APPROVAL SHEET

Title of Thesis: Representation Learning on Time Series with Symbolic Approximation and Deep Learning

Name of Candidate: Zhiguang Wang
Doctor of Philosophy
May, 2016

Thesis and Abstract Approved: _____
Tim Oates
Professor
Department of Computer Science and
Electrical Engineering

Date Approved: _____

ABSTRACT

Title of Thesis: Representation Learning on Time Series with Symbolic Approximation and Deep Learning

Thesis directed by: Tim Oates, Professor
Department of Computer Science and
Electrical Engineering

Most real-world data has a temporal component, whether it is measurements of natural (weather, sound) or man-made (stock market, robotics and even speech and language) phenomena. Analysis of temporal data has been the subject of active research for decades and is still considered to be a challenge in machine learning and data mining, due to the intrinsically structured temporal correlation.

In this thesis, we propose three different novel approaches to represent and model time-series. Time-Warping SAX and Pooling SAX are two extensions of the vanilla SAX approach that is used as a symbolic representation of time series. Time-Warping SAX extracts linear temporal dependencies by building a time-delay embedding vector to construct more informative SAX words. Pooling SAX applies a non-parametric weighting scheme to extract significant variables. These are data adaptive models that achieve state-of-the-art accuracy on time-series classification problems.

We also propose the Gramian Angular Field (GAF) and Markov Transition Field (MTF) as two novel approaches to encode a time-series as an image. These representations not only demonstrate potential for visual inspection by humans, but when they are combined with deep learning approaches (Convolutional Networks and Denoised Auto-encoders). They achieve state-of-the-art performance compared to other modern algorithms on classification and regression/imputation problems for different type of temporal data and trajectories. GAF and MTF are non-data adaptive approaches that allow us to learn models and extract the abstract representations supported by model-based approaches.

Finally, we develop a set of exponential-form based error estimator (NRAE/NAAE) with their learning approaches (Adaptive Training) to attach the non-convex optimization problems in training deep neural networks. Both in theory and practice, they are able to achieve optimality on accuracy and robustness against outliers/noise. They provide another perspectives to debunk the non-convexity of deep learning in high dimensional learning and recurrent architectures and benefit the modeling of high-dimensional temporal data.

Representation Learning on Time Series with Symbolic Approximation and Deep Learning

by

Zhiguang Wang

Thesis submitted to the Faculty of the Graduate School
of the University of Maryland in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2016

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

A time series represents a sequence of values obtained from measurements over time. Time series learning stems from the desire to reify our natural ability to analyze the shape of data. Humans rely on complex schemes to perform such tasks. Major time-series-related tasks include query by content, anomaly detection, motif discovery, prediction, clustering and classification. Despite the vast body of work devoted to this topic in the early years, (Antunes & Oliveira 2001) noted that the research has not been driven so much by actual problems but by an interest in proposing new approaches. But with the ever-growing maturity of time series learning techniques, this statement seems to have become obsolete. Nowadays, time-series analysis covers a wide range of real-life problems in various fields of research. Some examples include economic forecasting, intrusion detection, gene expression analysis, medical surveillance, hydrology and geography. Time-series learning is complex. The most prominent problems arise from the high dimensionality of time-series data and the difficulty of defining a form of representation based on the characteristics embedded in time series. With the rapid growth of digital sources of information, time series learning algorithms will have to match increasingly complicated datasets.

Time series representation and modeling are at the core of time-series learning systems, but those available approaches are not exhaustive as many tasks will require the use

of more specific models. For example, forecasting and imputation may require the use of a time series representation and a notion of similarity (mostly used to measure prediction accuracy) whereas model selection and statistical learning are also at the core. Moreover, high-dimensional, and noisy real-world time-series data cannot be described with analytical equations with parameters to solve since the dynamics are either too complex or unknown and traditional shallow methods, which contain a small number of non-linear operations, might be fail to accurately model such complex systems.

In this thesis, we improve and develop the approaches from two perspectives, i.e. Symbolic approximation and imaging with deep learning, to learn the representation and model time series. Based on the Symbolic Aggregate Approximation (SAX) method (Lin *et al.* 2003), we develop two novel extensions called time-Warping SAX and pooling SAX. Time-Warping SAX successfully extract the linear temporal dependencies by time-delay embedding vector to build more informative SAX words and thus Bag-of-Patterns. Pooling SAX apply a non-parametric weighting scheme to extract significant variables. It works well on handling the synchronous multivariate physiological time series. As data adaptive models, they both achieve good results on time series classification problems. Gramian Angular Field (GAF) and Markov Transition Field (MTF) are two novel approach to encode time series as images. These representations not only demonstrate potential for visual inspection by humans, but also achieve the state-of-the-art performance on classification and regression/imputation problems with deep learning approaches (Convolutional Networks and Denoised Auto-encoders). GAF and MTF are nondata adaptive approach, from which we learn the model and abstract representations with model-based approach.

When training deep neural networks (including convolutional networks and stacked autoencoders), the non-linearity always leads to a non-convex optimization problem. Inspired by previous work of Normalized Risk-Averting Error (NRAE) (Lo, Gui, & Peng 2012), we proposes a novel learning approach, Adaptive Normalized Risk-Averting Train-

ing (ANRAT) to attack the non-convex optimization problem in training deep neural networks. Theoretically, we demonstrate its effectiveness based on the expansion of the convexity region. By analyzing the gradient on the convexity index λ, we explain the reason why our learning method using gradient descent works. In practice, we show how this training method is successfully applied for improved training of deep neural networks to solve visual recognition tasks on the MNIST and CIFAR-10 datasets. Performance on deep/shallow multilayer perceptron and Denoised Auto-encoder is also explored. ANRAT can be combined with other quasi-Newton training methods, innovative network variants, regularization techniques and other common tricks in Deep Neural Networks (DNNs). Other than unsupervised pretraining, it provides a new perspective to address the non-convex optimization strategy in training DNNs. By pushing the robust-optimal (RO) index λ to $-\infty$, we are able to achieve robustness to outliers by optimizing a quasi-Minimin function. The robustness is realized and controlled adaptively by the RO index without any predefined threshold. Optimality is guaranteed by expansion of the convexity region in the Hessian matrix to largely avoid local optima. Detailed quantitative analysis on both robustness and optimality are provided. The results of proposed experiments on fitting tasks for three noisy non-convex functions and the digits recognition task on the MNIST dataset consolidate the conclusions.

The contribution of this thesis stems from the effort to explore three different technologies in time series modeling and the optimization problems coming along within modeling using DNNs. Time-warping SAX is able to capture the more complicated temporal correlations in multiple time scales, but when the temporal data is not primarily relying on the temporal part (like the shape data which is not a 'natural' time series), the advantage is not obvious compared to the vanilla SAX approach. Pooling SAX successfully extends SAX to multivariate temporal data in a non-parametric manner. Empirically, when the number of channels is smaller than 6, Pooling SAX always works better than or competitive

the current state-of-the-art approaches like SMTS (Baydogan & Runger 2014) or MTSBF (Baydogan, Runger, & Tuv 2013). Modeling temporal correlations as images to learn the features automatically using DNNs is a more general framework for a more wide range of temporal data modeling problem (signal and trajectory), which achieves the state-of-the-art results in classification and imputation tasks. Finally, our proposed non-convex optimization estimator and learning approach cast light on the future research of deep learning in high dimensional learning and recurrent architectures.

Chapter 2

BACKGROUND

2.1 Related Work

As mentioned earlier, time series are essentially high-dimensional data. Defining algorithms that work directly on the raw time series would therefore be computationally too expensive. The main motivation of representations is thus to emphasize the essential characteristics of the data in a concise and informative way. Additional benefits gained are efficient storage, speedup of processing, as well as implicit noise removal. Many representation techniques have been investigated. It is, however, possible to classify these approaches according to the kind of transformations applied. In order to perform such classification, we follow the taxonomy of (Keogh, Lonardi, & Ratanamahatana 2004) by dividing representations into three categories, namely non-data adaptive, data adaptive, and model based.

In non-data adaptive representations, the parameters of the transformation remain the same for every time series regardless of its nature. The Discrete Fourier Transformation (DFT) was used in the seminal work of (Agrawal, Faloutsos, & Swami 1993). It projects the time series on a sine and cosine function basis domain (Faloutsos, Ranganathan, & Manolopoulos 1994). The resulting representation is a set of sinusoidal coefficients. Instead of using a fixed set of basis functions, the DWT uses scaled and shifted versions of

a mother wavelet function (Chan, Fu, & Yu 2003). This gives a multiresolution decomposition where low frequencies are measured over larger intervals, thus providing better accuracy. The Discrete Cosine Transform (DCT) uses only a cosine basis; it has also been applied to time-series mining (Korn, Jagadish, & Faloutsos 1997).

Data adaptive approach modifies the parameters of a transformation depending on the data available. By adding a data-sensitive selection step, almost all nondata-adaptive methods can become data adaptive. For spectral decompositions, it usually consists of selecting a subset of the coefficients. Several inherently data-adaptive representations have also been used. Singular value Decomposition (SVD) has been proposed (Korn, Jagadish, & Faloutsos 1997) and later been enhanced for streams (Ravi Kanth, Agrawal, & Singh 1998). It has recently been adapted to find multiscale patterns in time-series streams (Papadimitriou & Yu 2006). Piecewise Linear Approximation (PLA) is a widely used approach for the segmentation task. The set of polynomial coefficients can be obtained by interpolation (Keogh & Pazzani 1998). Many derivatives of this technique have been introduced. The Landmarks system (Perng *et al.* 2000) extends this notion to include a multiresolution property. However, the extraction of features relies on several parameters that are highly data dependent. (Xie & Yan 2007) proposed a pattern-based representation of time series. The input series is approximated by a set of concave and convex patterns to improve the subsequence matching process.

Instead of producing a numeric output, it is also possible to discretize the data into symbols. This conversion into a symbolic representation also offers the advantage of implicitly performing noise removal by complexity reduction. A relational tree representation is used in (Bakshi & Stephanopoulos 1995). Nonterminal nodes of the tree correspond to valleys and terminal nodes to peaks in the time series. The Symbolic Aggregate approXimation (SAX) (Lin *et al.* 2003), based on the same underlying idea as Piecewise Aggregation Approximation (PAA) (Yi & Faloutsos 2000), calls on equal frequency histograms on

sliding windows to create a sequence of short words. An extension of this approach, called indexable Symbolic Aggregate approXimation (Camerra *et al.* 2010), has been proposed to make fast indexing possible by providing zero overlap at leaf nodes. The grid-based representation (An *et al.* 2003) places a two-dimensional grid over the time series. The final representation is a bit string describing which values were kept and which bins they were in. Another possibility is to discretize the series to a binary string (Ratanamahatana *et al.* 2005). Each bit indicates whether the series is above or below the average. That way, the series can be very efficiently manipulated. A very interesting approach has been proposed in (Ye & Keogh 2009); it is based on primitives called shapelets, that is, subsequences which are maximally representative of a class and thus fully discriminate classes through the use of a dictionary. This approach can be considered as a step towards bridging the gap between time series and shape analysis.

