±0.73%	89.4±0.73%
only FFT + DNN [85]	99.8±0.13%	99.4±0.14%	96.3±0.45%	95.6±0.48%
All transform + DNN[85]	99.8±0.13%	**99.5±0.13%**	96.5±0.44%	95.7±0.48%
Our method	**99.8±0.13%**	98.8±0.23%	**97.05±0.37%**	**96.36±0.63%**

Table 4.6: Accuracies comparison with previous works

4.3.2.3 Steady-State Visual Evoked Potential (SSVEP) dataset

This dataset contains multiple 8-channel EEGs collected on the occipital area from 10 healthy subjects under visual stimulations [74]. The EEGs were recored using BioSemi ActiveTwo EEG system (Biosemi, Inc.) with sampling frequency of 2048 Hz. The visual stimuli were presented on an LCD monitor 60 cm in front of the participant, and arranged as four rows and three columns of 6 cm $\times$ 6 cm squares (12 squares/stimuli in total), analogous to a phone keypad. These 12 squares (stimuli) display different flicker frequencies ranging from 9.25 Hz to 14.25 Hz in the interval of 0.5 Hz.

Model in [85]	# Para	Memory	Our method	# Para	Memory
Input		1×128=0.128k	Input		128×2=0.256k
			$\mathbb{C}$ Conv	48	128×2×8=2.048k
			ABS		128×8=1.024k
$\mathbb{R}$ Conv	192	128×32 = 4.096k	$\mathbb{R}$ Conv	656	128×16=2.048k
Pool		42×32 = 1.344k	Pool		42×16=0.672k
$\mathbb{R}$ Conv	5152	42×32 = 1.344k	$\mathbb{R}$ Conv	2592	42×32=1.344k
Fc1	241792	128=0.128k	Fc1	98688	256 = 0.256k
Fc2	4128	32=0.032k	Fc2	16640	32 = 0.032k
Fc3	128	3=0.003k	Fc3	256	3 = 0.003k
# Total	251392	**7.075k**		**118880**	8.163k

Table 4.7: Parameter and memory comparison between our method and method in [85]. $\mathbb{C}$ Conv: complex-valued convolutional layer, $\mathbb{R}$ Conv: real-valued convolutional layer, ABS: the layer of taking the modulus (absolute value), Pool: max pooling layer, Fc: fully connected layer.

During the EEG acquisition, a stimulus program turns one square into red for 1 second as indication and after that the participant was asked to gaze the indicated square for 4 seconds. The stimulus program will indicate every square only once in random order. A complete process as above is called a block; for each participant, the experiment includes 15 blocks.

We use all eight channels, and apply a 4th order Butterworth band-pass filter with a lower cutoff frequency of 6 Hz and a higher cutoff frequency of 80 Hz on the

data, which are consistent with the settings in [74, 57, 122]. Then we divided each 4 s EEG epoch into 1 s non-overlapping segments following the instructions in [122]. Next, we apply FFT on each segment with frequency resolution fixed as 0.2930 and extract the frequency components between 1 Hz and 50 Hz instead of the suggested between 3 Hz and 35 Hz [74].

Our input is a $N_{ch} \times N_{fc}$ complex-valued matrix, where $N_{ch} = 8$ is the number of channels, $N_{fc} = 169$ is the number of frequency components. We adopt the similar CNN architecture mentioned in [57]. The difference is we change the first convolution layer into complex-valued and let the kernel size equals to $N_{ch} \times 2$ instead of $N_{ch} \times 1$, and then take the absolute value as activation, then followed by a real-valued convolution layer and a fully connected layer. Our network architecture is shown in Figure 4.13. We use Adam algorithm[96] and complex Adam algorithm[93] with default settings as optimization algorithms. We use a mini-batch size equals to 32 and same dropout rate in both dropout layers. The learning rate is multiplied 0.1 every 75 epochs. The method to obtain the number of training epochs is similar to the method mentioned in 4.3.2.1; instead of using 60% and 20% of the data as training and validation sets, we use 80% and 10% of the data as training and validation sets; the optimal dropout rate is also determined in this step. The accuracies of our method reported in Table 4.8 is based on the averages of ten-fold cross-validation.

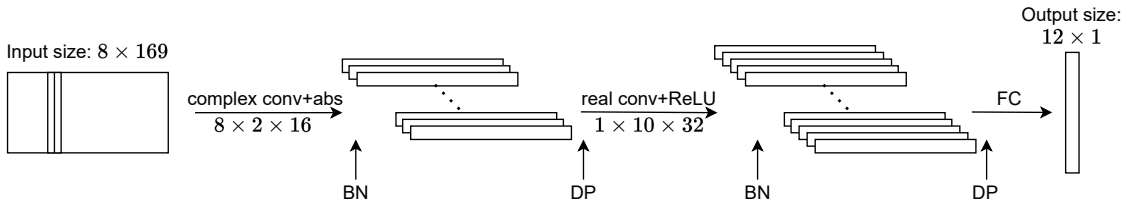


Figure 4.13: Our neural network structure for SSVEP dataset. FC: fully connected layer, BN: batch normalization layer, DP: dropour layer

Subject	CCA [57]	combined-CCA [74]	CCNN [57]	Our method
S1	29.17	78.89	77.92	86.81
S2	26.25	71.67	54.58	70.56
S3	59.44	94.44	94.58	96.94
S4	80.28	99.44	98.47	97.08
S5	52.36	100.00	99.86	99.58
S6	87.22	99.44	99.72	99.81
S7	69.17	98.33	95.00	97.36
S8	96.67	100.00	99.03	99.03
S9	66.39	98.89	97.78	97.92
S10	65.28	86.67	88.89	91.53
Mean(±std)	63.22(±21.67)	92.78(±10.22)	90.58(±14.34)	93.66(±9.07)

Table 4.8: Classification accuracies comparison for subject S1 to S10. The accuracies obtained using combined-CCA are based on leave-one-out cross-validation. The accuracies using CCA, CCNN and our method are based 10-fold cross-validation.

Form Table 4.8, we can see that our method has the highest average and the lowest standard deviation of the accuracies across all ten subjects. We also compare our accuracies with the results shown in the Figure 6 in paper[91]. We can see that our average accuracy is almost the same and our accuracy for subject 2 is higher than the method used in [91].

4.4 Discussion

In this paper, we proposed a novel neural network architecture that can capture the phase information in signals by using a complex-valued convolutional layer at the very beginning. In simulations, our framework significantly improves the classification performance compared with other methods; furthermore, our method can reduce the number of parameters and improve the accuracy simultaneously in the experiments for the real-world dataset. Besides, our framework can be used to build a more efficient hybrid complex-valued neural network structure. It can also be applied to find proper complex-valued filters on the frequency domain without prior knowledge, which is usually tricky. Currently, all the input signals to our neural network are relatively short. In the future, we plan to improve our method such that it can be applied to classify long-term EEG signals.

5 VCG Classification Based on Geometrical and Kinematical Properties

5.1 Introduction

Myocardial infarction (MI), also known as a heart attack, happens when blood flow to the heart muscle is reduced or blocked. According to an estimate from World Health Organization (WHO) in 2019, 17.9 million people died from cardiovascular diseases and 85% of these deaths were due to heart attack and stroke [123]. Therefore, it is crucial to detect MI accurately and timely, so that medical intervention can begin as early as possible.

In clinical diagnosis, an electrocardiogram (ECG), as a noninvasive and quick test, is commonly used at the initial stage to detect and analyze cardiovascular disease. Among different types of ECGs, the standard 12-lead ECG is the most widely adopted by cardiologists. The 12 leads are I, II, III, aVR, avL, aVF, V1, V2, V3, V4, V5 and V6. Among these 12 leads, Lead III, aVR, aVL, and aVF are generated from Lead I and II using a linear transformation, so the 12 lead ECG contains redundant information [124]. Besides, the standard 12 lead ECG system does not contain the posterior leads, which may cause difficulties in cardiac monitoring and diagnosis [125]. The Frank 3-lead VCG can solve the problems mentioned above. Due to historical reasons, Frank 3-lead VCG is not used as commonly as the 12-lead ECG

in making clinical decisions [126]. Fortunately, by using specific transformation, the 12-lead ECG can be transformed into 3-lead VCG without losing essential knowledge about the heart dynamics [127, 128], and vice versa[129, 130]. For the above reasons, using VCG to detect and localize MI can be more reliable and efficient.

Several methods have already been exploited to detect MI using features in VCG and ECG. Yang *et al.* [126] and Hafshejani *et al.* [131] combine VCG morphological features and ECG features to classify MI and health control using decision tree and neural networks. Methods based on the VCG features evaluated using wavelets transform show promising results in detecting MI. In [124], Tripathy *et al.* proposed using the complex wavelet sub-band L_1 norm and entropy as features and relevance vector machine as a classifier to detect MI. Khan *et al.* [125] proposed using the features captured by Fourier-Bessel series expansion based empirical wavelet transform with singular value decomposition and using SVM as a classifier. Prabhakararao *et al.* [132] suggested using multiscale eigenfeatures from the stationary wavelet transform subband matrices to detect posterior MI. In [133], Acharya *et al.* extracted 12 nonlinear features from the discrete wavelet transform of 12-lead ECG and adopted k-nearest neighbour to classify normal and MI ECG beats and localize inferior posterior infarction.

During the past decade, advances in deep learning have shown strong potential for the detection and localization of MI . In the work of Tripathy *et al.* [134], the authors introduce fixed-order Fourier-Bessel series expansion based empirical wavelet transform and use convolutional neural networks to localize MI. Karisik *et al.* [135] proposed using long-short-term memory (LSTM) network and template adaptation techniques to detect MI from VCG. Chuang *et al.* [136] exploited the VCG derived from Lead-I ECG and using LSTM with spline representation to classify MI automatically. However, models based on neural networks are not interpretable, although

they achieved state-of-art results in many areas [137]. Besides, for cardiologists, the features extracted from the frequency domain are not valid for interpretation [131].

Therefore, we want to propose several new interpretable and observable features based on VCG. Due to the VCG can be depicted as a movement track in a three-dimensional Cartesian coordinate with time as the parameter, we can extract features based on the VCG's kinematical and geometrical properties, such as velocity, acceleration, jolt, curvature and torsion. These properties are intuitive in describing trajectory. We extract local and global features based on these properties, then combine these newly proposed features and some features mentioned in previous work [126, 138] to detect MI from healthy control; furthermore, we use these features to classify anterior and inferior MI. The total number of features before feature selection is 95 and we give each feature an abbreviation (See Table 5.3, 5.4, 5.5). We test our method on two public benchmark datasets: PTB [139] and PTB-XL [140]. In the PTB dataset, compared with previous results obtained using interpretable features, our approach can increase the classification accuracies from 94.43% to 96.04% (sensitivity from 97.08% to 99.01%), and from 95.23% to 97.28% (sensitivity from 95.79% to 98.95%) concerning different data cleaning methods. In the PTB-XL dataset, we demonstrate our approach in distinguishing between MI and health control (HC), and distinguishing between anterior MI (AMI) and inferior MI (IMI). Due to PTB-XL dataset only contains ECG, we also compare our method under different ECG to VCG transformations, such as Dower transform[127], Kors Quasi-Orthogonal transform [141, 128], Kors transform [141, 128], QLSV transform [142], and PLSV transform [142]. Our method can achieve accuracies of 94.02% with Kors transform in distinguishing MI from HC, and 82.14% with PLSV transform in classifying AMI and IMI.

The rest of this chapter is structured as follows: section 2 briefly introduces the

benchmark dataset we use and the details about the implementation of our methods. Section 3 shows our experiments and results. Section 4 summaries this chapter and discusses future work.

5.2 Materials and Methods

5.2.1 Datasets

We use two publicly available datasets: Physikalisch-Technische Bundesanstalt (PTB) [139] and PTB-XL [140], to evaluate our method on the classification and localization of myocardial infarction. Both datasets were downloaded from Physionet [143].