The model-based approach is based on the assumption that the time series observed has been produced by an underlying model. The goal is thus to find parameters of such a model as a representation. Two time series are therefore considered similar if they have been produced by the same set of parameters driving the underlying model. Several parametric temporal models may be considered, including statistical modeling by feature extraction (Nanopoulos, Alcock, & Manolopoulos 2001), Auto-Regressive and Moving Average (ARMA) models (Kalpakis, Gada, & Puttagunta 2001), Markov Chains (MCs) (Sebastiani *et al.* 1999), or HMMs (Panuccio, Bicego, & Murino 2002). MCs are obviously simpler than HMMs so they fit well shorter series but their expressive power is far more limited. The time-series bitmaps introduced in (Kumar *et al.* 2005) can also be considered as a model-based representation for time series, even if it mainly aims at providing a visualization of time series.

2.2 Motivation

Traditional approaches for modeling temporal data include the estimation of parameters from an assumed time series model. The estimated parameters can then be used as features in a classifier to perform classification, or as the basis to make prediction or perform imputation/regression. However, more complex, high-dimensional, and noisy real-world time series data cannot be described with analytical equations with parameters to solve since the dynamics are either too complex or unknown (Taylor 2009) and traditional shallow methods, which contain only a small number of non-linear operations, do not have the capacity to accurately model such complex data.

Recall time-series data consists of sampled data points taken from a continuous, real-valued process over time. There are a number of characteristics of time-series data that make it different from other types of data (Längkvist, Karlsson, & Loutfi 2014).

- The sampled time-series data often contain significant noise and have high dimensionality. To deal with this, signal processing techniques such as dimensionality reduction techniques, wavelet analysis or filtering can be applied to remove some of the noise and reduce the dimensionality. The use of feature extraction has a number of advantages (Nanopoulos, Alcock, & Manolopoulos 2001). However, valuable information could be lost and the choice of features and signal processing techniques may require expertise with the data.

- It is not certain that there is enough information available to understand the process. For example, in electronic nose data, where an array of sensors with different selectivity for a number of gases are combined to identify a particular smell, there is no guarantee that the selection of sensors is actually able to identify the target odour. In financial data when observing a single stock, which only measures a small aspect of

a complex system, there is most likely not enough information to predict the future (Flash & Hochner 2005).

- Time-series have an explicit dependency on the time variable. Given an input $x(t)$ at time t, the model predicts $y(t)$, but an identical input at a later time could be associated with a different prediction. To solve this problem, the model either has to include more data input from the past or must have a memory of past inputs. For long-term dependencies the first approach could make the input size too large for the model to handle. Another challenge is that the length of the time-dependencies could be unknown. Many time-series are also non-stationary, meaning that the characteristics of the data, such as mean, variance, and frequency, changes over time. For some time-series data, the change in frequency is so relevant to the task that it is more beneficial to work in the frequency-domain than in the time-domain.

- There is a difference between time-series data and other types of data when it comes to invariance. In other domains, for example computer vision, it is important to have features that are invariant to translations, rotations, and scale. Most features used for time-series need to be invariant to translations in time.

In conclusion, time-series data is high-dimensional and complex with unique properties that make them challenging to analyze and model. There is large interest in representing the time-series data in order to reduce the dimensionality and extract relevant information. The key for any successful application lies in choosing the right representation. Various time-series problems contain different degrees of the properties discussed in this section and prior knowledge or assumptions about these properties is often infused in the chosen model or feature representation.

To better model complex real-world data, one approach is to develop hand-crafted features that capture the relevant information. This might be task specific and requires

expertise with the data. However, such hand-crafted features can achieve good performance in specific tasks. The alternative is to use unsupervised feature learning in order to learn a layer of feature representations from unlabeled data. This has the advantage that the unlabeled data, which is plentiful and easy to obtain, is utilized and that the features are learned from the data instead of being hand-crafted. These layers of feature representations can be stacked to create deep networks, which are more capable of modeling complex structures in the data. We argue that the hybrid of hand-crafted with unsupervised learning features has the potential to learn more powerful representation and models. However, much focus in the feature learning community has been on developing models for static data and not so much on time-series data.

In another hand, when we model time series using deep learning or recurrent networks based approaches, we found that training is hard. Neural network model always leads to a non-convex optimization problem. The optimization algorithm impacts the quality of the local minimum because it is hard to find a global minimum or estimate how far a particular local minimum is from the best possible solution. The most standard approach to DNNs optimization is Stochastic Gradient Descent (SGD). There are many variants of SGD, researchers and practitioners typically choose a particular variant empirically. Nearly all DNNs optimization algorithms in popular use are gradient-based, recent work has shown that more advanced quasi-Newton methods such as L-BFGS, Hessian-free (HF) and Kronecker-factored Approximate Curvature (K-FAC) approach can yield better results for DNNs tasks (Ngiam *et al.* 2011; Martens 2010; Martens & Grosse 2015). Although the high computational complexity using second order derivatives can be addressed by hardware extensions (GPU or clusters) or batch methods when dealing with massive data, SGD still provides a robust default choice for optimizing DNNs.

Instead of fortifying the training approach for DNNs, we focused on designing the

error function to convex the error space. The convexification approach has been studied in the optimization community for decades, but never be seriously applied within deep learning. Two well-known methods are the graduated nonconvexity method (Blake & Zisserman 1987) and the LiuFloudas convexification method (Liu & Floudas 1993). LiuFloudas convexification method can be applied to optimization problem where the error criterion is twice continuously differentiable, although determining the weight α of the added quadratic function for convexing the error criterion involves much computation when dealing with massive data and parameters.

Chapter 3

SYMBOLIC APPROXIMATION FOR TIME SERIES CLASSIFICATION

3.1 Introduction

As one of the successful techniques to discretize and reformulate raw time-series data, symbolic time series analysis has been used in many different application areas to identify temporal patterns (Daw, Finney, & Tracy 2003). Aligned Cluster Analysis (ACA) was introduced as an unsupervised approach to cluster the temporal patterns in human motion data (Zhou, Torre, & Hodgins 2008). It is an extension of kernel k-means clustering but requires significant computational capacity. Persist is an unsupervised discretization method to maximize the persistence measurement of each symbol (Mörchen & Ultsch 2006). The Piecewise Aggregation Approximation (PAA) method was proposed by Keogh (Keogh *et al.* 2001), which then upgrades to Symbolic Aggregation approXimation (SAX)(Lin *et al.* 2003). In SAX, each aggregation value after the PAA process is mapped into equiprobable intervals according to a standard normal distribution to produce symbolic representations. SAX has become a common representation method due to its simplicity and effectiveness on various types of data mining tasks (Camerra *et al.* 2010).

SAX forms the PAA in temporal order using a sliding window. Such representation is

effective in several data mining tasks such as indexing (Camerra *et al.* 2010) and visualization (Kumar *et al.* 2005). As one of the effective features for classification, a bag-of-words makes use of SAX words to encode non-linearity and benefits from invariance to rotations (Lin, Khade, & Li 2012). Lin *et al*. also reported state-of-the-art results using a One-Nearest-Neighbor classifier (1NN) on University of California Riverside (UCR) Time Series Classification/Clustering databases (Keogh *et al.* 2006). Oates *et al*. applied SAX and Bag-of-Patterns (BoP) to predict outcomes for traumatic brain injury patients (Oates *et al.* 2012a) and explored representation diversity by ensemble voting to further improve classification performance (Oates *et al.* 2012b).

The principal idea of SAX is to smooth the input time series using Piecewise Aggregation approXimation (PAA) and assign symbols to the PAA bins. The overall time series trend will be extracted as a symbolic sequence.

The algorithm requires three parameters: window length w, number of symbols s and alphabet size a. Different parameters lead to different representations of the time series. Given a normalized time series of length L, we first reduce the dimensionality by dividing it into $[L/w]$ non-overlapping sliding windows with skip size 1. Each sliding window is partitioned into s subwindows and mean values are computed to reduce volume and smooth the noise. Then PAA values are mapped to a probability density function $\mathcal{N}(0, 1)$, which is divided into several equiprobable segments. Letters starting from A to Z are assigned to each PAA value according to their corresponding segments (Figure. 3.1).

After discretization and symbolization, BoP is built by a sliding window of length w and converting each subsequence into a SAX word. The BoP representation can catch the features shared in the same structure among different instances regardless of where they occur. We build our features based on BoP histogram of word counts that is analogous to the bag-of-words (Wang *et al.* 2013; Lin, Khade, & Li 2012; Baydogan, Runger, & Tuv 2013).

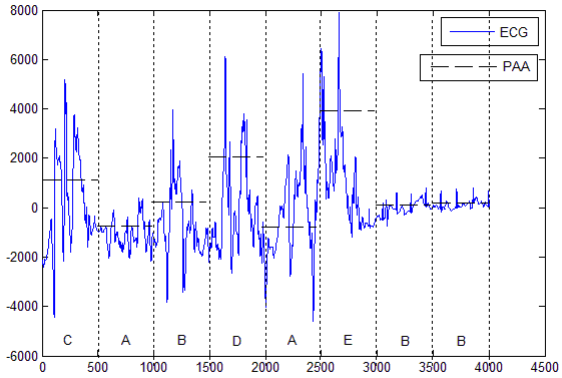

FIG. 3.1. PAA and SAX word for the ECG data. The time series of length 4000 are partitioned into 8 segments. In each segment we compute means to map them to the equiprobable interval. After discretization by PAA and symbolization by SAX, we convert the time series into SAX word sequence $CABDAEBB$.

This chapter will explore two extensions of SAX methods to further improve this symbolic representations by considering the internal temporal dependency in SAX words and extend the SAX approach to the multivariate time series classification problem.

3.2 Time Warping SAX-BoP Representation for Time Series Classification

3.2.1 Background

While standard SAX with BoP obtains state-of-the-art performance on benchmark datasets and real world applications, we hypothesize that we can improve SAX by capturing more temporal information in the BoP representation. Our work is inspired by two observations. First, correlation is common in time series. Statistical time series analysis utilizes AutoCorrelation Functions (*ACF*) and Partial AutoCorrelation Functions (*PACF*) to interpret the internal linear correlations in ARIMA models (Box, Jenkins, & Reinsel 2013). We propose to explicitly extract the implicit linear correlation in time series and thus to help reveal the intrinsic statistical properties to potentially improve the expressive

capacity.

Another motivation is based on the observation that BoP extracts local patterns, regardless of where they occur within time series. Standard SAX keeps the temporal ordering information while this information might be ignored after building a BoP. We attempt to seal the temporal correlation in the bags by borrowing the concept of time embedding vector from dynamic systems to overcome the loss of information.

Among a number of papers that explore the application of SAX and its derivatives, no work has been investigated to integrate BoP patterns and the SAX words with linear temporal correlations. This time warping approach to build SAX words is based on a few research works about symbolic dynamics of time series. The research on permutation entropy (Bandt & Pompe 2002; Riedl, Müller, & Wessel 2013) and fractal dimensions (Clark 1990) include details of delay-time embedding vectors. Oates *et al*. apply SAX with a bag-of-words to detect traumatic brain injury (Oates *et al.* 2012a) and explore the representation diversity via ensembles of different BoP representations (Oates *et al.* 2012b) and provides us the paradigm to analyze and apply our approaches on physiological data.

3.2.2 Motivation

The first motivation relies on the internal linear correlations embedded in time-series. For a stationary process $Z = Z_1, Z_2, ..., Z_t$, the covariance between Z_t and Z_{t+k} is defined as:

$$\gamma_k = cov(Z_t, Z_{t+k}) = E[(Z_t - \mu)(Z_{t+k} - \mu)] \tag{3.1}$$

The correlation between Z_t and Z_{t+k} is:

$$\rho_k = \frac{cov(Z_t, Z_{t+k})}{\sqrt{var(Z_t)}\sqrt{var(Z_{t+k})}} \tag{3.2}$$

As a function of k, ρ_k is called the AutoCorrelation Function (*ACF*). It represents the correlation between Z_t and Z_{t+k} at time lag k.

If we remove the mutual linear dependency on the intervening variables $Z_{t+1}, Z_{t+2}, ..., Z_{t+k-1}$, the conditional correlation is given by:

$$corr(Z_t, Z_{t+k}|Z_{t+1}, Z_{t+2}, ..., Z_{t+k-1}) \tag{3.3}$$

The resulting value is called the Partial AutoCorrelation Function (*PACF*) in time series analysis.

ACF and *PACF* helps discover the internal correlation and identify the order of ARIMA models in statistical time series analysis (Box, Jenkins, & Reinsel 2013). Time series have intrinsic correlations even though sometimes we observe no obvious periodic trends. As shown in Figure 3.2, although there is no obvious periodic trend in the raw data, the *ACF* and *PACF* values show significant linear dependencies at different time lags. In this example, the raw data needs a first order or even higher order differencing to reveal the essential linear temporal dependency in a stationary time series.

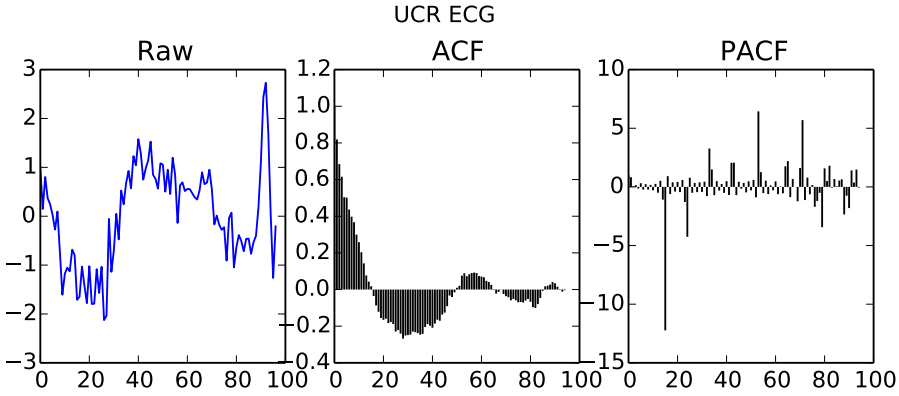

FIG. 3.2. Plots of (a) raw data (left), (b) *ACF* (middle) and (c) *PACF* (right) on ECG dataset from UCR Time Series Classification/Clustering database.

Correlations revealed by *ACF* and *PACF* are more general than periodicity and are very commonly observed in time series data. A number of *ACF* and *PACF* plots from UCR time series databases with various type of data show similar phenomena as seen in Figure 3.2.

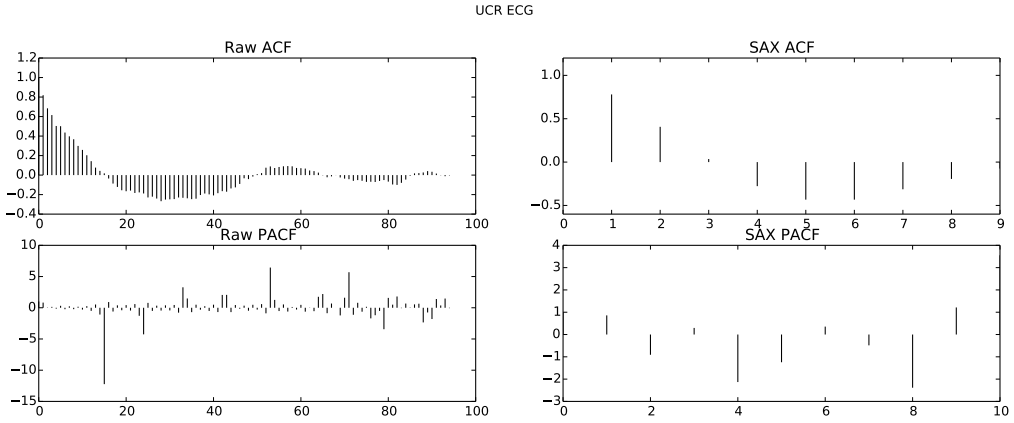

FIG. 3.3. Plots of (a) raw data *ACF* (upper left), (b) SAX word *ACF* (upper right) and (c) raw data *PACF* (bottom left) and (d) SAX word *PACF* (bottom right) on ECG datasets from UCR Time Series Classification/Clustering database. Standard SAX preserves the temporal correlation, but this advantage might be ignored in BoP representations.

In Figure 3.3, standard SAX word shows their intrinsic property of preserving the major internal correlation embedded in raw data. This is because standard SAX builds PAA bins and slides windows sequentially in temporal order. Sequential information in temporal order is one of the essential properties of time series, and standard SAX works quite well on tasks like similarity evaluation and time series indexing. However, this advantage might be lost in the BoP representation because of the order invariance in BoPs.

A Bag-of-Patterns (BoP) is a histogram-based representation for time series data, similar to the bag-of-words approach that is widely accepted by the natural language processing and information retrieval communities. Given a time series of length L, a BoP representation is constructed by sliding a window of length n ($n << L$) to map each subsequence of

length n into a SAX word. Then the $L-n+1$ subsequences are represented as a histogram of word counts.

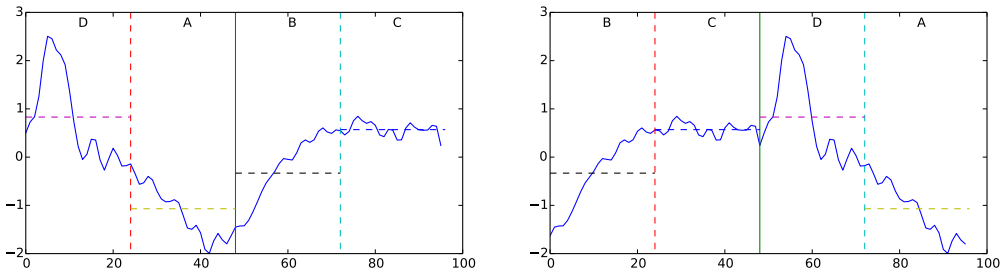

FIG. 3.4. PAA and SAX procedure for ECG data. (a) Time series are partitioned into two windows and each window has two PAA bins (left). We exchange the order of two windows in (b) (right), but the BoP patterns are same, i.e, *BC*: 1, *DA*: 1

BoP can handle time series with varying length. It is invariant to the shift of pattern locations by extracting local structures regardless of where they occur. Nonetheless, the standard SAX approach preserves the temporal order by sliding windows sequentially. BoP runs the risk of losing this temporal dependency due to its shift invariance. To clearly state the problem, consider the toy example as shown in Figure 3.4. We partition the time series into two windows with two PAA bins in each half (Figure 3.4(a)). Then we exchange the order of the two windows as shown in Figure 3.4(b). Obviously, these two time series have different temporal information from each other, but they produce the same BoP.

The BoP reveals the higher level structures, its invariance to shifts also allows rotation-invariant matching in shape datasets. In the next section, we introduce time warping SAX approaches. They capture the temporal correlation information embedded in time series by taking advantage of the invariance to shift and temporal order in BoP to generate more informative representations.

3.2.3 Time Warping SAX

In this section, we introduce three time warping SAX approaches inspired by the idea of time delay embedding vectors (Bandt & Pompe 2002; Riedl, Müller, & Wessel 2013). They take advantage of the order-invariance of BoP to capture temporal correlation information through building words and bags with a time warping procedure.

Given a time series $T = \{t_1, t_2, ..., t_L\}$ of length L, a sliding window of length n and a number of PAA w, the standard SAX method partitions the time series into $\lceil \frac{L}{n} \rceil - 1$ equal-sized sliding windows $t_i, t_{i+1}, ..., t_{i+n-1}$. Each window is then divided into w PAA bins. In time warping SAX, we change the step size $i + 1$ by $i + \tau$ to build the new time warping sequence $t_i, t_{i+\tau}, ..., t_{i+n-1}$. The delay-time embedding vector $t_i, t_{i+\tau(n-1)}, ..., t_{i+\tau(n-1)}$ is used to embed the time series into a higher dimensional space to reconstruct the original state space vector. τ is the delay time and n is the embedding dimension. In the time warping procedure, τ has the same mechanism with the time lag k in *ACF* and *PACF*. The embedding dimension n is equivalent to the sliding window size.

We will introduce two time warping SAX approaches named *Skipword* and *Skipbin*. They correspond to the different discretization granularity in standard SAX. Moreover, we explore the effect of a skip step on sliding window in standard SAX and call it the *Skipwindow* SAX approach.

Skipword SAX applies the time warping approach to build PAA bins. Because each corresponding SAX word is the mean value of PAA bins, the idea of *Skipword* is to seal the temporal correlations into single SAX word through delay-time embedding vectors.