The PTB dataset contains 549 ECG records digitized at 1000 Hz with 16-bit resolution collected from 290 subjects. This dataset has 80 records from 52 healthy volunteers and 368 records from 148 MI patients (Table 5.1a). Each record includes the conventional 12-lead ECG along with a 3-lead Frank VCG.

Diagnostic class	# of subjects	# of records
Myocardial infarction	148	368
Healthy controls	52	80
Total	200	448

(a) Number of the subjects and records in PTB dataset

Superclass	Description	# of records
NORM	Normal ECG	9528
MI	Myocardial Infarction	5486
STTC	ST/T Change	5250
CD	Conduction Disturbance	4907
HYP	Hypertrophy	2655
	Total	27826

(b) Five superclasses in PTB-XL dataset

Table 5.1: Number of the subject and/or records in the datasets used in our experiment.

The PTB-XL dataset contains 21837 clinical ECG records from 18885 patients. Each record lasts for 10 seconds and only includes a 12-lead ECG. This dataset contains five superclasses (Table 5.1b) and provides two sampling frequency choices: 100 Hz and 500 Hz. In our experiment, we only use the ECG records sampled at 500 Hz in the superclasses of NORM and MI. Besides, each ECG record in this dataset corresponds to one SCP-ECG statement, along with the likelihood information for the diagnosis, which means one record can have more than one cardiac condition.

5.2.2 pre-processing

Raw VCG signals are usually contaminated by noise, DC offset, baseline drift, and power line interference. For PTB dataset, we first remove the baseline drift using a wavelet transform filter, then apply a high pass (0.05 Hz) and low pass (150 Hz) filter to remove the DC offset and artifacts. Next, we apply a Notch filter to remove power line interference (50/60 Hz). Finally, we subtract the most frequent amplitude to correct the isoline line. Next, we use ECGDeli [144] to delineate the VCG signals. ECGDeli contains a voting algorithm that aims to use multi-channels to refine the delineation process. In our case, we only use the three channels in VCG (Vx, Vy, and Vz) to delineate the signals (See Diagram 5.1a).

We apply the same pre-processing procedures mentioned above for the ECG records in the PTB-XL dataset. Next, because the PTB-XL dataset does not contain Frank 3-lead VCG, we use linear transformation and LSTM network to obtain the synthesized VCG. We then delineate the signals using ECGDeli [144] and extract the features (see Figure 5.1b). More details about the transformation can be found in section 5.3.2.1. Note that when using linear transformation to obtain VCG, we apply isoline correction again to remove the shift of signals.

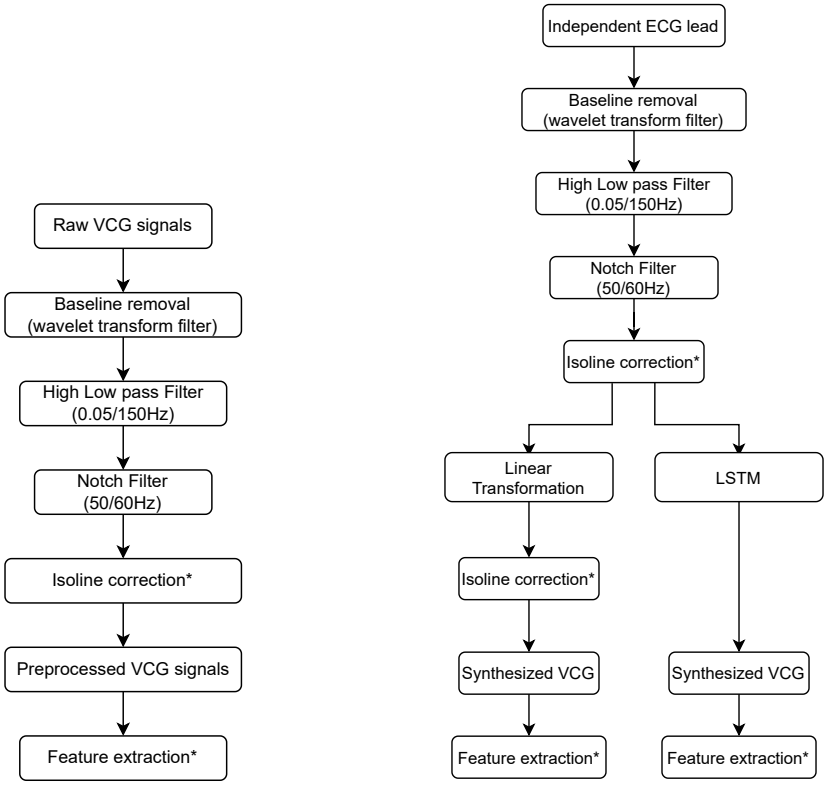


Figure 5.1: (a) Diagram of the pre-processing steps for the PTB dataset. (b) Diagram of the pre-processing steps for the PTB-XL dataset. We use the build-in function in ECGDeli[144] in the steps ending with *.

5.2.3 VCG vector and octant features

Due to the three axes Vx, Vy, and Vz in VCG signals being perpendicular to each other, the VCG signals can be drawn in a three-dimensional Cartesian coordinate containing eight octants. In the 1980s, Laufberger published a series of papers [145, 146, 147, 148] indicating that the octant distribution of VCG signals has the potential to detect myocardial infarction. In the recent decade, octant distributions have been used as essential features in the analysis of MI [126, 131]. Figure 5.2 shows an

example of HC and MI VCG signals in the Cartesian coordinate where different color means different octant. Table 5.2 shows the signs of the coordinates of the points on VCG corresponding to different octants.

We can see from Figure 5.2 that the ratio of VCG components of HC and MI in different octants are different, therefore, we use the eight ratios as features. Moreover, we can view the amplitudes of VCG in different octants following a specific distribution, so we take the maximum, average and variance as features to describe the distribution. Another two important features we use are the octant location and magnitude of the T-vector which begins at the isoelectric point and ends at the farthermost point in the T loops. The T-vector location is a discrete value and can be any integer between 0 and 7; such variable is usually treated as categorical value. However, based on the discover in [126], the T-vector location exhibits properties of ordinals, so in our experiments, we treat T-vector as a numeric value. We totally extract 34 features based on the VCG vectors and their octant distributions (See Table 5.3).

5.2.4 Morphological and shape features

Morphological and shape features are analyzed and applied to localize heart disease [149, 138]. We use the same methods mentioned in [138] to calculate the perimeter and planar area of the projected QRS-loop (Figure 5.3), and the volume of the QRS-loop (Figure 5.5). Besides, we use the path length of the original QRS-loop and the ratio of the path length to the projected perimeter as two new features. We define the projected height of the QRS-loop as the ratio of the volume to the planar area. We also define the span of the QRS-loop as the maximum distance of the points on the QRS-loop projected to the normal vector of the optimal plane. We also use

octant	V_x	V_y	V_z
0	−	−	−
1	−	−	+
2	−	+	−
3	−	+	+
4	+	−	−
5	+	−	+
6	+	+	−
7	+	+	+

Table 5.2: The sign of V_x, V_y, V_z in the corresponding octant.

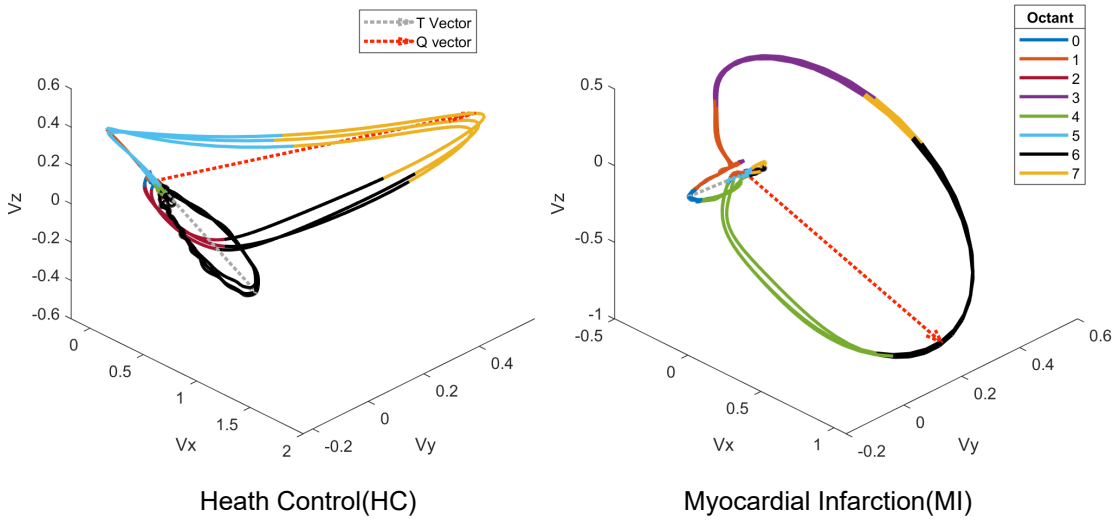


Figure 5.2: Two example VCG signals in health control (HC) and myocardial infarction (MI) groups. Different colors on the VCG are corresponding to different octants. The grey and scarlet dotted line are the T and Q vector with the largest magnitude.

	Feature description	Abbrev.	# of features
Octant features (8 octants)	Ratio (percentage of the components)	R1 to R8	8
	Max magnitude	MaxMag1 to MaxMag8	8
	Average magnitude	AveMag1 to AveMag8	8
	Variance of the magnitudes	VarMag1 to VarMag8	8
	Octant location of T-vector	OctT	1
Vector feature	Magnitude of T-vector	MagT	1
	Totol		34

Table 5.3: Details of octant features

the angle between T-vector and Q-vector as an essential feature. The definition of Q-vector is similar to the T-vector; Q-vector begins at the isoelectric point and ends at the farthermost point in the Q loops.

For the shape features, we use the areas under QRS wave, J wave and T wave. Because each VCG contains three channels V_x,V_y and V_z, in each channel, there are three areas, we totally have 9 shape features. (See Table 5.4)

The total number of morphological and shape features is 19.

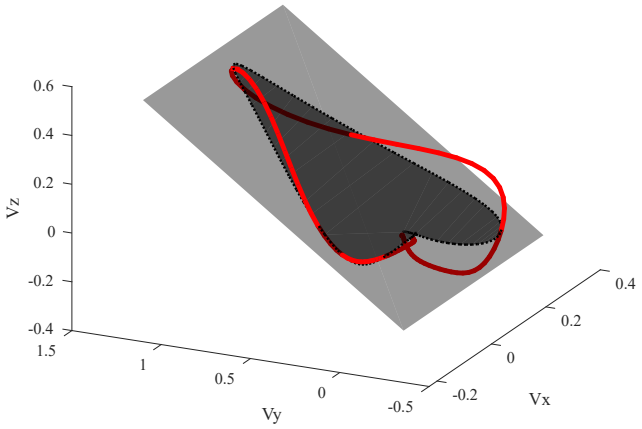


Figure 5.3: The gray plane is the optimum plane calculated using the least square method. The solid red line is a QRS complex selected from a recording in the health control group. The dotted line is the projection of the QRS complex on the optimum plane. The darker area in the optimum plane is the QRS complex's projection area.

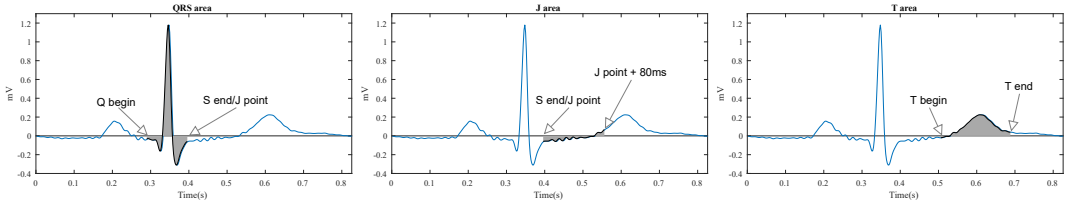


Figure 5.4: The grey color areas in this figure shows the areas under QRS wave, J wave and T wave. The waveform in this figure is a part of (3.27s to 4.09s) the V_y lead generated from "00070_hrm.dat" file using Kors tranformation matrix.