Consider a simple sequence $T = \{1, 2, ..., 10\}$ with delay time $\tau = 2$, embedding dimension $n = 6$ and number of bins $w = 3$. T is divided into three windows with three PAA bins in each window as $\{1\ 3 \mid 2\ 4 \mid 3\ 5\}$, $\{6\ 8 \mid 7\ 9 \mid 8\ 10\}$. After calculating the mean values in the PAA bins, T is discretized as $\{2\ 3\ 4\}$, $\{7\ 8\ 9\}$. For simplicity, we skip

the Gaussian mapping procedure in the following examples to map the rounding of each PAA mean to its corresponding SAX word with the dictionary {1: A, 2: B, ..., 26: Z}. Thus, we get the BoP pattern *BCD*: 1, *GHI*: 1. Then the window slides forward to generate more BoP patterns with the same loop. The over-simplification here is only to facilitate the explanation of our approaches. In our experiments, we process the data with the full pipeline of standard SAX including the Gaussian mapping.

Skipbin SAX applies time warping approaches to build the sliding window instead of PAA bins. Each window is cut into PAA bins following the standard SAX procedure. By enlarging the granularity of temporal correlations, we expect different embedding information to be sealed in BoP representations.

Consider the same sequence $T = \{1, 2, ..., 10\}$ with the delay time $\tau = 2$, embedding dimension $n = 5$ and the number of bins $w = 3$. We apply time warping on subsequence to build two windows and partition each window into three PAA bins {1 3 | 5 7 | 9}, {2 4 | 6 8 | 10}. After figuring out the mean values in each PAA bins, T is discretized into {2 6 9} and {3 7 10} with the BoP patterns *BFI*: 1, *CGJ*: 1. Then The window slides forward to generate more BoP patterns in the same way.

The *Skipwindow* SAX approach is the same as standard SAX. The only difference is the changing time steps among each sliding window. For the sequence $T = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$, we set time step $t = 5$, window length $n = 5$ and words number $w = 3$. The *Skipwindow* SAX discretizes the sequence into {1 2 | 3 4 | 5}, {6 7 | 8 9 | 10}. No overlaps occurs in this example due to the large skip size ($t = 5$). The final BoP patterns are *BDE*: 1, *GIJ*: 1.

By using a time warping procedure based on delay-time embedding vectors, time warping SAX approaches seal the correlation information into different sizes of 'bags', i.e., PAA bins and sliding windows. *Skipword* has smaller granularity because each SAX word will contain temporal correlation with different delay embedding vector parameters.

Instead of embedding the correlation into each single SAX word, the *Skipbin* approach looks at larger temporal scales to encode the correlation into the windows that contain several PAA bins. *Skipbin* SAX words have longer memory than *Skipword* SAX words because the temporal order is preserved in sliding windows with larger scale than a single PAA bin. *Skipwindow* SAX is a straight-forward extension of the standard SAX approach with a larger parameter space as the time step t will change instead of fixing it at 1. We assume that *Skipwindow* also captures some temporal dependency when building the BoP representations. However, when the skip size increases, the BoP discards more phases of the original time series and results in more information loss.

Delay-time embedding is a powerful tool. In principle, almost any delay time τ and embedding dimension n works when we need to analyze the correlation behavior of a complex dynamical system if we have unlimited precise data. However, choosing the embedding vector $\tau(n-1)$ is not trivial. We want both τ and and $\tau(n-1)$ to be close to some characteristic decorrelation time. One suggested principle is exactly the *ACF* (Clark 1990).

Note that if we tune the parameters of the three time warping SAX approaches, the standard SAX has a chance to be replicated. That is, standard SAX is a specific subset of time warping SAX. While we take temporal correlation into consideration, we also add one more dimension, the time delay τ in the parameter space. To avoid 'cheating' and reveal the real impact of the internal correlation, we get rid of the parameter intervals that will reduce time warping SAX to the standard SAX.

3.2.4 Experiments and Results of Classification on Univariate Time Series

The benchmark datasets are from the UCR Time Series Classification and Clustering home page (Keogh *et al.* 2006). We choose the subsets for which the BoP error rate with standard SAX is above 0.1. The error rate is the fraction of incorrectly classified instances. For each dataset, Table. 3.1 gives its name, the number of classes and the length of the

individual time series. The datasets are pre-split into training and testing sets to facilitate experimental comparisons. We also give the number of training and test instances. The test error rate of the standard SAX-BoP approach is directly taken from (Oates *et al.* 2012b) (Table. 3.1).

Table 3.1. BENCHMARK TIME SERIES DATASETS AND SUMMARY STATISTICS

	Class	Train	Test	Length	Error Rate (Oates *et al.* 2012b)
Coffee	2	28	28	286	0.1071
Oliveoil	4	30	30	570	0.1667
ECG200	2	100	100	96	0.22
Lighting2	2	60	61	637	0.2295
Lighting7	7	70	73	319	0.3973
Beef	5	30	30	470	0.4667
Adiac	37	390	391	176	0.3836
50words	50	450	455	270	0.4396
FaceAll	14	560	1690	131	0.2497
OSULeaf	6	200	242	427	0.3058
SwedishLeaf	15	500	625	128	0.2064
yoga	2	300	3000	426	0.1677

The second dataset is the patient vital signs signals (ECG and PPG) collected from University of Maryland School of Medicine. All patient data are anonymous in order to protect patient privacy. 556 patient's ECG and PPG data were collected in 68 to 128 minutes at a 240 Hz sampling rate. Among them, 237 patient's vital signs data are less than 128 minutes long and all the data is quite noisy. The label demonstrates if the patient needed a blood transfusion for Packed Red Blood Cell (pRBC) within 6 hours of admission. The vital signs time series are preprocessed by filtering outliers and integrating the means in each minute to reduce data size. Because the data is highly skewed with only 17 positive instance, we down-sampled 17 negative instance to rebuild a balanced dataset.

We construct BoPs for the datasets in Table 3.1 by looping over the hyperparameters on the training set. Given a time series of length m, we set $n \in \{0.15m, 0.16m, ..., 0.36m\}$

,$w \in \{2, 4, 6, 8\}$ and $a \in \{3, 4, ..., 10\}$. Moreover, the delay-time τ and skip time t loops in $\{0.01m, 0.03m, ..., 0.15m\}$. We classify the time series with each BoP on the training set to select the optimal parameters with Leave-One-Out Cross Validation (LOOCV) for the test set. If two or more representations tie, we choose the representation with the smallest possible vocabulary size a^w and delay-time τ. For vital signs data, we apply LOOCV to evaluate the classification performance on the training set alone.

Table 3.2. 1NN error rate of time warping SAX and standard SAX on test datasets

Dataset	SAX	*Skipword*	*Skipbin*	*Skipwindow*
Coffee	0.1071	**0.0357**	**0.0357**	**0.0357**
Oliveoil	0.1667	**0.1**	**0.1**	**0.1**
ECG200	0.22	**0.12**	**0.12**	**0.12**
Lighting2	0.2295	0.2295	**0.1475**	**0.1475**
Lighting7	0.3973	0.2603	**0.2466**	0.3288
Beef	0.4667	0.4333	**0.4**	0.4667
Adiac	**0.3836**	0.5166	0.5141	0.5166
50words	0.4396	0.3867	**0.3756**	0.3978
FaceAll	0.2497	0.2438	0.2438	**0.2349**
OSULeaf	**0.3058**	0.3430	0.3347	0.3554
SwedishLeaf	**0.2064**	0.2400	0.2336	0.2496
yoga	0.1677	0.1670	**0.1603**	0.2017

Table 3.2 shows the LOOCV error rate on test sets. Except for "Adiac", "OSULeaf" and "SwedishLeaf", *Skipbin* and *Skipword* SAX methods outperform standard SAX. *Skipwindow* SAX also demonstrates specific improvements, although it is always worse than the *Skipbin* and *Skipword* approaches. Because *Skipwindow* is the direct extension of standard SAX with a larger parameter space, the improvement seems "obvious" as it searches four parameters as opposed to three. However, *Skipwindow* will discard more original data when time step grow large (particularly when $t > w$) and leads to significant information loss.

Among these three time warping SAX approaches, *Skipbin* shows better expressive

capacity than the other two based on the classification performance. This is most likely because *Skipbin* SAX applies the time warping procedure to build windows that contain several PAA bins. It takes advantage of delay-time embedding vectors in capturing the temporal correlation in the larger window but also preserves sequential order of the subsequence within each PAA bin. With appropriate combinations of hyperparameters, *Skipbin* extracts the temporal dependency as well as the sequential information to learn more powerful BoP representations.

FIG. 3.5. Illustration of *Skipword* (left) and *Skipbin* (right) SAX. *Skipword* extracts the temporal correlation through using delay-time embedding vector in each single word; *Skipbin* captures the linear correlation in each word and preserves the sequential order in the PAA bins.

It is necessary to explore the representation diversity between time warping SAX and standard SAX approaches. Different datasets will require different parameters to properly demonstrate the expressive capacity and maximize the classification accuracy (Oates *et al.* 2012b). The different time warping procedures in time warping SAX approaches result in the diverse BoP patterns as illustrated in Figure 3.5.

A time warping SAX representation is determined by the window size n, the number of words w, the alphabet size a and the delay-time/skip step τ/t. Because various datasets and different time warping approaches need specific combinations of hyperparameters to learn appropriate BoPs, one would expect to observe significant representation diversity on

Table 3.3. BEST FOUR REPRESENTATIONS $(n, w, a, \tau/t)$ BY ERROR RATE FOR TWO DATASETS.

Dataset	SAX	*Skipword*	*Skipbin*	*Skipwindow*
50words	94, 8, 3 , 1	65, 4, 3, 27	65, 4, 3, 27	95, 4, 3, 3
	97, 4, 3 , 1	95, 4, 3, 19	68, 4, 3, 27	86, 4, 3, 3
	94, 4, 4 , 1	78, 4, 3, 19	86, 4, 3, 3	90, 4, 3, 3
	86, 8, 3 , 1	89, 4, 3, 19	62, 4, 3, 27	97, 4, 3, 3
Lighting7	70, 8, 4 , 1	48, 2, 8, 3	48, 2, 10, 3	48, 2, 10, 3
	70, 8, 3 ,1	51, 4, 5, 3	73, 2, 10, 41	57, 2, 10, 3
	51, 2, 7, 1	90, 4, 7, 3	57, 2, 10, 22	70, 2, 7, 3
	51, 2, 9, 1	54, 2, 6, 3	60, 2, 10, 41	96, 2, 5, 3

different SAX approaches and datasets. We also assume that we can find some patterns in the representation diversities. Table. 3.3 shows four optimal representations with the best training error rate on two datasets. The best representations for the "50Words" have the same word and alphabet sizes $(4, 3)$ on all time warping SAX approaches. The Window length and delay-time change respectively to learn the BoPs with significant temporal correlations. The "Lighting7" dataset shows more obvious diversity in the best representations, but the *Skipbin* SAX approach has the same word and alphabet sizes $(2, 10)$, which means it prefers short words and high alphabetical resolution to build the BoP patterns. On these two datasets, the skip step for *Skipwindow* SAX keeps on the lowest value $(0.01m)$, which is in accordance with our analysis that a large skip step brings much information loss. Thus *Skipwindow* SAX is trying to avoid such information loss using small sliding steps.

Figure. 3.6 shows the plots of training error rate on the "50word" and "Lighting7" datasets with all representations sorted in ascending order over three time warping SAX approaches. Although we observe slight differences among the error rates of the best representations for three time warping SAX approaches in Table. 3.2, the error rate curve over all representations show more details about the difference among time warping approaches.

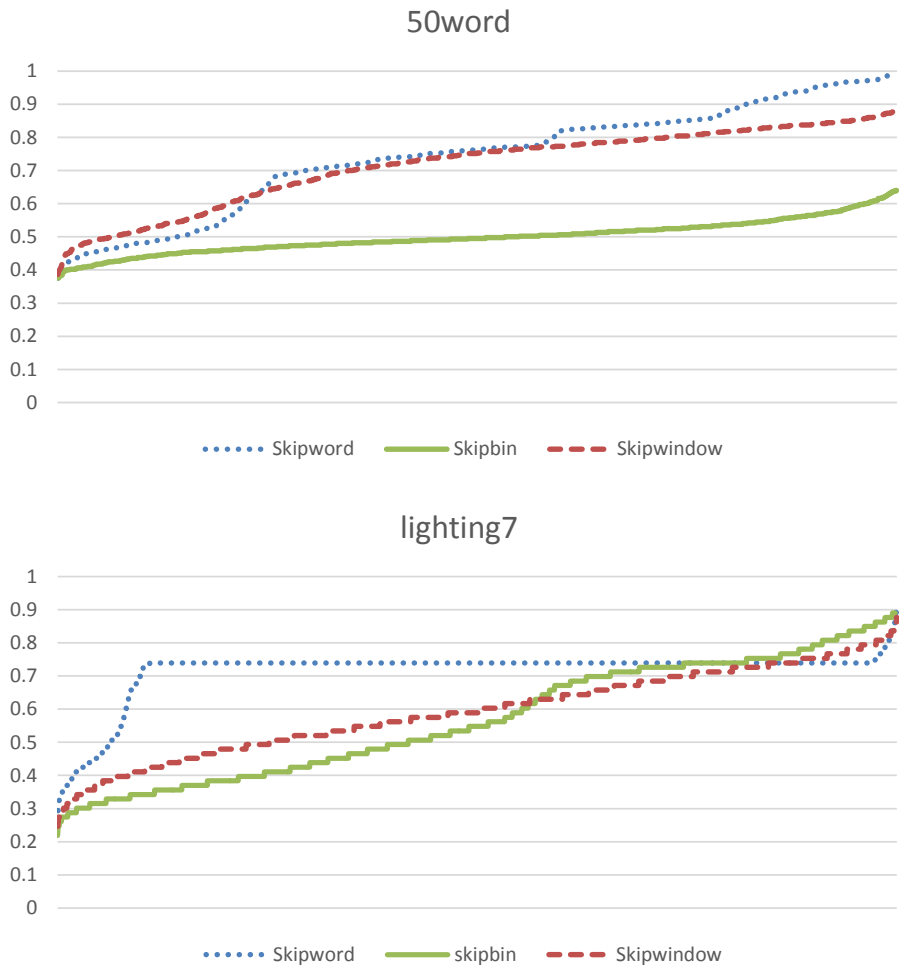

FIG. 3.6. Curve of test error rate for all representations of time warping SAX by ascending rank for the ”50word” and ”Lighting7” datasets. Y axis is the percent error rate.

Three time warping SAX approaches have the similar best performance, but the curve of *Skipbin* SAX stabilizes at the lower horizon with more representations than *Skipword* and *Skipwindow*. It implies that *Skipbin* SAX is more likely to have robust and better classification accuracy. On ”Lighting7”, *Skipword* SAX stays at the best error rate at the very beginning but follows by a sharp slope where the classification accuracy rapidly decreases. Through this sharp rising in error rate, the curve leaps to the stable state where the representations show significantly worse classification performance. This means that on some

Table 3.4. ERROR RATE PREDICTING PATIENT OUTCOMES USING VITAL SIGNS WITH DIFFERENT SAX APPROACHES

	ECG	PPG
Standard SAX	0.2353	0.2059
Skipbin SAX	0.2059	0.1765
Skipword SAX	0.2353	0.1765
Skipwindow SAX	0.1765	0.2353

dataset, *Skipword* achieves good classification performance but has large variance and more significant risk of overfitting. *Skipwindow* SAX always achieve medium performance but with good robustness because the sloping region for both datasets are smooth.

The next experiment explores the utility of time warping SAX in predicting if the patient needs a blood transfusion based on their vital sign data. We use a 1NN classifier and report the best LOOCV error rate in Table. 3.4.

On the ECG dataset, *Skipword* is equivalent to standard SAX while *Skipbin* and *Skipwindow* SAX outperforms standard SAX. On PPG data, *Skipwindow* approach is worse than standard SAX, but the other two approaches both overtake standard SAX with 82.35% classification accuracy. These results on real world physiological data support our analysis that *Skipbin* and *Skipword* SAX are more likely to have better classification performance than standard SAX as they capture the temporal correlation and take advantage of the shift-invariance of BoP. Because the *Skipwindow* approach is a natural extension based on standard SAX, its performance might outperform its prototype if we can find the optimal parameters by mimicking SAX. Time warping SAX has one more parameter (τ or t) which requires more effort to tune the parameters. However, our approaches and results make a strong suggestion to consider the linear or even non-linear temporal correlations when building SAX words.

3.3 Pooling SAX-BoP Approaches with Boosting to Classify Multivariate Time Series

3.3.1 Background and Motivation

This section is motivated by the real-world problems in healthcare. Non-invasive, continuous, high resolution vital signs data, such as Electrocardiography (ECG) and Photoplethysmograph (PPG), are commonly used in hospital settings for better monitoring of patient outcomes to optimize early care. Such mechanism can help doctors to judge patient status more accurately and quickly, thus to get thorough preparation for future treatments (Kononenko 2001). The work in this section is strongly motivated by the real world problem to predict the potential needs of the patient for pRBC (packed Red Blood Cell) in the next few hours from very high resolution vital signs data (ECG and PPG).

We formulate this task as a regular multivariate time series classification problem. Because our data is massive and noisy, the SAX approach tends to be our first choice as the representation for classification. However, multivariate time series data are not only characterized by individual attributes, but also by the relationships between the attributes (Bankó & Abonyi 2012). Such information is not captured by the similarity between the individual sequences (Weng & Shen 2008). To deal with the classification problem on multivariate time series, several similarity measurements including Edit distance with Real Penalty (ERP) and Time Warping Edit Distance (TWED) are summarized and tested on several benchmark datasets (Lin J 2012). Recently, a new symbolic representation for multivariate time series classification (SMTS) is proposed. SMTS builds a tree learner with two ensembles to learn the segmentations and a high-dimensional codebook (Baydogan & Runger 2014).

While the above methods provide new perspectives to handle the issue of multivariate data, some of the them are time consuming (e.g. SMTS) or do not handle the curse of

dimensionality (distance on raw data). Can we design such a method to handle the specific type of multivariate *physiological* time series data? The answer is yes. We note that strong correlation lies among multivariate time series, especially among the synchronous physiological data. Yu *et al*. made progress in automatically estimating the reliability of reference heart rates (HRr) derived from ECG and PPG waveforms recorded by monitors (Yu *et al.* 2006). Lu *et al*. compared 5-minute recordings to demonstrate a very high correlation level in the temporal and frequency domains with the nonlinear dynamic analyses between HRV measures derived from PPG and ECG (Lu *et al.* 2009). They confirmed where Heart Rate Variability (HRV) measures can be accurately derived in healthy subjects. PPG could also provide accurate interpulse intervals as a practical alternative to ECG for HRV analysis. Such strong correlations will greatly simplify the feature fusion procedure for us. Some work has been done to visualize the correlation among multivariate physiological time series data (Ordonez, Oates, & Desjardins 2012).

Another work that inspired our approache is the special pooling structure in convolutional neural networks - one of the most successful deep learning architectures (LeCun *et al.* 1989). In this work, we address the classification problem on synchronous high resolution vital signs data based on the new Pooling SAX with BoP representations (Pooling SAX-BoP) and Boosting algorithms (Freund & Schapire 1995).

3.3.2 Pooling SAX-BoP Approaches

In brief, we propose a post-processing stage to pool the significant SAX word of each variable with different weighting schemes that is analogous to multiple pooling structures for feature extraction in Convolutional Neural Networks (CNNs). Instead of weights trained by the label, we apply the non-parametric weights to determine the information density of each variable and pool out the significant word at each time. Finally, we use the pooled-out sequence as single outputs of the multivariate word sequence to build the BoP

representations. We call such feature extraction and fusion methods the Pooling SAX-BoP approach.

Definition. *Let $\mathcal{X}_k^t$ denotes a subsequence/bin of time series in channel k at time t. Operator $\mathcal{G}$ denotes the process of calculating PAA values in each bin, $\mathcal{F}$ is the function to map PAA values to the corresponding SAX word with respect to the standard normal distribution $\mathcal{N}(0,1)$. $\mathcal{W}_k$ is the nonparametric weights, $\mathcal{S}$ is the pooling output.*

Given a subsequence/bin in a sliding window of multiple time series (MTS) $\mathcal{X}_1^t, \mathcal{X}_2^t, \cdots, \mathcal{X}_k^t$, the pooling functions of four approaches are given below.

- **Max Pooling**

$$\begin{aligned} K_{Max} &= \arg\max_k \mathcal{W}_k \cdot \mathcal{F}(\mathcal{G}(\mathcal{X}_k^t)) \\ \mathcal{S} &= \mathcal{F}(\mathcal{G}(\mathcal{X}_{K_{Max}}^t)) \end{aligned} \tag{3.4}$$

  Considering the toy example, we extract the SAX word, $[a]$ and $[c]$ from a bin in the bivariate time series. Given $\mathcal{W}_1 = 0.8$, $\mathcal{W}_2 = 0.2$. Since $0.8 \times [a] = 0.8 \times 1 > 0.2 \times 3 = 0.2 \times [c]$, so the $\mathcal{S} = [a]$.

- **Min Pooling**

$$\begin{aligned} K_{Min} &= \arg\min_k \mathcal{W}_k \cdot \mathcal{F}(\mathcal{G}(\mathcal{X}_k^t)) \\ \mathcal{S} &= \mathcal{F}(\mathcal{G}(\mathcal{X}_{K_{Min}}^t)) \end{aligned} \tag{3.5}$$

  Considering the same toy example above with the SAX word $[a]$ and $[c]$. Given $\mathcal{W}_1 = 0.8$, $\mathcal{W}_2 = 0.2$, the $\mathcal{S} = [c]$.