5.2.5 Geometric and kinematics features

Due to the VCG changes with time, in a 3-dimensional Cartesian coordinate system, we can regard the trajectory of VCG as a parametrically-defined space curve with time as the parameter. We can write this space curve and its vector form as

	Feature description	Abbrev.	# of features
Morphological features	projected QRS-loop planar area	PPA	1
	projected QRS-loop perimeter	PPL	1
	QRS-loop path length/duration of QRS-loop	PL	1
	The ratio of PPL to PP	PPL/PL	1
	QRS-loop volume	VOL	1
	Height (the ratio of VOL to PPA)	Height	1
	Span (mean, min, max)	MeanS, MinS, MaxS	3
	The angle between T-vector and Q-vector	AngTQ	1
Shape features	Area under QRS segment (V_x,V_y and V_z)	AQRSx,AQRSy,AQRSz	3
	Area under T segment (V_x,V_y and V_z)	ATx,ATy,ATz	3
	Area under J segment (V_x,V_y and V_z)	AJx,AJy,AJz	3
	Total		19

Table 5.4: Details of morphological and shape features

the following:

$$\boldsymbol{r}(t) = (x, y, z) \tag{2.1}$$

where t is time, $x = x(t)$, $y = y(t)$ and $z = z(t)$ are independent and corresponding to the 3 leads of VCG: V_x, V_y and V_z, respectively. In addition to Equation (2.1), we can use the following curvature (κ) and torsion (τ) uniquely determine a space

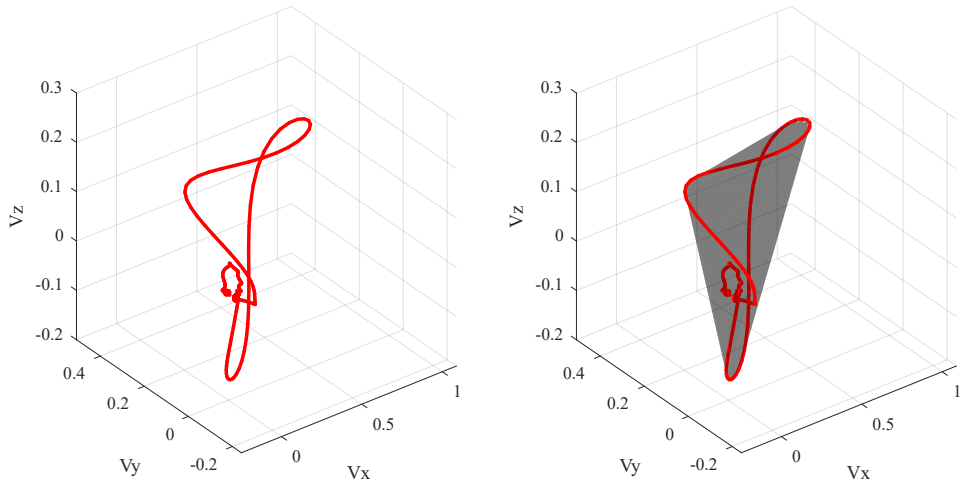


Figure 5.5: This left figure shows an example VCG selected from HC groups and the shaded area in right figure shows the minimum convex hull that contains this piece of VCG.

curve [150]:

$$\kappa = \frac{||\boldsymbol{r}'(t) \times \boldsymbol{r}''(t)||}{||\boldsymbol{r}'(t)||^3} = \frac{\sqrt{(z''y' - y''z')^2 + (x''z' - z''x')^2 + (y''x' - x''y')^2}}{(x'^2 + y'^2 + z'^2)^{\frac{3}{2}}} \tag{2.2}$$

$$\tau = \frac{(\boldsymbol{r}'(t) \times \boldsymbol{r}''(t)) \cdot \boldsymbol{r}'''(t)}{||\boldsymbol{r}'(t) \times \boldsymbol{r}''(t)||^2} = \frac{x'''(y'z'' - y''z') + y'''(x''z' - x'z'') + z'''(x'y'' - x''y')}{(y'z'' - y''z')^2 + (x''z' - x'z'')^2 + (x'y'' - x''y')^2} \tag{2.3}$$

where $'$, $''$, and $'''$ represent the first, second, and third order derivatives. Curvature depicts the deviation of the curve from being a straight line, and torsion measures the intensity of a curve bending out of the osculating plane.

From Equation (2.2) and (2.3), we can find that the curvature and torsion can be determined by the first, second, and third-order derivatives of $\boldsymbol{r}(t)$ with respect to time t. Furthermore, first, second, and third-order derivatives have their kinematics meanings, which are velocity, acceleration, and jolt, respectively. Equa-

tions (2.4), (2.5) and (2.6) show the formulas:

$$|\mathrm{v}| = |\boldsymbol{r}'(t)| = \sqrt{x'^2 + y'^2 + z'^2} \tag{2.4}$$

$$|\mathrm{a}| = |\boldsymbol{r}''(t)| = \sqrt{x''^2 + y''^2 + z''^2} \tag{2.5}$$

$$|\mathrm{j}| = |\boldsymbol{r}'''(t)| = \sqrt{x'''^2 + y'''^2 + z'''^2} \tag{2.6}$$

where |v|, |a| and |j| are the magnitudes of the velocity, acceleration and jolt of VCG, respectively.

The newly proposed features are based on the curvature, torsion, velocity, acceleration, and jolt in the QRS complex. From equations (2) to (6), we can see that to calculate κ, τ, |v|, |a|, and |j|, we only need to evaluate nine derivatives: x', y', z', x'', y'', z'', x''', y''' and z'''. Due to the VCG records being discrete data, we need to use a numerical method to approximate these derivatives.

5.2.5.1 Approximation method

The regular numerical approximation of the derivatives based on the difference method is unstable, even if the signals have high signal-to-noise ratio (SNR) [151]. Here we give an example to show the instability of using the central difference method. We first simulate a piece of ECG signal using the code in [152] and use the Savitzky-Golay filter to smooth the simulated signal; then we add white noise (SNR = 60 dB) (See figure 5.6). We apply the central difference scheme on simulated ECG signals to approximate the first, second and third-order derivatives (see Figure 5.7a and 5.7b for the results). We can see from Figure 5.7b, even such noise can result in the numerical differentiation method fails in the approximation of the high order derivatives.

Therefore, we use the continuous wavelets transform(CWT) method to approxi-

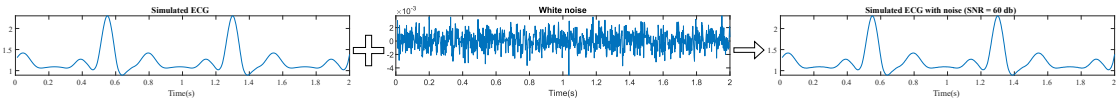
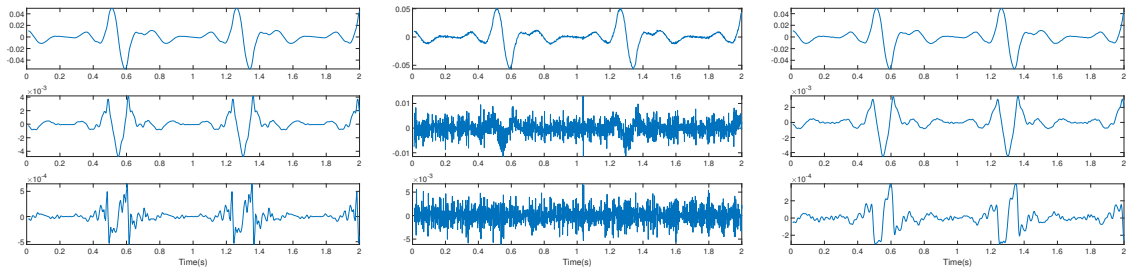


Figure 5.6: The process of producing an ECG with noise.. The left graph is the simulated ECG waveform; the middle graph is the white noise; the right graph is the simulated ECG with noise whose SNR is 60 dB and it is hard to find the difference between the noised ECG and the original ECG by visual inspection.



(a) Without noise (central difference) **(b)** With noise (central difference) **(c)** With noise (CWT)

Figure 5.7: Derivatives approximated using the central difference method and CWT method under the different cases (with or without noise). From Figure 5.7a, we can see that the CWT method is stable, and the approximated derivatives are closer to the ones obtained using the simulated ECG without noise. The rows from top to bottom are first, second, and third-order derivatives.

mate the derivatives [153, 154]. The CWT method can be understood as the derivative of a signal smoothed by a specific kernel (smoothing function) [155]. Here, we give the formula and smoothing function used in this chapter. Suppose we have a signal y(x); the CWT of y(x) can be defined as:

$$y(\bar{x})^w = \int_{-\infty}^{+\infty} y(x)\frac{1}{\sqrt{s}}\psi(\frac{x-\bar{x}}{s})dx,$$

where s is the dilation parameter, $\bar{x}$ is the translation parameter and $\psi(x)$ is the mother wavelet. In this context, we let

$$\psi(x) = -\frac{d\theta(x)}{dx}, \qquad \theta(x) = -\mathrm{e}^{-x^2}\sqrt[4]{2/\pi},$$

here we choose a modified gaussian function $\theta(x)$ as the smoothing function. $\theta(x)$ has nonzero constant integral as:

$$K = \int_{-\infty}^{+\infty} \theta(x)dx = -\sqrt[4]{2/\pi}$$

Then, the derivative of y(x) can be approximated by:

$$\frac{dy(x)}{dx} \approx \frac{y(\bar{x})^w}{Ks^{\frac{3}{2}}}$$

We apply the CWT method on the first-order derivatives to approximate the second-order derivatives. Similarly, we can approximate the third-order derivatives from second-order derivatives.

Note that the dilation s (the scale of wavelet kernel) is important in obtaining the proper derivative. We can find a detailed discussion about how to choose proper dilation in [151]. Here, based on our experiment, we find that: a small dilation can result in high noise in the approximated derivative; however, a large dilation can dilute the local features (See Figure 5.8). We can see from equations (2.2) and (2.3), the calculation of curvature and torsion use division leads to a noise-sensitive operation. Figure 5.9 shows the effect of dilation on the approximation of

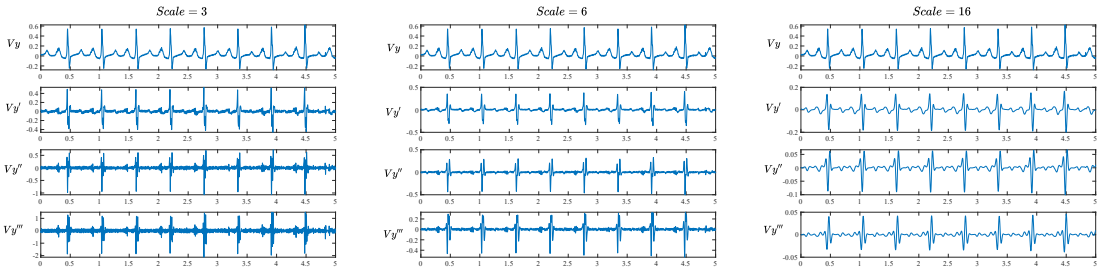


Figure 5.8: The approximated first, second and third-derivative of the 3-lead VCG signals, respectively (dilation = 3,6,16). When the dilation is 3, the third-order derivatives contain more noise. When the dilation is 16, the third order derivatives are overly smoothed, which results in losing local features.

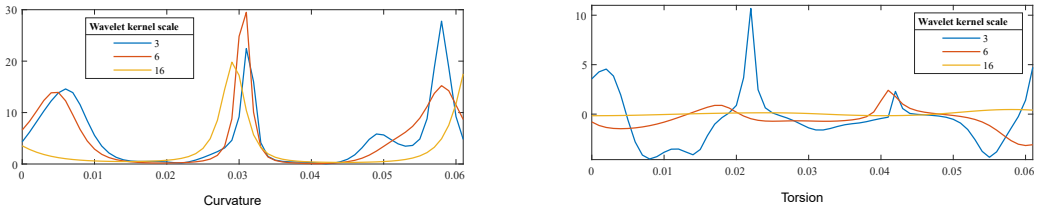


Figure 5.9: Curvature and torsion of an example QRS complex under different dilation.

curvatures and torsions. Our experiments found that the dilation should be chosen as an integer number around 8.

5.2.5.2 Features extraction

In our method, we only consider the geometric and kinematics features in each QRS-complex because the noise severely affects the feature extraction on P and T loop. Because the features we want to extract contain local features that can not be directly used in the classification algorithm, so we have to first summarize them. We can categorize the features we extracted in this section into two categories: "per record" and "on every QRS complex"; "per record" means the values are collected

from all QRS complex, "on every QRS complex" means the values are collected on each QRS complex. Take the curvature features as an example: "per record" features include sum, standard deviation, maximum, average and variance, which are calculated from the values from all QRS complex; "on every QRS complex" features include average of the maximum, average of the average and standard deviation of the average, which are first calculated the maximum and the average on each QRS complex and then based on these value, we further calculate average and standard deviation. Torsion features are similar to curvature features, except of two features need to evaluate the absolute value. Due to torsion can be negative or non-negative, we take the absolute value of it and calculate the sum and standard deviation of them per record. We also extract quantiles of the curvature and torsion per record for the cumulative probability are 0.1 to 0.9 with interval equals to 0.1 (See Table 5.5).

5.2.6 Variable selection

The total number of extracted features is 95. We use RliefF [156] to reduce the dimension of the variables. RliefF is a feature selection algorithm based on the filter-method approach and is sensitive to feature interactions. RliefF algorithm first uses the k-nearest neighborhood (KNN) to find the k-nearest observations of a randomly selected observation, then penalizes the predictors that give different values to neighbors of the same class and rewards the predictors that give different values to neighbors of different classes. There are three main parameters in the RliefF algorithm — m (the number of iterations), k (the number of nearest neighbors), and τ (the relevance threshold). Because our dataset is relatively small, we set m as the number of observations [157]. The parameter k is used to calculate the nearest k hits and k misses, and it is a user-defined parameter that controls the locality of the

	Feature description	Abbrev.	# of features
Curvature features	Sum of the curvatures per record	SumCur	1
	Standard deviation of the curvatures per record	SdCur	1
	Average of the maximum curvatures on every QRS complex	AveMaxCur	1
	Average of the average curvatures on every QRS complex	AveMeanCur	1
	Standard deviation of the averages curvatures on every QRS complex	SdMeanCur	1
	Maximum curvature per record	MaxCur	1
	Average curvature per record	AveCur	1
	Variance of all curvatures per record	VarCur	1
	10th to 90th quantile of curvatures per record	CurQ1 to CurQ9	9
Torsion features	Sum of the torsions per record	SumTor	1
	Standard deviation of the torsions per record	SdTor	1
	Sum of the absolute value of torsion per record	SumAbsTor	1
	Standard deviation of the absolute value of torsion per record	SdAbsTor	1
	Average of the maximum torsions on every QRS complex	AveMaxTor	1
	Average of the average torsions on every QRS complex	AveQRSTor	1
	Standard deviation of the average torsions on every QRS complex	SdQRSTor	1
	Maximum torsion per record	MaxTor	1
	Average torsion per record	AveTor	1
	Variance of all torsions per record	VarTor	1
	10th to 90th quantile of Torsions per record	TorQ1 to TorQ9	9
Kinematics features	The maximum of first derivative per record	MaxSpeed	1
	The average of the maximum first derivatives on every QRS complex	AveMaxSpeed	1
	The maximum of second derivative per record	MaxAcc	1
	The average of the maximum second derivatives on every QRS complex	AveMaxAcc	1
	The maximum of third derivative per record	MaxJolt	1
	The average of the maximum third derivatives on every QRS complex	AveMaxJolt	1
	Total		42

Table 5.5: Details of curvature, torsion and kinematics features

estimates [156]. In our experiment, we set k as the integer equals to the round down of 15% of the total number of observations. After we set m and k, we can perform

RliefF and obtain a weight for each predictor. The weights can be either positive or non-positive. The features with positive weights are considered relevant features, and the larger the feature's weight, the greater discriminating power it contains. However, not all the features with positive weight should be accepted [158], so we need to define the relevance threshold τ such that any features with a weight larger than this threshold should be selected. In the original paper on the theoretical analysis of the Relief method [159], a suggested range of τ is given by:

$$0 < \tau \leq \frac{1}{\sqrt{m\alpha}}$$

where α is the probability of accepting an irrelevant feature as relevant. In practice, it is also suggested to choose some number of features instead of choosing a τ [158]. In our experiment, we keep all the features with a weight larger than 15% of the largest weight.

5.3 Experiments and Results

5.3.1 Experiments on the PTB dataset

We perform two classification tasks based on the PTB dataset. In the first task, we keep all 448 ECG records (MI: 368 and HC: 80) (See Table 5.1a) and perform the classification between HC and MI. To compare our results against the results in the literature [160], we adopt 10-fold cross-validation. In this experiment, we use a support vector machine (SVM) with the radial basis function (RBF) kernel as a classifier. In the second experiment, we follow the data filtering method mentioned in [138]; we only keep the ECG records collected within one week after the MI happened and then further separate the MI records into anterior-MI and inferior-MI. In this experiment, we perform the multi-class classification among the individuals with HC,

anterior-MI and inferior-MI. To compare with the result in [138], we adopt leave-one-out cross-validation and build two binary classifiers using SVM (with RBF kernel): the first classifier is used to classify between HC and MI, and the second classifier is used to classify between anterior-MI and inferior-MI. For individuals with more than one record, we first extract features from each heartbeat (R-R interval) and then combine them.

localization	# of subjects	# of ECG records
Inferior	23	43
Infero-lateral	17	25
Infero-postero-lateral	6	10
Total	46	78
Anterior	14	21
Antero-septal	24	41
Antero-lateral	10	17
Antero-septo-lateral	1	2
Total	49	81

Table 5.6: Details of PTB dataset

5.3.1.1 Results and discussions

The accuracies comparison with [160] is shown in Table 5.7, which is the average accuracy of 100 times repeat experiments using 10 percent as the test set and 90 percent as the training set. To compare with the results in [138], we adopted Leave-one-out cross-validation (Table 5.9a and 5.9b).

	MI	HC	Total
[160]	97.08(2.88)	82.50(15.39)	94.43(3.00)
Our method	**99.14(0.34)**	**84.95(1.53)**	**96.61(0.40)**

Table 5.7: Results comparison with [160] (HC vs MI).

	Accuracy	Sensitivity	Specificity
[138]	0.9523	0.9579	0.9423
Our method	**0.9728**	**0.9895**	**0.9423**

Table 5.8: Results comparison (per patient) with [138] (HC vs MI). The results are based on leave-one-out cross-validation.

	HC	AMI	IMI	Total
HC	49(**49**)	0(**1**)	3(**2**)	52
AMI	2(**0**)	44(**48**)	3(**1**)	49
IMI	2(**1**)	**5**(6)	39(**39**)	46

(a) Contingency table: compared with [138](per patient) HC vs AMI vs IMI. Our results are shown in the parenthesis.

	HC	AMI	IMI
Sensitivity	0.9423(**0.9423**)	0.8980(**0.9796**)	0.8478(**0.8478**)
Specificity	0.9579(**0.9895**)	0.9490(0.9286)	0.9406(**0.9703**)
Accuracy	0.8980(**0.9252**)	0.8980(**0.9252**)	0.8980(**0.9252**)

(b) The comparison (per patient) of sensitivity, specificity and accuracy with [138]. Our results are shown in the parenthesis.

Table 5.9: Results comparison (per patient) with [138] (HC vs AMI vs IMI)

Here, we only give the feature analysis based on the second experiment; because in this experiment, the classification is for individuals and only the ECGs obtained within the first week after MI was selected, which makes the extracted features be more representative. Figure 5.10 shows the 15 selected features to classify MI and HC. We can see that the weight of T vector Magnitude is significantly larger than

the weights of other selected features, which is consistent with the previous results in [126]; we can also see that our newly discovered features AveMaxSpeed, AveMaxTor, MaxTor and AveMaxJolt play important roles in the classification. Figure 5.11 shows the 35 selected features to classify Anterior MI and Inferior MI. We can see that our newly discovered feauters: MaxCur, TorQ7, AveMaxCur, SumCur, VarTor and MaxJolt are participated into the classification.

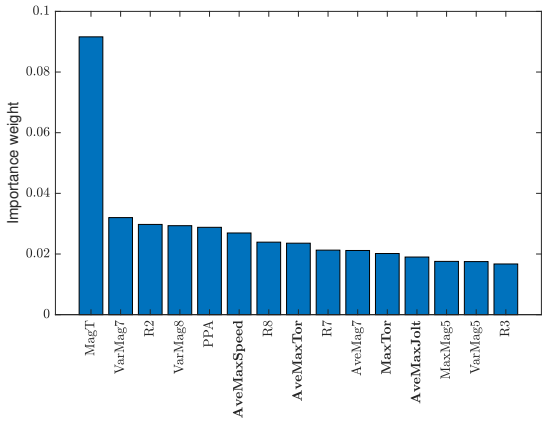


Figure 5.10: 15 selected features to classify between MI and HC. The bold abbreviations show our newly proposed features.

5.3.2 Experiments on PTB-XL dataset

In this dataset, we perform the classification task on differentiating between normal and myocardial infarction ECG. For normal ECG records, we only keep the ones with the SCP-ECG statements showing "NORM" and for MI ECG record, we only keep the records whose subclasses belongs to "AMI", "IMI", "LMI" and "PMI". For both "NORM" and "MI", we only keep the dataset with the 100.0 likelihood. Note that some records are belonged to different subclasses. As long as the record shows likelihood of 100.0 in one of these subclasses, we keep it. Next we filter out all

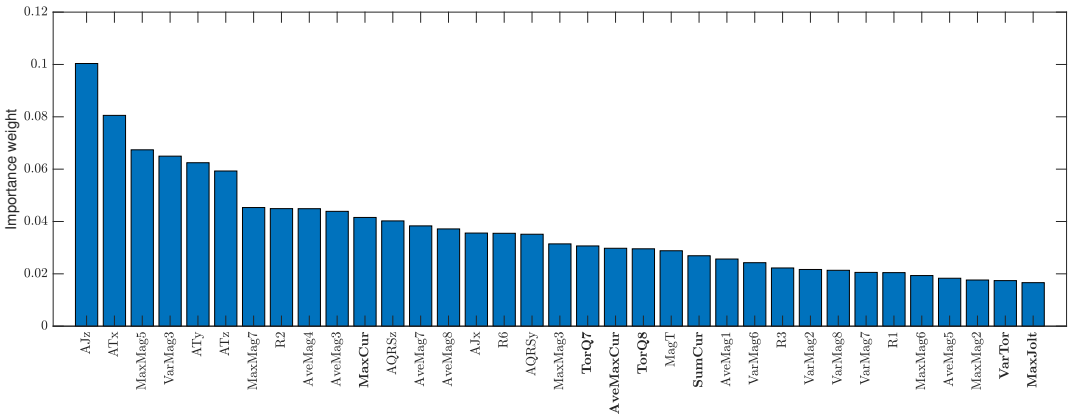


Figure 5.11: 35 selected features to further classify AMI and IMI. The bold abbreviations show our newly proposed features.

the records that are not validated by human. Finally, there are 6004 normal ECG records come from 5746 patients and 1754 MI ECG records come from 1563 patients left. See Table 5.10 for more details about the distribution of gender and age in the selected records.