- **Max-Min Pooling**

  After extracting two significant SAX words (or, say their variable index K_{Min} and

K_{Max}) from the pooling functions in Equations. (1) and (2),

$$\mathcal{S} = [\mathcal{F}(\mathcal{G}(\mathcal{X}^t_{K_{Max}})), \mathcal{F}(\mathcal{G}(\mathcal{X}^t_{K_{Min}}))] \tag{3.6}$$

Considering the same above example with the maximum pooling word $[a]$ and the minimum pooling word $[c]$, then the $\mathcal{S} = [a, c]$.

- **Max-Min-Mean Pooling (3M Pooling)**

  Given the significant SAX word indexK_{Min} and K_{Max} from the pooling functions in Equations. (1) and (2), we consider the weighted average of the SAX words among different channels as

$$\begin{aligned} \mathcal{S}_{Mean} &= \lceil \sum_{k=1}^{K} \frac{\mathcal{W}_k}{Z} \cdot \mathcal{F}(\mathcal{G}(\mathcal{X}^t_k)) \rceil \\ Z &= \sum_{k=1}^{K} \mathcal{W}_k \end{aligned} \tag{3.7}$$

  Then

$$\mathcal{S} = [\mathcal{F}(\mathcal{G}(\mathcal{X}^t_{K_{Max}})), \mathcal{S}_{Mean}, \mathcal{F}(\mathcal{G}(\mathcal{X}^t_{K_{Min}}))] \tag{3.8}$$

  Given two synchronous SAX words $[a]$ and $[c]$ with the weights $\mathcal{W}_1 = 0.8, \mathcal{W}_2 = 0.2$, from Equation (4), $\mathcal{S}_{Mean} = b$, then $\mathcal{S} = [a, b, c]$

The four pooling approaches above are actually analogous to the pooling architecture in CNNs. Max pooling in CNNs attempts to extract the significant weight vectors with respect to the labels to achieve translation invariance. For multivariate time series, we suppose to pool out the most significant channels with more information density. Max/Min Pooling also provide us the translation invariance cross multiple channels at the same

temporal interval. Max-Min pooling and 3M pooling actually act much like multiple K-pooling. The motivation stems from the significance of the extreme values in time series. 3M pooling combines the average pooling with multiple K-pooling. In 3M pooling, we observe how the weighted average value regulates the behavior of the BoP representations together with max and min values.

The weight scheme $\mathcal{W}_k$ is a series of non-parametric weights. They decide the significance level of each SAX word in different channels at the same temporal interval. We have multiple choices to define $\mathcal{W}_k$. Entropy concerns more about the information density of each channel, average KL-divergence measures the difference between the object variable and other variables. Permutation Entropy (Bandt & Pompe 2002) evaluates the complexity of a given time series. All these measurements are nonparametric but have different computational complexity and focus on different aspects. In this proposal, we mainly explore the physiological time series data, where, strong correlations are always observed among different channels/variables. We care more about the information density in synchronous variables as they tend to have more significant regulations in synchronous data. Our weight scheme is defined as rescaled variance:

$$\mathcal{W}_k = \frac{\sum_{t=1}^{L_k}(\mathcal{X}_k^t - \bar{\mathcal{X}}_k)^2}{L} \tag{3.9}$$

In the above equation, $\mathcal{X}_k^t$ is rescaled into the interval $[0, 1]$. Such rescaled variance actually evaluates the information quantity regardless of the magnitude in each channel. This weight scheme regulates the pooling behavior to extract the significant features while preferring the channels with more information density.

After pooling out the single sequence of SAX word from the multivariate time series, we build the Bag-of-Patterns representation to classify the multivariate time series with a 1NN classifier.

3.3.3 Experiments on Multivariate Time Series

Table 3.5. CV error rates on two standard dataset

	ECG	wafer
Pooling SAX Max	**0.115**	0.0293
Pooling SAX Min	**0.115**	0.0168
Pooling SAX Max-Min	**0.115**	0.0242
Pooling SAX 3M	**0.115**	0.0242
SMTS	0.134	**0.01**
Euclidean	0.1778	0.0833
DTW(full)	0.1889	0.0909
DTW(window)	0.1722	0.0656
EDR	0.2	0.3131
ERP	0.1944	0.0556
LCSS	0.1278	0.1363
LCSS Relaxed	0.1278	0.1091
TWED	0.1278	0.0318

Lin *et al.* compared multiple distance measurements for classification on multivariate time series and use 10 fold cross-validation (CV) to evaluate the performance (Lin J 2012). They chose four multivariate time series datasets. Two of them are too short to use SAX approaches (the average length of "AUSLAN" is 57 and the lengths of "Japanese Vowels" range from 7 to 29). We use the other two datasets and compare the CV error rate with other approaches (including the current state-of-the-art approach SMTS (Baydogan & Runger 2014)) in Table 3.5. Baydogan *et al.* split these two dataset into training and testing sets. They reported the performance on the training-testing data of their SMTS and MTSBF methods. We also train our approach with CV on the training set and compare the performance on the test set (Table 3.6). Our methods are proved to be quite competitive with the SMTS approach while our approaches are simpler (average running time is 3 hours) without any ensemble framework. However, SMTS needs two ensembles to learn a tree-based codebook. Their average running time is reported to be 18 hours.

In the next experiment, we test our approach on the bivariate high resolution vital signs data. The clinical data is collected in the University of Maryland Medical School. All the data is anonymous in order to protect the privacy. The ECG and PPG data of 556 patients were recorded for 68 to 128 minutes with 240 Hz sampling rate. Default values (e.g. -31556 when there is no input) and missing values are allowed. The data is quite massive with more than 1.7 million data points. The label indicates whether a patient needs blood transfusion of pRBC or not in the next 6 hours. The data is highly skewed with only 17 positive samples.

Considering the vital signs data is highly periodical, we preprocess the data second by second. Among 7680 seconds, we get rid of any interval with default or missing values. This may lead to information loss because some seconds only contain few missing number or default values. However, inconsistency caused by missing and default values may be a larger hazard due to the noise and false information. Based on the fact that the normalized time series has approximate Gaussian distribution (Lin *et al.* 2003), we applied a variance filter to further regulate the outliers. According to the *3-Sigma* rule (Pukelsheim 1994), if a value is more than three standard deviations away from the mean, the probability of that point incurring is naturally lower than 0.27%. Thus, we truncate these outliers at the lower and upper bounds. What we need is the overall trend encoded in the time series, the last step of preprocessing is calculating means in each second to reduce the volume and keep

Table 3.6. Test error rates on two standard dataset

	ECG	wafer
SMTS	0.182	0.035
MTSBF	0.165	**0.015**
Pooling SAX Max	**0.16**	0.02
Pooling SAX Min	0.18	0.033
Pooling SAX Max-Min	0.20	0.031
Pooling SAX 3M	0.18	0.039

the overall trend like the PAA strategy.

To guarantee the results of our dataset are not biased, we build 100 new balanced datasets using bootstraping (Manly 2006) by keeping all the 17 positive samples fixed and randomly choosing 17 negative samples with replacement. To compare the performance of the Pooling SAX-BoP approaches on this clinical task, we also test the standard SAX-BoP approach on each single variable of the vital signs data and report the statistics of the best error rate, precision and recall with Leave-One-Out cross validation (LOOCV) on the 100 bootstrap dataset (Table 3.7).

Table 3.7. The LOOCV statistics of the best performance for standard SAX-BoP and multivariate Pooling SAX-BoP on 100 bootstrap dataset

	Error Rate	**Precision**	**Recall**
PPG	$0.192 \pm (0.037)$	$0.813 \pm (0.043)$	$0.808 \pm (0.037)$
ECG	$0.256 \pm (0.038)$	$0.748 \pm (0.037)$	$0.744 \pm (0.038)$
MAX	$0.169 \pm (0.017)$	$0.831 \pm (0.017)$	$0.831 \pm (0.02)$
MIN	$0.196 \pm (0.049)$	$0.807 \pm (0.048)$	$0.804 \pm (0.047)$
MAX-MIN	$0.188 \pm (0.048)$	$0.813 \pm (0.047)$	$0.811 \pm (0.046)$
3M	$0.181 \pm (0.040)$	$0.824 \pm (0.040)$	$0.818 \pm (0.034)$

Figure 3.7 shows the ranked curve of classification error rate, precision and recall on four pooling SAX-BoP approaches. In our experiments, except for the Max Pooling, all other approaches have the equivalent best error rate (0.117) among 100 bootstrap datasets. Although Max Pooling cannot reach to the same performance as the others, it is more stable with slight oscillation in the performance curve. It also demonstrates better average statistics with small standard deviation. Our experiments on the clinical physiological data imply that Pooling SAX-BoP approaches improve the expressive power of the BoP representations and enhance the classification performance. They not only demonstrate the competitive state-of-the-art performance on standard datasets but also work well to solve real world problems.

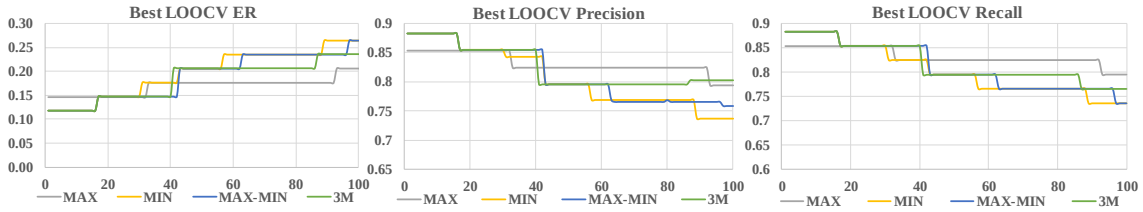

FIG. 3.7. ranked curve of classification error rate, precision and recall on four pooling SAX-BoP approaches. X axis represents the index of ranked bootstrap samplings and Y axis is the error rate.

3.3.4 Ensemble Learning on BoP Representations Using Boosting Algorithm

SAX reduces the volume but also drops details of the raw data. The Pooling SAX methods attempt to extract the significant SAX words while trying to preserve the core information. This means SAX based approaches run the risk of information loss where the key structures in specific time series might be discarded, thus leading to the samples being misclassified. Some work has been proposed to explore ways of exploiting representational diversity for time series classification via ensembles of the representations (Oates *et al.* 2012b). We randomly select a balanced dataset to test majority voting.

In Figure. 3.8 (a), we observe the effect of information loss in the voting results. As the number of voting agents increases from 10 to 70, there is no change in LOOCV error rates for Max Pooilng SAX. The enhancement of performance on other pooling structures is also not so clear. That is, information loss leads to the failure to interpret key features among the top Pooling SAX-BoP representations.