	# of records	# of patients	Gender	Age
NORM	6004	5746	M:3014(52.5%) F:2732(47.5%)	59.73(±16.79)
MI	1754	1563	M:789(50.5%) F:774(49.5%)	59.04(±17.71)
All	7758	7309	M:3803(52.0%) F:3506(48.0%)	59.58(±16.99)

Table 5.10: Details of PTB XL dataset after filter

5.3.2.1 ECG to VCG Transformation

Due to this dataset does not provide VCG leads, we need to use transformation methods to obtain 3-lead VCG from 12-lead ECG. There are linear and non-linear

transformation methods. Most linear methods are based on applying the linear transformation on the independent eight leads (I, II and V1 to V6) and can be described in the following form:

$$V = W_t E \tag{3.7}$$

where V represents the transformed VCG signal whose rows are corresponding to Vx, Vy, Vz lead, W_t is the transformation matrix and E represents the ECG signal whose rows are formed by independent ECG leads. Nonlinear transformation methods include using artificial neural network to synthesize VCG signals from leads I, II and V2 [161] and using long-short-term memory to generate VCG signals from single lead ECG (I lead) [136].

In our experiments, we test the effect of both linear and nonlinear transformations on our classification task. In linear transformation, we adopt five transformation matrices [162], which are inverse Dower [127], Kors Quasi-Orthogonal [141, 128], Kors [141, 128], QLSV [142] and PLSV [142] (see Appendix for the matrices). Inverse Dower matrix is the pseudoinversion of the Dower matrix for deriving ECG from VCG [130, 127]. The idea of Kors Quasi-Orthogonal method is choosing the ECG lead which shows the highest median correlation with each VCG lead and then adjust the amplitude of selected ECG lead [141]. The matrix of Kors method was derived from the regression on the QRS complex using multivariate regression [128]. PLSV and QLSV are based on least square value (LSV) optimization and focus on the recovery of P and Q loop respectively [142].

In nonlinear transformation, we train a lstm network using PTB dataset. Our neural network consists of two LSTM layers with 50 hidden units each and one fully-connected layer. The input is the eight independent leads and the output is the 3 leads VCG. We fix the input and output sequence length to 5000 by first down-

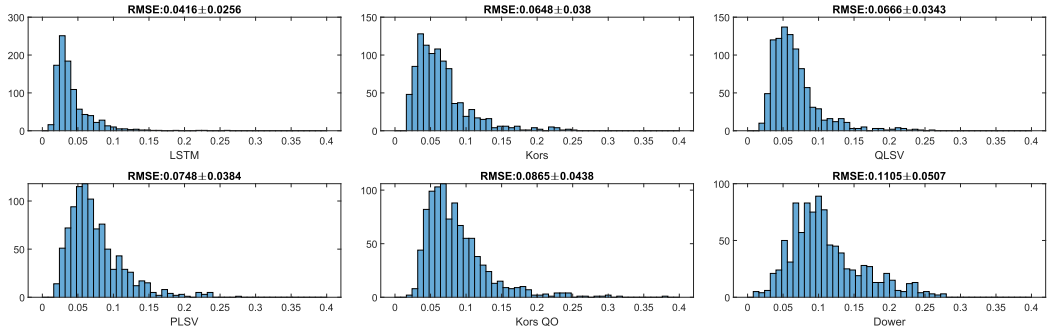


Figure 5.12: The distributions of RMSE for different transformation methods.

sampling the ECG and VCG signals from 1000 Hz to 500 Hz, and then randomly cutting the signals into the pieces of 10 seconds with overlaps. To prevent overfitting, we add two dropout layers with 0.3 dropout rate: the first one is between the two LSTM layers and the second one is between the LSTM layer and fully-connected layer. We use Adam as optimizer, and set mini-batch size as 128. We randomly keep 20% data as test set and use the other 80% data as training set to obtain the optimal number of epoches. Then we use whole PTB dataset as training set and use the obtained number of epoches to train our final LSTM network. We then use this network to obtain 3-lead VCGs from the 12-lead ECG in PTB-XL dataset.

To compare the performances of different transfromation methods, we calculate the root mean square error (RMSE) of applying all the methods on the test set mentioned above. Figure 5.12 shows the histograms of the RMSEs. We can see that the transformation using LSTM and Dower matrix have the lowest and highest RMSE respectively.

We plot three representative reconstructed VCG signals (See Figure 5.13). These three VCG signals are generated using LSTM network. We select the VCGs whose average RMSE are the lowest, the medium and the highest among test set in LSTM

network.

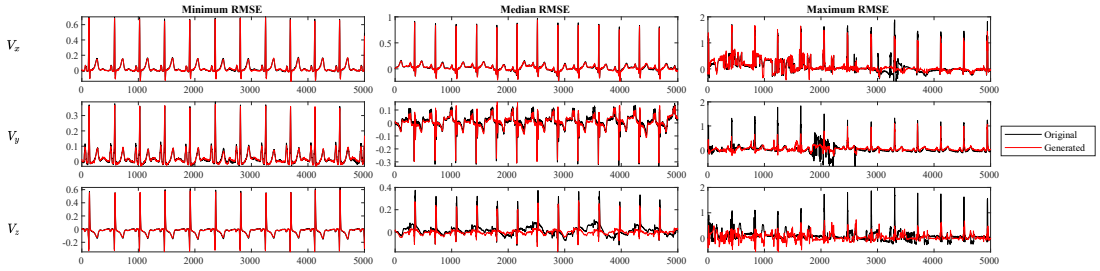


Figure 5.13: This figure shows the reconstruction from ECG to VCG using LSTM network. The left column shows an example that it has the lowest RMSE. The middle column shows the reconstruction with median RMSE and the right column shows the reconstruction with highest RMSE. These three examples come from test set.

We use error-correcting output codes (ECOC) [163] model using SVM binary learners to classify these three groups.

5.3.2.2 Results and discussions

We also use RliefF to select the variables for this experiment. Table 5.11 shows the best average accuracies for each transformation method under 100 times 10-fold cross validation. We can see that our results show lower classification accuracy compared with the accuracies obtained for PTB dataset (Table 5.7 and Table 5.9a). There are two main reasons to be concerned: firstly the proportion of the MI in the selected dataset is about 22.6%; such imbalanced dataset can skew the machine learning algorithm towards the majority. Secondly, the 12-lead ECG lacks the lead attached to the back; our newly proposed features may be sensitive to the lack of back lead, although some papers [127, 128] mention that the 3-lead VCG synthesized from 12-lead ECG will not lose significant diagnostic information. We can also find

HC vs MI

Method	Accuracy	Sensitivity	Specificity
Kors QO	0.9348±0.0014	0.9634±0.0011	0.8149±0.0052
Kors	**0.9402±0.0013**	**0.9684±0.0013**	**0.8217±0.0044**
ID	0.9386±0.0014	0.9674±0.0011	0.8178±0.0046
QLSV	0.9373±0.0012	0.9661±0.0011	0.8162±0.0048
PLSV	0.9391±0.0012	0.9647±0.0010	0.8316±0.0043
LSTM	0.9353±0.0014	0.9637±0.0014	0.8157±0.0040

that by using our features, not the transformation based on LSTM (with minimum RMSE) shows the best performance, the best performance happens when we use PLSV transformation.

5.4 Conclusions

In this study, we combine our newly proposed features, based on differential geometry to capture intrinsic properties, with features used in previous works to classify and localize myocardial infarction. Experimental results show that our method has an improvement in both accuracy and sensitivity compared to previous methods. We also demonstrate our method on the VCG signals synthesized from 12-lead ECG using different transformation methods. In our future work, we aim to develop a more efficient method to better summarize the local features, especially the local information extracted from the curvature and torsion.

AMI vs IMI

Method	Accuracy	Sensitivity	Specificity
Kors QO	0.7791±0.0060	0.5403±0.0131	0.8935±0.0066
Kors	0.7847±0.0062	0.5773±0.0129	0.8840±0.0062
ID	0.8149±0.0057	0.6339±0.0123	0.9015±0.0059
QLSV	0.7971±0.0060	0.6168±0.0133	0.8835±0.0064
PLSV	**0.8214±0.0060**	**0.6330±0.0121**	**0.9117±0.0073**
LSTM	0.8044±0.0058	0.5922±0.0138	0.9026±0.0057

Table 5.11: Classification performances of the proposed method on PTB-XL dataset. The mean and standard deviation of accuracies, sensitivities and specificities. "Kors QO": Kors Quasi-Orthogonal, "ID": inverse Dower

Bibliography

[1] George EP Box et al. *Time Series Analysis: Forecasting and Control*. John Wiley & Sons, 2015.

[2] Peter J Brockwell and Richard A Davis. *Introduction to Time Series and Forecasting*. Springer, 2002.

[3] Robert H Shumway and David S Stoffer. *Time Series Analysis and Its Applications: With R Examples*. Springer, 2006.

[4] M.B. Priestley. *Spectral Analysis and Time Series*. Academic press, 1981.

[5] Wayne A Fuller. *Introduction to Statistical Time Series*. John Wiley & Sons, 1996.

[6] Petre Stoica and Randolph Moses. *Spectral Analysis of Signals*. Prentice Hall, 2005.

[7] Leif Sörnmo and Pablo Laguna. *Bioelectrical Signal Processing in Cardiac and Neurological Applications*. Elsevier, 2005.

[8] Hans Berger. "Über das Elektrenkephalogramm des Menschen". In: *Archiv für Psychiatrie und Nervenkrankheiten* 87.1 (1929), pp. 527–570.

[9] Michael X Cohen. *Analyzing Neural Time Series Data: Theory and Practice*. MIT Press, 2014.

[10] Paul L Nunez and Ramesh Srinivasan. *Electric Fields of the Brain: The Neurophysics of EEG*. Oxford University Press, 2006.

[11] Steven J Luck. *An Introduction to the Event-Related Potential Technique*. MIT Press, 2014.

[12] Ernst Niedermeyer and Fernando Lopes da Silva, eds. *Electroencephalography: Basic Principles, Clinical Applications, and Related Fields*. 5th ed. Lippincott Williams & Wilkins, 2005.

[13] Sylvain Baillet, John C Mosher, and Richard M Leahy. "Electromagnetic brain mapping". In: *IEEE Signal Processing Magazine* 18.6 (2001), pp. 14–30.

[14] Cornelis J Stam and Elisabeth CW van Straaten. "Modern network science of neurological disorders". In: *Nature Reviews Neuroscience* 13.10 (2012), pp. 683–695.

[15] Roberta Grech et al. "Review on solving the inverse problem in EEG source analysis". In: *Journal of Neuroengineering and Rehabilitation* 5.1 (2008), pp. 1–33.

[16] György Buzsáki and Andreas Draguhn. "Neuronal oscillations in cortical networks". In: *Science* 304.5679 (2004), pp. 1926–1929.

[17] Wolfgang Klimesch. "EEG alpha and theta oscillations reflect cognitive and memory performance: a review and analysis". In: *Brain Research Reviews* 29.2-3 (1999), pp. 169–195.

[18] Jonathan R Wolpaw et al. "Brain-computer interfaces for communication and control". In: *Clinical Neurophysiology* 113.6 (2002), pp. 767–791.

[19] Niels Birbaumer and Leonardo G Cohen. "Brain-computer interfaces: communication and restoration of movement in paralysis". In: *The Journal of Physiology* 579.3 (2007), pp. 621–636.

[20] John H Gruzelier. "EEG-neurofeedback for optimising performance. I: A review of cognitive and affective outcome in healthy participants". In: *Neuroscience & Biobehavioral Reviews* 44 (2014), pp. 124–141.

[21] René J Huster et al. "Methods for simultaneous EEG-fMRI: an introductory review". In: *Journal of Neuroscience* 32.18 (2012), pp. 6053–6060.

[22] Scott Makeig et al. "Mining event-related brain dynamics". In: *Trends in Cognitive Sciences* 8.5 (2004), pp. 204–210.

[23] Yannick Roy et al. "Deep learning-based electroencephalography analysis: a systematic review". In: *Journal of Neural Engineering* 16.5 (2019), p. 051001.

[24] Robin Tibor Schirrmeister et al. "Deep learning with convolutional neural networks for EEG decoding and visualization". In: *Human Brain Mapping* 38.11 (2017), pp. 5391–5420.

[25] Pouya Bashivan, Irina Rish, and Steve Heisig. "Learning Representations from EEG with Deep Recurrent-Convolutional Neural Networks". In: *arXiv preprint arXiv:1511.06448* (2016).

[26] Ary L Goldberger, Zachary D Goldberger, and Alexei Shvilkin. *Goldberger's Clinical Electrocardiography: A Simplified Approach*. 9th ed. Elsevier, 2017.

[27] Augustus D Waller. "A demonstration on man of electromotive changes accompanying the heart's beat". In: *Journal of Physiology* 8.5 (1887), pp. 229–234.

[28] Willem Einthoven. "The different forms of the human electrocardiogram and their signification". In: *Lancet* 1 (1912), pp. 853–861.

[29] Borys Surawicz and Timothy K Knilans. *Chou's Electrocardiography in Clinical Practice: Adult and Pediatric*. 6th ed. Saunders Elsevier, 2008.

[30] Arthur J Moss et al. "Prophylactic implantation of a defibrillator in patients with myocardial infarction and reduced ejection fraction". In: *New England Journal of Medicine* 346.12 (2002), pp. 877–883.

[31] Pentti M Rautaharju, Borys Surawicz, and Leonard S Gettes. "AHA/ACCF/HRS Recommendations for the Standardization and Interpretation of the Electrocardiogram". In: *Journal of the American College of Cardiology* 53.11 (2009), pp. 982–991.

[32] E Lepeschkin and B Surawicz. "The Measurement of the Q-T Interval of the Electrocardiogram". In: *Circulation* 23.4 (1961), pp. 505–510.

[33] Paul Kligfield et al. "Recommendations for the Standardization and Interpretation of the Electrocardiogram: Part I: The Electrocardiogram and Its Technology: A Scientific Statement From the American Heart Association Electrocardiography and Arrhythmias Committee, Council on Clinical Cardiology; the American College of Cardiology Foundation; and the Heart Rhythm Society". In: *Circulation* 115.10 (2007), pp. 1306–1324.

[34] Gari D Clifford et al. "AF Classification from a Short Single Lead ECG Recording: The PhysioNet/Computing in Cardiology Challenge 2017". In: *Physiological Measurement* 38.5 (2020), pp. 831–848.

[35] Hang Du, Rebecca Pillai Riddell, and Xiaogang Wang. "A hybrid complex-valued neural network framework with applications to electroencephalogram

(EEG)". en. In: *Biomedical Signal Processing and Control* 85 (Aug. 2023), p. 104862. ISSN: 17468094. DOI: 10.1016/j.bspc.2023.104862. URL: https://linkinghub.elsevier.com/retrieve/pii/S1746809423002951.

[36] Y. LeCun et al. "Backpropagation Applied to Handwritten Zip Code Recognition". en. In: *Neural Computation* 1.4 (Dec. 1989), pp. 541–551. ISSN: 0899-7667, 1530-888X. DOI: 10.1162/neco.1989.1.4.541. URL: https://direct.mit.edu/neco/article/1/4/541-551/5515 (visited on 04/06/2023).

[37] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Las Vegas, NV, USA: IEEE, June 2016, pp. 770–778. ISBN: 9781467388511. DOI: 10.1109/CVPR.2016.90. URL: http://ieeexplore.ieee.org/document/7780459/ (visited on 04/06/2023).

[38] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet classification with deep convolutional neural networks". en. In: *Communications of the ACM* 60.6 (May 2017), pp. 84–90. ISSN: 0001-0782, 1557-7317. DOI: 10.1145/3065386. URL: https://dl.acm.org/doi/10.1145/3065386 (visited on 04/06/2023).

[39] Ross Girshick et al. "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation". In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. Columbus, OH, USA: IEEE, June 2014, pp. 580–587. ISBN: 9781479951185. DOI: 10.1109/CVPR.2014.81. URL: http://ieeexplore.ieee.org/document/6909475/ (visited on 04/06/2023).

[40] Ross Girshick. "Fast R-CNN". In: *2015 IEEE International Conference on Computer Vision (ICCV)*. Santiago, Chile: IEEE, Dec. 2015, pp. 1440–1448.

ISBN: 9781467383912. DOI: 10.1109/ICCV.2015.169. URL: http://ieeexplore.ieee.org/document/7410526/ (visited on 04/06/2023).

[41] Joseph Redmon et al. "You Only Look Once: Unified, Real-Time Object Detection". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Las Vegas, NV, USA: IEEE, June 2016, pp. 779–788. ISBN: 9781467388511. DOI: 10.1109/CVPR.2016.91. URL: http://ieeexplore.ieee.org/document/7780460/ (visited on 04/06/2023).

[42] Jonathan Long, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. ISSN: 1063-6919. June 2015, pp. 3431–3440. DOI: 10.1109/CVPR.2015.7298965.

[43] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.12 (Dec. 2017), pp. 2481–2495. ISSN: 0162-8828, 2160-9292, 1939-3539. DOI: 10.1109/TPAMI.2016.2644615. URL: https://ieeexplore.ieee.org/document/7803544/ (visited on 04/06/2023).

[44] Liang-Chieh Chen et al. "DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40.4 (Apr. 2018), pp. 834–848. ISSN: 0162-8828, 2160-9292. DOI: 10.1109/TPAMI.2017.2699184. URL: http://ieeexplore.ieee.org/document/7913730/ (visited on 04/06/2023).

[45] Yoon Kim. "Convolutional Neural Networks for Sentence Classification". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Lan-*

guage Processing (EMNLP). Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1746–1751. DOI: 10.3115/v1/D14-1181. URL: https://aclanthology.org/D14-1181.

[46] Alexis Conneau et al. "Very Deep Convolutional Networks for Text Classification". en. In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Valencia, Spain: Association for Computational Linguistics, 2017, pp. 1107–1116. DOI: 10.18653/v1/E17-1104. URL: http://aclweb.org/anthology/E17-1104 (visited on 04/06/2023).

[47] Andre Esteva et al. "Dermatologist-level classification of skin cancer with deep neural networks". en. In: *Nature* 542.7639 (Feb. 2017), pp. 115–118. ISSN: 0028-0836, 1476-4687. DOI: 10.1038/nature21056. URL: http://www.nature.com/articles/nature21056 (visited on 04/06/2023).

[48] Mohammad Havaei et al. "Brain tumor segmentation with Deep Neural Networks". en. In: *Medical Image Analysis* 35 (Jan. 2017), pp. 18–31. ISSN: 13618415. DOI: 10.1016/j.media.2016.05.004. URL: https://linkinghub.elsevier.com/retrieve/pii/S1361841516300330 (visited on 04/06/2023).

[49] Akira Hirose. *Complex-Valued Neural Networks: Theories and Applications (Series on Innovative Intelligence, 5)*. World Scientific Press, 2004. ISBN: 9789812384645.

[50] Nitzan Guberman. "On Complex Valued Convolutional Neural Networks". In: *CoRR* abs/1602.09046 (2016). arXiv: 1602.09046. URL: http://arxiv.org/abs/1602.09046.

[51] Chiheb Trabelsi et al. “Deep Complex Networks”. In: *CoRR* abs/1705.09792 (2017). arXiv: 1705.09792. URL: http://arxiv.org/abs/1705.09792.

[52] Patrick Virtue. “Complex-valued Deep Learning with Applications to Magnetic Resonance Image Synthesis”. PhD thesis. University of California at Berkeley, 2019.

[53] Joan Bruna and S. Mallat. “Invariant Scattering Convolution Networks”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.8 (Aug. 2013), pp. 1872–1886. ISSN: 0162-8828, 2160-9292. DOI: 10.1109/TPAMI.2012.230. URL: http://ieeexplore.ieee.org/document/6522407/ (visited on 07/20/2021).

[54] David P. Reichert and Thomas Serre. *Neuronal Synchrony in Complex-Valued Deep Networks*. 2014. arXiv: 1312.6115 [stat.ML].

[55] Tara N. Sainath et al. “Low-rank matrix factorization for Deep Neural Network training with high-dimensional output targets”. In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. 2013, pp. 6655–6659. DOI: 10.1109/ICASSP.2013.6638949.

[56] W. Heisenberg. “Über den anschaulichen Inhalt der quantentheoretischen Kinematik und Mechanik”. In: *Zeitschrift für Physik* 33 (1925), pp. 879–893.

[57] Aravind Ravi et al. “Comparing user-dependent and user-independent training of CNN for SSVEP BCI”. In: *Journal of Neural Engineering* 17.2 (Apr. 2020), p. 026028. ISSN: 1741-2560, 1741-2552. DOI: 10.1088/1741-2552/ab6a67. (Visited on 01/09/2023).

[58] W. Wirtinger. “Zur formalen Theorie der Funktionen von mehr komplexen Veränderlichen”. de. In: *Mathematische Annalen* 97.1 (Dec. 1927), pp. 357–

375. ISSN: 0025-5831, 1432-1807. DOI: 10.1007/BF01447872. URL: http://link.springer.com/10.1007/BF01447872 (visited on 03/17/2021).

[59] Huisheng Zhang and Danilo P. Mandic. "Is a Complex-Valued Stepsize Advantageous in Complex-Valued Gradient Learning Algorithms?" In: *IEEE Transactions on Neural Networks and Learning Systems* 27.12 (Dec. 2016), pp. 2730–2735. ISSN: 2162-237X, 2162-2388. DOI: 10.1109/TNNLS.2015.2494361. URL: http://ieeexplore.ieee.org/document/7321072/ (visited on 07/20/2021).

[60] Walter Rudin. *Function Theory in the Unit Ball of* $\mathbb{C}^n$. Ed. by M. Artin et al. Vol. 241. Grundlehren der mathematischen Wissenschaften. New York, NY: Springer New York, 1980. DOI: 10.1007/978-1-4613-8098-6. URL: http://link.springer.com/10.1007/978-1-4613-8098-6 (visited on 07/20/2021).

[61] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[62] Ralph Andrzejak et al. "Indications of nonlinear deterministic and finite-dimensional structures in time series of brain electrical activity: Dependence on recording region and brain state". In: *Physical review. E, Statistical, nonlinear, and soft matter physics* 64 (Jan. 2002), p. 061907. DOI: 10.1103/PhysRevE.64.061907.

[63] Y. Lecun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. DOI: 10.1109/5.726791.

[64] Alex Krizhevsky. "Learning Multiple Layers of Features from Tiny Images". In: *University of Toronto* (May 2012).

[65] Jordi Feliu-Fabà, Yuwei Fan, and Lexing Ying. "Meta-learning pseudo-differential operators with deep neural networks". In: *Journal of Computational Physics* 408 (2020), p. 109309. ISSN: 0021-9991. DOI: https://doi.org/10.1016/j.jcp.2020.109309. URL: https://www.sciencedirect.com/science/article/pii/S0021999120300838.

[66] Jordi Feliu-Fabà, Yuwei Fan, and Lexing Ying. "Meta-learning pseudo-differential operators with deep neural networks". In: *Journal of Computational Physics* 408 (2020), p. 109309. ISSN: 0021-9991.

[67] Christopher Rackauckas et al. "Universal Differential Equations for Scientific Machine Learning". In: *arXiv preprint arXiv:2001.04385* (2020).

[68] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. Adaptive computation and machine learning. Cambridge, Massachusetts: The MIT Press, 2016. ISBN: 9780262035613.

[69] Vernon Lawhern et al. "EEGNet: A Compact Convolutional Network for EEG-based Brain-Computer Interfaces". In: *Journal of Neural Engineering* 15 (Nov. 2016).

[70] Xia Chen et al. *Toward reliable signals decoding for electroencephalogram: A benchmark study to EEGNeX*. 2022.