To solve the problem of misclassification caused by the feature missing in Pooling SAX-BoP approaches, we apply a Boosting algorithm to build a non-linear classifier (Freund, Schapire, & others 1996). Boosting adaptively changes the sample weights according to previous classification results to focus on the toughest samples. The missing feature dimension caused by information loss is hit by the larger weight during the iterative pro-

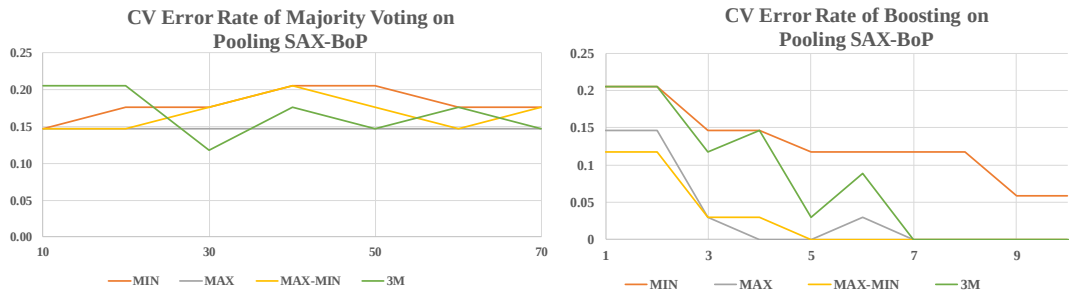

FIG. 3.8. LOOCV Error Rate of a) majority voting (left) and b) Boosting (right) with Pooling SAX-BoP approaches on vital signs data.

cess. Instead of increasing the dimensionality of the feature space in kernel methods, we use Boosting to tune the linear classification hyperplane of several weak classifiers into a nonlinear classification hyperplane by weighted summation. Despite missing dimensions, the nonlinear hyperplane potentially classifies some tough samples in the linear situations.

We apply a slightly modified version of the Boosting algorithm for SAX-BoP representations to classify the balanced vital signs dataset. The trick is to combine each SAX-BoP pattern with a NN classifier as a weak learner in the Boosting algorithm. What Boosting does for the 1NN classifier is to create an ensemble of models with locally modified distance weighting (Athitsos & Sclaroff 2005). After about 10 turns of voting, the converged performance is significantly enhanced (Figure 3.8 (b)).

Recall that we preprocessed high resolution vital signs data by averaging in each 1 second interval. If voting through the diversity of BoP representations enriches the information and enhance the performance, multiple preprocessing frequency may also capture different temporal information of the vital signs data, respectively. In the preprocessing stage, we calculate the mean value in each 1 , 0.5 and 0.3 second interval and combine the Pooling SAX-BoPs of these three preprocessing frequencies together into one large dataset. Because better weak learners will be selected in each iteration from three resampling frequency samples, different resampling rates will mix various temporal diversity into

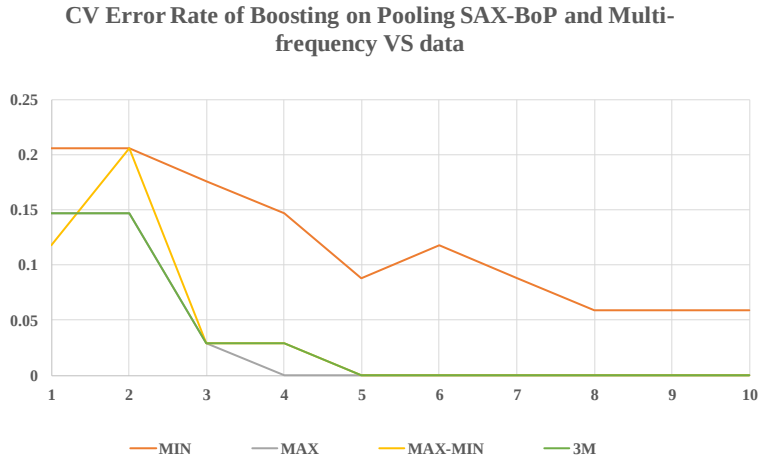

FIG. 3.9. LOOCV Error Rate of Boosting with Pooling SAX-BoP approaches on multiple frequency vital signs data.

Boosting to further enhance the classification performance and accelerate the convergence rate (Figure.3.9).

However, boosting runs the risking of overfitting. The direct application of VC theory shows that boosting can work well if we provide simple weak classifiers which satisfy the weak learning condition if run for enough but not too many rounds (Schapire 2013). We update the error rate from the optimal SAX-BoP representation and 1NNs with the higher accuracy than 50%. The convergence curve in Figure. 3.8 (b) shows that Boosting in our case does not need many iterations (about 10 turns) to converge to its stable performance. We applied the trained Boosting classifier on the multiple frequency balanced dataset to classify the complete dataset with 556 samples. The Boosting classifier achieved a 4.856% error rate, in which the test error rate among the positive samples is 11.76% (2 out of 17 is misclassified). Meanwhile, predicting using the trained Boosting classifier is very fast and convenient for real-time classification of physiological data in a second-by-second manner.

3.4 Conclusion

In this chapter, we extended SAX approach to more complex temporal dynamics and multivariate data. Motivated by the internal correlation embedded in time series and intrinsic property of BoP representations, we proposed time warping SAX to integrate the temporal correlation when building SAX words and BoP representations.Time warping SAX approaches lead to more accurate predictions of patient outcomes based on high resolution vital signs data when the temporal pattern is more complex and primary among the data. Pooling SAX-BoP with Boosting approach is proposed to solve classification problems on the multivariate vital signs time series. The experiments on two standard datasets and the real world vital signs demonstrate the effectiveness and efficiency of pooling SAX-BoP compared with the current state-of-the-art approaches. Instead of majority voting, the Boosting algorithm is applied to significantly improve the performance.

Chapter 4

IMAGING AND MODELING TEMPORAL DATA

4.1 Introduction

Symbolic approximation based approach is simple and effective for modeling temporal data and classification. However, the performance is not so bleeding-edge compared to a bunch of more advanced techniques. The possible reasons lay on several aspects. First, its strong assumption of univariate Gaussian distribution cannot be satisfied in most real datasets. Simple counting on bag-of-words lack the capability of learning more complex dynamics in temporal data. Although the nearest neighbor classifier fits well with SAX-BoP, it is very simple in some way. All above leads to the degradation on the representational power of complicated non-linear temporal correlations among temporal data.

In this section, we will discuss a more general framework to model temporal data using the techniques from computer vision and deep neural networks. It enables a explicit encoding of the static/dynamic temporal correlations with a much more powerful learning mechanism using deep learning framework for visual recognition tasks. Unsupervised learning and pre-training tricks allow us to fully exploit the benefit of learning using unlabeled data. We also extend this approach to the trajectory dataset to cover a larger range of temporal datasets.

4.2 Background

Inspired by recent successes of deep learning in computer vision, we consider the problem of encoding temporal information spatially as images to allow machines to ”visually” recognize and classify temporal data, especially time series data.

Recognition tasks in speech and audio have been well studied. Researchers have achieved success using combinations of Hidden Markov Models (HMMs) with acoustic models based on Gaussian Mixture Models (GMMs) (Reynolds & Rose 1995; Leggetter & Woodland 1995). An alternative approach is to use deep neural networks to produce posterior probabilities over HMM states. Deep learning has become increasingly popular since the introduction of effective ways to train multiple hidden layers (Hinton, Osindero, & Teh 2006) and has been proposed as a replacement for GMMs to model acoustic data in speech recognition tasks (Mohamed, Dahl, & Hinton 2012). These Deep Neural Network - Hidden Markov Model hybrid systems (DNN-HMM) achieved remarkable performance in a variety of speech recognition tasks (Hinton *et al.* 2012; Deng, Hinton, & Kingsbury 2013; Deng *et al.* 2013). Such success stems from learning distributed representations via deeply layered structure and unsupervised pretraining by stacking single layer Restricted Boltzmann Machines (RBM).

Another deep learning architecture used in computer vision is convolutional neural networks (CNNs) (LeCun *et al.* 1998). CNNs exploit translational invariance within their structures by extracting features through receptive fields (Hubel & Wiesel 1962) and learn with weight sharing. CNNs are the state-of-the-art approach in various image recognition and computer vision tasks (Lawrence *et al.* 1997; Krizhevsky, Sutskever, & Hinton 2012; LeCun, Kavukcuoglu, & Farabet 2010). Since unsupervised pretraining has been shown to improve performance (Erhan *et al.* 2010), sparse coding and Topographic Independent Component Analysis (TICA) are integrated as unsupervised pretraining approaches to learn

more diverse features with complex invariances (Kavukcuoglu *et al.* 2010; Ngiam *et al.* 2010).

CNNs were proposed for speech processing because of their invariance to shifts in time and frequency (LeCun & Bengio 1995). Recently, CNNs have been shown to further improve hybrid model performance by applying convolution and max-pooling in the frequency domain on the TIMIT Acoustic-Phonetic Continuous Speech Corpus recognition task (Abdel-Hamid *et al.* 2012). A heterogeneous pooling approach proved to be beneficial for training acoustic invariance (Deng, Abdel-Hamid, & Yu 2013). Further exploration with limited weight sharing and a weighted softmax pooling layer has been proposed to optimize CNN structures for speech recognition tasks (Abdel-Hamid, Deng, & Yu 2013).

However, except for audio and speech data, relatively little work has explored feature learning in the context of typical time series analysis tasks with current deep learning architectures. (Zheng *et al.* 2014) explores supervised feature learning with CNNs to classify multi-channel time series with two datasets. They extracted subsequences with sliding windows and compared their results to Dynamic Time Warping (DTW) with a 1-Nearest-Neighbor classifier (1NN-DTW). Our motivation is to explore a novel framework to encode time series as images and thus to take advantage of the success of deep learning architectures in computer vision to learn features and identify structure in time series. Unlike speech recognition systems in which acoustic/speech data input is typically represented by concatenating Mel-frequency cepstral coefficients (MFCCs) or perceptual linear predictive coefficient (PLPs) (Hermansky 1990), typical time series data are not likely to benefit from transformations applied to speech or acoustic data.

In this chapter, we propose two types of representations for explicitly encoding the temporal patterns in time series as images. We test our approach on twelve time series datasets produced from 2D shape, physiological surveillance, industry and other domains. Two real spatial-temporal trajectory datasets are also considered for experiments to demon-

strate the performance of our approach. We applied deep Convolutional Neural Networks with a pretraining stage that exploits local orthogonality by Topographic ICA (Ngiam *et al.* 2010) to "visually" inspect and classify time series. We report our classification performance both on GAF and MTF separately, and GAF-MTF which resulted from combining GAF and MTF representations into single image. By comparing our results with the current best hand-crafted representation and classification methods on both time series and trajectory data, we show that our approach in practice achieves competitive performance with the state of the art with only cursory exploration of hyperparameters. In addition to exploring the high level features learned by Tiled CNNs, we provide an in-depth analysis in terms of the duality between time series and images. This helps us to more precisely identify the reasons why our approaches work well.