[71] Thoru Yamada and Elizabeth Meng. *Practical Guide for Clinical Neurophysiologic Testing: EEG*. Philadelphia, PA: Wolters Kluwer Health, Oct. 2017. ISBN: 9781496383037. URL: https://shop.lww.com/Practical-Guide-for-Clinical-Neurophysiologic-Testing%E2%80%93EEG/p/9781496383020.

[72] Ujwal Chaudhary, Niels Birbaumer, and Ander Ramos-Murguialday. "Brain–computer interfaces for communication and rehabilitation". en. In: *Nature Reviews Neurology* 12.9 (Sept. 2016), pp. 513–525. ISSN: 1759-4758, 1759-4766. DOI: 10.1038/nrneurol.2016.113.

[73] Sriram Ramgopal et al. "Seizure detection, seizure prediction, and closed-loop warning systems in epilepsy". In: *Epilepsy and Behavior* 37 (2014), pp. 291–307. ISSN: 1525-5050. DOI: https://doi.org/10.1016/j.yebeh.2014.06.023. URL: https://www.sciencedirect.com/science/article/pii/S1525505014002297.

[74] Masaki Nakanishi et al. "A Comparison Study of Canonical Correlation Analysis Based Methods for Detecting Steady-State Visual Evoked Potentials". en. In: *PLOS ONE* 10.10 (Oct. 2015). Ed. by Dezhong Yao, e0140703. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0140703. (Visited on 01/09/2023).

[75] Xiaotong Gu et al. "EEG-Based Brain-Computer Interfaces (BCIs): A Survey of Recent Studies on Signal Sensing Technologies and Computational Intelligence Approaches and Their Applications". In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 18.5 (2021), pp. 1645–1666. DOI: 10.1109/TCBB.2021.3052811.

[76] Berkant Tacer and Patrick J. Loughlin. "Non-stationary signal classification using the joint moments of time-frequency distributions". In: *Pattern Recognition* 31.11 (1998), pp. 1635–1641. ISSN: 0031-3203.

[77] Manuel Davy et al. "Optimized support vector machines for nonstationary signal classification". In: *Signal Processing Letters, IEEE* 9 (Jan. 2003), pp. 442–445.

[78] L. Boubchir, S. Al-Maadeed, and A. Bouridane. "On the use of time-frequency features for detecting and classifying epileptic seizure activities in non-stationary EEG signals". In: *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2014, pp. 5889–5893.

[79] Omer Alcin et al. "Multi-category EEG signal classification developing Time-Frequency Texture Features based Fisher Vector encoding method". In: *Neurocomputing* 218 (Aug. 2016).

[80] Hamid Hassanpour, Mostefa Mesbah, and Boualem Boashash. "Time–frequency based newborn EEG seizure detection using low and high frequency signatures". In: *Physiological Measurement* 25.4 (Aug. 2004), pp. 935–944. ISSN: 0967-3334, 1361-6579.

[81] A. T. Tzallas, M. G. Tsipouras, and D. I. Fotiadis. "Epileptic Seizure Detection in EEGs Using Time–Frequency Analysis". In: *IEEE Transactions on Information Technology in Biomedicine* 13.5 (2009), pp. 703–710.

[82] Boualem Boashash, Ghasem Azemi, and Nabeel Khan. "Principles of time–frequency feature extraction for change detection in non-stationary signals: Applications to newborn EEG abnormality detection". In: *Pattern Recognition* 48 (Mar. 2015), pp. 616–627.

[83] M. R. Islam and M. Ahmad. "Wavelet Analysis Based Classification of Emotion from EEG Signal". In: *2019 International Conference on Electrical, Computer and Communication Engineering (ECCE)*. 2019, pp. 1–6.

[84] Fasil O.K. and Rajesh R. "Time-domain exponential energy for epileptic EEG signal classification". en. In: *Neuroscience Letters* 694 (Feb. 2019), pp. 1–8. ISSN: 03043940.

[85] Rubén San-Segundo et al. "Classification of epileptic EEG recordings using signal transforms and convolutional neural networks". In: *Computers in Biology and Medicine* 109 (2019), pp. 148–158. ISSN: 0010-4825.

[86] Dongye Zhao et al. "Learning joint space-time-frequency features for EEG decoding on small labeled data". In: *Neural Networks* 114 (Mar. 2019).

[87] Yunyuan Gao et al. "Deep Convolutional Neural Network-Based Epileptic Electroencephalogram (EEG) Signal Classification". In: *Frontiers in Neurology* 11 (2020), p. 375. ISSN: 1664-2295.

[88] Zhongke Gao et al. "Complex networks and deep learning for EEG signal analysis". In: *Cognitive Neurodynamics* 15.3 (June 2021), pp. 369–388. ISSN: 1871-4080, 1871-4099.

[89] Akira Hirose and Shotaro Yoshida. "Generalization Characteristics of Complex-Valued Feedforward Neural Networks in Relation to Signal Coherence". In: *IEEE Transactions on Neural Networks and Learning Systems* 23.4 (2012), pp. 541–551.

[90] Nitzan Guberman. "On Complex Valued Convolutional Neural Networks". In: *CoRR* abs/1602.09046 (2016). arXiv: 1602.09046. URL: http://arxiv.org/abs/1602.09046.

[91] Akira Ikeda and Yoshikazu Washizawa. "Steady-State Visual Evoked Potential Classification Using Complex Valued Convolutional Neural Networks". en. In: *Sensors* 21.16 (Aug. 2021), p. 5309. ISSN: 1424-8220. DOI: 10.3390/s21165309. (Visited on 01/09/2023).

[92] Chiheb Trabelsi et al. "Deep Complex Networks". In: *arXiv preprint: arXiv:1705.09792* (2017).

[93] Simone Scardapane et al. “Complex-Valued Neural Networks With Nonparametric Activation Functions”. In: *IEEE Transactions on Emerging Topics in Computational Intelligence* PP (Feb. 2018).

[94] Joshua Bassey, Lijun Qian, and Xianfang Li. *A Survey of Complex-Valued Neural Networks*. 2021. arXiv: 2101.12249 [stat.ML].

[95] Ralph Andrzejak et al. “Indications of nonlinear deterministic and finite-dimensional structures in time series of brain electrical activity: Dependence on recording region and brain state”. In: *Physical review. E, Statistical, nonlinear, and soft matter physics* 64 (Jan. 2002), p. 061907.

[96] Diederik Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: *International Conference on Learning Representations* (Dec. 2014).

[97] Andy M. Sarroff. “Complex Neural Networks For Audio”. PhD thesis. Dartmouth College, 2018.

[98] Xavier Glorot and Yoshua Bengio. “Understanding the difficulty of training deep feedforward neural networks”. In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. Vol. 9. Proceedings of Machine Learning Research. JMLR Workshop and Conference Proceedings, May 2010, pp. 249–256.

[99] Nick Yeung et al. “Detection of synchronized oscillations in the electroencephalogram: An evaluation of methods”. en. In: *Psychophysiology* 41.6 (Nov. 2004), pp. 822–832. ISSN: 0048-5772, 1469-8986.

[100] Nick Yeung et al. “Theta phase resetting and the error-related negativity”. en. In: *Psychophysiology* 44.1 (Jan. 2007). ISSN: 0048-5772, 1469-8986.

[101] Christos E. Vasios. "Classification of Event-Related Potentials Associated with Response Errors in Actors and Observers Based on Autoregressive Modeling". en. In: *The Open Medical Informatics Journal* 3.1 (May 2009), pp. 32–43. ISSN: 18744311.

[102] Sutrisno Ibrahim, Ridha Djemal, and Abdullah Alsuwailem. "Electroencephalography (EEG) signal processing for epilepsy and autism spectrum disorder diagnosis". en. In: *Biocybernetics and Biomedical Engineering* 38.1 (2018), pp. 16–26. ISSN: 02085216.

[103] Anindya Bijoy Das and Mohammed Imamul Hassan Bhuiyan. "Discrimination and classification of focal and non-focal EEG signals using entropy-based features in the EMD-DWT domain". en. In: *Biomedical Signal Processing and Control* 29 (Aug. 2016), pp. 11–21. ISSN: 17468094.

[104] Hansem Sohn et al. "Approximate entropy (ApEn) analysis of EEG in attention-deficit/hyperactivity disorder (ADHD) during cognitive tasks". In: vol. 14. Aug. 2006, pp. 1083–1086. ISBN: 978-3-540-36839-7.

[105] Joshua S. Richman and J. Randall Moorman. "Physiological time-series analysis using approximate entropy and sample entropy". en. In: *American Journal of Physiology-Heart and Circulatory Physiology* 278.6 (June 2000), H2039–H2049. ISSN: 0363-6135, 1522-1539.

[106] Peng Li et al. "Assessing the complexity of short-term heartbeat interval series by distribution entropy". en. In: *Medical & Biological Engineering & Computing* 53.1 (Jan. 2015), pp. 77–87. ISSN: 0140-0118, 1741-0444.

[107] Samantha Simons, Pedro Espino, and Daniel Abásolo. "Fuzzy Entropy Analysis of the Electroencephalogram in Patients with Alzheimer's Disease: Is the

Method Superior to Sample Entropy?" en. In: *Entropy* 20.1 (Jan. 2018), p. 21. ISSN: 1099-4300.

[108] Weiting Chen et al. "Measuring complexity using FuzzyEn, ApEn, and SampEn". en. In: *Medical Engineering & Physics* 31.1 (Jan. 2009), pp. 61–68. ISSN: 13504533.

[109] Bo Shi et al. "Entropy Analysis of Short-Term Heartbeat Interval Time Series during Regular Walking". en. In: *Entropy* 19.10 (Oct. 2017), p. 568. ISSN: 1099-4300.

[110] S. Escalera, O. Pujol, and P. Radeva. "On the Decoding Process in Ternary Error-Correcting Output Codes". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32.1 (Jan. 2010), pp. 120–134. ISSN: 0162-8828.

[111] Sergio Escalera, Oriol Pujol, and Petia Radeva. "Separability of ternary codes for sparse designs of error-correcting output codes". en. In: *Pattern Recognition Letters* 30.3 (Feb. 2009), pp. 285–297. ISSN: 01678655.

[112] Peng Li et al. "Detection of epileptic seizure based on entropy analysis of short-term EEG". en. In: *PLOS ONE* 13.3 (Mar. 2018). Ed. by Maxim Bazhenov, e0193691. ISSN: 1932-6203.

[113] Sándor Beniczky and Donald L. Schomer. "Electroencephalography: basic biophysical and technological aspects important for clinical applications". In: *Epileptic Disorders* 22.6 (Dec. 2020), pp. 697–715. ISSN: 1294-9361, 1950-6945. (Visited on 07/27/2022).

[114] *Simulated EEG data generator*. URL: https://data.mrc.ox.ac.uk/data-set/simulated-eeg-data-generator (visited on 09/12/2021).

[115] Ville Mäkinen, Hannu Tiitinen, and Patrick May. "Auditory event-related responses are generated independently of ongoing brain activity". In: *NeuroImage* 24.4 (Feb. 2005), pp. 961–968. ISSN: 10538119.

[116] Lina Wang et al. "Automatic Epileptic Seizure Detection in EEG Signals Using Multi-Domain Feature Extraction and Nonlinear Analysis". In: *Entropy* 19.6 (May 2017), p. 222. ISSN: 1099-4300.

[117] T. Sunil Kumar, Vivek Kanhangad, and Ram Bilas Pachori. "Classification of seizure and seizure-free EEG signals using local binary patterns". In: *Biomedical Signal Processing and Control* 15 (2015), pp. 33–40. ISSN: 1746-8094.

[118] Khalid Abualsaud et al. "Ensemble Classifier for Epileptic Seizure Detection for Imperfect EEG Data". In: *The Scientific World Journal* 2015 (2015), pp. 1–15. ISSN: 2356-6140, 1537-744X.

[119] N. Sadati, H. R. Mohseni, and A. Maghsoudi. "Epileptic Seizure Detection Using Neural Fuzzy Networks". In: *2006 IEEE International Conference on Fuzzy Systems*. 2006, pp. 596–600.

[120] Qiuyi Wu and Ernest Fokoue. *Epileptic Seizure Recognition Data Set*. 2017. URL: https://archive.ics.uci.edu/ml/datasets/Epileptic+Seizure+Recognition.

[121] Dheeru Dua and Casey Graff. *UCI Machine Learning Repository*. 2017. URL: http://archive.ics.uci.edu/ml.

[122] Nicholas Waytowich et al. "Compact Convolutional Neural Networks for Classification of Asynchronous Steady-state Visual Evoked Potentials". In: *Journal of Neural Engineering* 15 (Mar. 2018). DOI: 10.1088/1741-2552/aae5d8.

[123] *Cardiovascular diseases (cvds)*. URL: https://www.who.int/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds).

[124] R. K. Tripathy and S. Dandapat. "Detection of myocardial infarction from vectorcardiogram using relevance vector machine". en. In: *Signal, Image and Video Processing* 11.6 (Sept. 2017), pp. 1139–1146. ISSN: 1863-1703, 1863-1711. (Visited on 07/01/2022).

[125] Sibghatullah I. Khan and Ram Bilas Pachori. "Automated Detection of Posterior Myocardial Infarction From Vectorcardiogram Signals Using Fourier–Bessel Series Expansion Based Empirical Wavelet Transform". In: *IEEE Sensors Letters* 5.5 (2021), pp. 1–4.

[126] Hui Yang et al. "Identification of myocardial infarction (MI) using spatio-temporal heart dynamics". en. In: *Medical Engineering & Physics* 34.4 (May 2012), pp. 485–497. ISSN: 13504533.

[127] Lars Edenbrandt and Olle Pahlm. "Vectorcardiogram synthesized from a 12-lead ECG: Superiority of the inverse Dower matrix". In: *Journal of Electrocardiology* 21.4 (1988), pp. 361–367. ISSN: 0022-0736.

[128] J. A. Kors et al. "Reconstruction of the Frank vectorcardiogram from standard electrocardiographic leads: diagnostic comparison of different methods". en. In: *European Heart Journal* 11.12 (Dec. 1990), pp. 1083–1092. ISSN: 1522-9645, 0195-668X. DOI: 10.1093/oxfordjournals.eurheartj.a059647. URL: https://academic.oup.com/eurheartj/article/525149/Reconstruction (visited on 03/26/2021).

[129] Gordon E. Dower, H. B. Machado, and John A. Osborne. "On Deriving the Electrocardiogram from Vectorcardiographic Leads". In: *Clinical Cardiology* 3 (1980).

[130] Gordon E. Dower. "A lead synthesizer for the Frank system to simulate the standard 12-lead electrocardiogram". In: *Journal of Electrocardiology* 1.1 (1968), pp. 101–116. ISSN: 0022-0736.

[131] Nastaran Jafari Hafshejani et al. "Identification of myocardial infarction using morphological features of electrocardiogram and vectorcardiogram". en. In: *IET Signal Processing* 15.9 (Dec. 2021), pp. 674–685. ISSN: 1751-9675, 1751-9683. (Visited on 03/12/2022).

[132] Eedara Prabhakararao and Samarendra Dandapat. "Automated Detection of Posterior Myocardial Infarction From VCG Signals Using Stationary Wavelet Transform Based Features". In: *IEEE Sensors Letters* 4.6 (2020), pp. 1–4.

[133] U. Rajendra Acharya et al. "Automated detection and localization of myocardial infarction using electrocardiogram: a comparative study of different leads". In: *Knowledge-Based Systems* 99 (2016), pp. 146–156. ISSN: 0950-7051.

[134] Rajesh Kumar Tripathy, Abhijit Bhattacharyya, and Ram Bilas Pachori. "Localization of Myocardial Infarction From Multi-Lead ECG Signals Using Multiscale Analysis and Convolutional Neural Network". In: *IEEE Sensors Journal* 19.23 (2019), pp. 11437–11448.

[135] Filip Karisik and Mathias Baumert. "A Long Short-Term Memory Network to Classify Myocardial Infarction Using Vectorcardiographic Ventricular Depolarization and Repolarization". In: *2019 Computing in Cardiology (CinC)*. 2019, Page 1–Page 4.

[136] Yu-Hung Chuang et al. "Automatic Classification of Myocardial Infarction Using Spline Representation of Single-Lead Derived Vectorcardiography". en. In: *Sensors* 20.24 (Dec. 2020), p. 7246. ISSN: 1424-8220. (Visited on 05/27/2022).

[137] Xiaofei Sun et al. *Interpreting Deep Learning Models in Natural Language Processing: A Review*. 2021.

[138] Raúl Correa et al. "Identification of Patients with Myocardial Infarction: Vectorcardiographic and Electrocardiographic Analysis". en. In: *Methods of Information in Medicine* 55.03 (2016), pp. 242–249. ISSN: 0026-1270, 2511-705X.

[139] R. Bousseljot, D. Kreiseler, and A. Schnabel. "Nutzung der EKG-Signaldatenbank CARDIODAT der PTB über das Internet". In: *Biomedizinische Technik/Biomedical Engineering* (July 2009), pp. 317–318. ISSN: 0013-5585, 1862-278X.

[140] Patrick Wagner et al. "PTB-XL, a large publicly available electrocardiography dataset". en. In: *Scientific Data* 7.1 (Dec. 2020), p. 154. ISSN: 2052-4463.

[141] Jan A. Kors, Jan L. Talmon, and Jan H. van Bemmel. "Multilead ECG analysis". In: *Computers and Biomedical Research* 19.1 (1986), pp. 28–46. ISSN: 0010-4809.

[142] M. S. Guillem, A. V. Sahakian, and S. Swiryn. "Derivation of orthogonal leads from the 12-lead ECG. accuracy of a single transform for the derivation of atrial and ventricular waves". In: *2006 Computers in Cardiology*. 2006, pp. 249–252.

[143] A. L. Goldberger et al. "PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals". In: *Circulation* 101.23 (2000), e215–e220.

[144] Nicolas Pilia et al. “ECGdeli - An open source ECG delineation toolbox for MATLAB”. In: *SoftwareX* 13 (2021), p. 100639. ISSN: 2352-7110.

[145] Vilém Laufberger. “Octant vectorcardiography and its data basis”. In: *Physiologia bohemoslovaca* 30.6 (1981), pp. 481–495.

[146] Vilém Laufberger. “Octant vectorcardiography”. In: *Physiologia bohemoslovaca* 29.6 (1982), pp. 481–494.

[147] Vilém Laufberger. “Octant vectorcardiography - the evaluation by peaks”. In: *Physiol Bohemoslov* 31 (1982), pp. 1–9.

[148] Vilém Laufberger. “Octant vectorcardiography and automatic diagnosis of coronary artery disease.” In: *Physiologia bohemoslovaca* 31.6 (1982), pp. 485–495.

[149] Golriz Sedaghat et al. “Quantitative Assessment of Vectorcardiographic Loop Morphology”. en. In: *Journal of Electrocardiology* 49.2 (Mar. 2016), pp. 154–163. ISSN: 00220736. (Visited on 06/30/2022).

[150] Manfredo Perdigão do Carmo. *Differential geometry of curves and surfaces*. Prentice-Hall, 1976, pp. 21–22.

[151] Lei Nie et al. “Approximate Derivative Calculated by Using Continuous Wavelet Transform”. en. In: *Journal of Chemical Information and Computer Sciences* 42.2 (Mar. 2002), pp. 274–283. ISSN: 0095-2338.

[152] karthik raviprakash karthik. *ECG simulation using MATLAB*. URL: https://www.mathworks.com/matlabcentral/fileexchange/10858-ecg-simulation-using-matlab.

[153] A. Messina. “Detecting damage in beams through digital differentiator filters and continuous wavelet transforms”. In: *Journal of Sound and Vibration* 272.1 (2004), pp. 385–412. ISSN: 0022-460X.

[154] S. G. Mallat. *A wavelet tour of signal processing*. 2nd ed. San Diego: Academic Press, 1999. ISBN: 9780124666061.

[155] Jianwen Luo, Bai Jing, and Shao Jinhua. “Application of the wavelet transforms on axial strain calculation in ultrasound elastography”. In: *Progress in Natural Science* 16 (Sept. 2006), pp. 942–947.

[156] Marko Robnik-Šikonja and Igor Kononenko. “Theoretical and empirical analysis of ReliefF and RReliefF”. In: *Machine learning* 53.1 (2003), pp. 23–69.

[157] Igor Kononenko. “Estimating attributes: Analysis and extensions of RELIEF”. In: (1994), pp. 171–182.

[158] Ryan J. Urbanowicz et al. “Relief-based feature selection: Introduction and review”. In: *Journal of Biomedical Informatics* 85 (2018), pp. 189–203. ISSN: 1532-0464.

[159] Kenji Kira and Larry A. Rendell. “The Feature Selection Problem: Traditional Methods and a New Algorithm”. In: (1992).

[160] Hui Yang. “Nonlinear Stochastic Modeling and Analysis of Cardiovascular System Dynamics - Diagnostic and Prognostic Applications”. Ph.D. thesis. Oklahoma State University.

[161] Michal Vozda, Tomás Peterek, and Martin Cerný. “Novel Method for Deriving Vectorcardiographic Leads Based on Artificial Neural Networks”. In: 2014.

[162] Rene Jaros, Radek Martinek, and Lukas Danys. “Comparison of Different Electrocardiography with Vectorcardiography Transformations”. en. In: *Sensors* 19.14 (July 2019), p. 3072. ISSN: 1424-8220. DOI: 10.3390/s19143072. URL: https://www.mdpi.com/1424-8220/19/14/3072 (visited on 03/26/2021).

[163] T. G. Dietterich and G. Bakiri. “Solving Multiclass Learning Problems via Error-Correcting Output Codes”. In: *Journal of Artificial Intelligence Research* 2 (Jan. 1995), pp. 263–286. ISSN: 1076-9757.

A Appendix

A.1 ECG to VCG transformation matrix

The following linear transformation matrices (ECG to VCG) are used in chapter 5:

The Kors Quasi-Orthogonal Transformation matrix:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -0.5 & 0 & 0 & 0 & 0 \end{bmatrix}$$

The Dower transformation matrix:

$$\begin{bmatrix} 0.156 & -0.010 & -0.172 & -0.074 & 0.122 & 0.231 & 0.239 & 0.194 \\ -0.227 & 0.887 & 0.057 & -0.019 & -0.106 & -0.022 & 0.041 & 0.048 \\ 0.022 & 0.102 & -0.229 & -0.310 & -0.246 & -0.063 & 0.055 & 0.108 \end{bmatrix}$$

The Kors transformation matrix:

$$\begin{bmatrix} 0.38 & -0.07 & -0.13 & 0.05 & -0.01 & 0.14 & 0.06 & 0.54 \\ -0.07 & 0.93 & 0.06 & -0.02 & -0.05 & 0.06 & -0.17 & 0.13 \\ 0.11 & -0.23 & -0.43 & -0.06 & -0.14 & -0.20 & -0.11 & 0.31 \end{bmatrix}$$

The QLSV transformation matrix:

$$\begin{bmatrix} 0.199 & -0.018 & -0.147 & -0.058 & 0.037 & 0.139 & 0.232 & 0.226 \\ -0.146 & 0.503 & 0.023 & -0.085 & -0.003 & 0.033 & 0.060 & 0.104 \\ 0.085 & -0.130 & -0.184 & -0.163 & -0.190 & -0.119 & -0.023 & 0.043 \end{bmatrix}$$

The PLSV transformation matrix:

$$\begin{bmatrix} 0.370 & -0.154 & -0.266 & 0.027 & 0.065 & 0.131 & 0.203 & 0.220 \\ -0.131 & 0.717 & 0.088 & -0.088 & 0.003 & 0.042 & 0.047 & 0.067 \\ 0.184 & -0.114 & -0.319 & -0.198 & -0.167 & -0.099 & -0.009 & 0.060 \end{bmatrix}$$