4.3 Motivation

Learning the (long) temporal correlations that are often embedded in time series remains a major challenge in time series analysis and modeling. Most real-world data has a temporal component, whether it is measurements of natural (weather, sound) or man-made (stock market, robotics) phenomena. Traditional approaches for modeling and representing time-series data fall into three categories. In time series learning problems, non-data adaptive models, such as Discrete Fourier Transformation (DFT) (Agrawal, Faloutsos, & Swami 1993), Discrete Wavelet Transformation (DWT) (Chan, Fu, & Yu 2003), and Discrete Cosine Transformation (DCT) (Korn, Jagadish, & Faloutsos 1997), compute the transformation with an algorithm that is invariant with respect to the data to capture the intrinsic temporal correlation with the different basis functions. Meanwhile, researchers explored in the model-based approaches to model time series, such as Auto-Regressive Moving Average models (ARMA) (Kalpakis, Gada, & Puttagunta 2001) and Hidden Markov Models

(HMMs) (Panuccio, Bicego, & Murino 2002), in which the underlying data is assumed to fit a specific type of model to explicitly function the temporal patterns. The estimated parameters can then be used as features for classification or regression. However, more complex, high-dimensional, and noisy real-world time-series data are often difficult to model because the dynamics are either too complex or unknown. Traditional methods, which contain a small number of non-linear operations, might not have the capacity to accurately model such complex systems.

If implicitly learning the complex temporal correlation is difficult, how about reformulating the data to explicitly or even visually encode the temporal dependency, allowing the algorithms to learn more easily? Actually, reformulating the features of time series as visual clues has raised much attention in computer science and physics. The typical examples in speech recognition tasks are that acoustic/speech data input is typically represented by MFCCs or PLPs to explicitly represent the temporal and frequency information. Recently, researchers are trying to build different network structures from time series for visual inspection or designing distance measures. Recurrence Networks were proposed to analyze the structural properties of time series from complex systems (Donner *et al.* 2010; 2011). They build adjacency matrices from the predefined recurrence functions to interpret the time series as complex networks. Silva et al. extended the recurrence plot paradigm for time series classification using compression distance (Silva *et al.* 2013). Another way to build a weighted adjacency matrix is extracting transition dynamics from the first order Markov matrix (Campanharo *et al.* 2011). Although these maps demonstrate distinct topological properties among different time series, it remains unclear how these topological properties relate to the original time series since they have no exact inverse operations. One of our contributions is to propose a set of off-line algorithm to encode the complex correlations in time series into images for visual inspection and classification. The proposed encoding functions have exact/approximate inverse maps, making such transformations

more interpretable.

4.4 Encoding Time Series to Images

We first introduce our two frameworks for encoding time series as images. The first type of image is a Gramian Angular field (GAF), in which we represent time series in a polar coordinate system instead of the typical Cartesian coordinates. In the Gramian matrix, each element is actually the cosine of the sum/difference of angles. Inspired by previous work on the duality between time series and complex networks (Campanharo *et al.* 2011), the main idea of the second framework, the Markov Transition Field (MTF), is to build the Markov matrix of quantile bins after discretization and encode the dynamic transition probability in a quasi-Gramian matrix.

4.4.1 Gramian Angular Field

Given a time series $X = \{x_1, x_2, ..., x_n\}$ of n real-valued observations, we rescale X so that all values fall in the interval $[-1, 1]$:

$$\tilde{x_i} = \frac{(x_i - max(X) + (x_i - min(X))}{max(X) - min(X)} \tag{4.1}$$

Thus we can represent the rescaled time series $\tilde{X}$ in polar coordinates by encoding the value as the angular cosine and time stamp as the radius with the equation below:

$$\begin{cases} \phi = \arccos(\tilde{x_i}), -1 \leq \tilde{x_i} \leq 1, \tilde{x_i} \in \tilde{X} \\ r = \frac{t_i}{N}, t_i \in \mathbb{N} \end{cases} \tag{4.2}$$

In the equation above, t_i is the time stamp and N is a constant factor to regularize the span of the polar coordinate system. the polar coordinate based representation is a interest-

ing way to understand time series. As time increases, corresponding values warp among different angular points on the spanning circles, like water rippling. The encoding map of equation 4.2 has two important properties. First, it is bijective as $\cos(\phi)$ is monotonic when $\phi \in [0, \pi]$. Given a time series, the proposed map produces one and only one result in the polar coordinate system with a unique inverse function. Second, as opposed to Cartesian coordinates, polar coordinates preserve absolute temporal relations. In Cartesian coordinates, the area is defined by $S_{i,j} = \int_{x(i)}^{x(j)} f(x(t))dx(t)$, we have $S_{i,i+k} = S_{j,j+k}$ if $f(x(t))$ has the same values on $[i, i+k]$ and $[j, j+k]$. However, in polar coordinates, if the area is defined as $S'_{i,j} = \int_{\phi(i)}^{\phi(j)} r[\phi(t)]^2 d(\phi(t))$ respectively, then $S'_{i,i+k} \neq S'_{j,j+k}$. That is, the corresponding area from time stamp i to time stamp j is not only dependent on the time interval $|i-j|$, but also determined by the absolute value of i and j. We will address more details in our future work.

After transforming the rescaled time series into the polar coordinate system, we can easily exploit the angular perspective by considering the trigonometric sum between each point to identify the temporal correlation within different time intervals. The Gramian Angular Summation Field (GASF) is defined as follows:

$$G = \begin{bmatrix} \cos(\phi_1 + \phi_1) & \cdots & \cos(\phi_1 + \phi_n) \\ \cos(\phi_2 + \phi_1) & \cdots & \cos(\phi_2 + \phi_n) \\ \vdots & \ddots & \vdots \\ \cos(\phi_n + \phi_1) & \cdots & \cos(\phi_n + \phi_n) \end{bmatrix} \tag{4.3}$$

$$= \tilde{X}' \cdot \tilde{X} - \sqrt{I - \tilde{X}^2}' \cdot \sqrt{I - \tilde{X}^2} \tag{4.4}$$

I is the unit row vector $[1, 1, ..., 1]$. After transforming to the polar coordinate system, we take time series at each time step as a 1-D metric space. By defining the inner product $< x, y >= x \cdot y - \sqrt{1-x^2} \cdot \sqrt{1-y^2}$, G is a Gramian matrix:

$$\begin{bmatrix} < \tilde{x_1}, \tilde{x_1} > & \cdots & < \tilde{x_1}, \tilde{x_n} > \\ < \tilde{x_2}, \tilde{x_1} > & \cdots & < \tilde{x_2}, \tilde{x_n} > \\ \vdots & \ddots & \vdots \\ < \tilde{x_n}, \tilde{x_1} > & \cdots & < \tilde{x_n}, \tilde{x_n} > \end{bmatrix} \tag{4.5}$$

Whereas, if we define the inner product as $< x, y >= \sqrt{1-x^2} \cdot y - x \cdot \sqrt{1-y^2}$, now we have built another type of GAF image - Gramian Angular Difference Field (GADF). The deal is, instead of summing up the angle at each 2D time positions, we calculate their temporal difference. The GADF matrix is like:

$$G = \begin{bmatrix} \sin(\phi_1 - \phi_1) & \cdots & \sin(\phi_1 - \phi_n) \\ \sin(\phi_2 - \phi_1) & \cdots & \sin(\phi_2 - \phi_n) \\ \vdots & \ddots & \vdots \\ \sin(\phi_n - \phi_1) & \cdots & \sin(\phi_n - \phi_n) \end{bmatrix} \tag{4.6}$$

$$= \sqrt{I - \tilde{X}^2}' \cdot \tilde{X} - \tilde{X}' \cdot \sqrt{I - \tilde{X}^2} \tag{4.7}$$

The GAF has several advantages. First, it provides a way to preserve the temporal dependency, since time increases as the position moves from top-left to bottom-right. The GAF contains temporal correlations because $G_{(i,j||i-j|=k)}$ represents the relative correlation (temporal summation or difference) by superposition/differencing of directions with respect to time interval k. The main diagonal $G_{i,i}$ is the special case when $k = 0$, which contains the original value/angular information. With the main diagonal, we will approximately reconstruct the time series from the high level features learned by the deep neural network. However, the GAF is large because the size of the Gramian matrix is $n \times n$ when

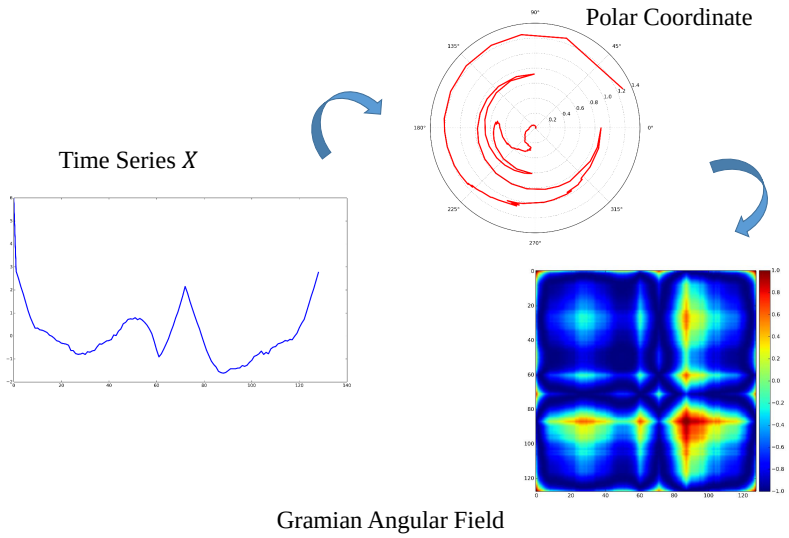

FIG. 4.1. Illustration of the proposed encoding map of Gramian Angular Field. X is a sequence of typical time series in dataset 'SwedishLeaf'. After X is rescaled by eq. (4.1) and smoothed by PAA optionally, we transform it into polar coordinate system by eq. (4.2) and finally calculate its GASF image with eq. (4.4). In this example, we build GASF without PAA smoothing, so the GAF has a high resolution of 128×128.

the length of the raw time series is n. To reduce the size of the GAF, we apply Piecewise Aggregation Approximation (Keogh & Pazzani 2000) to smooth the time series while keeping trends. The full procedure for generating the GAF is illustrated in Figure 4.1.

4.4.2 Markov Transition Field

We propose another framework that is similar to (Campanharo *et al.* 2011) for encoding dynamical transition statistics, but we extend that idea by representing the Markov transition probabilities sequentially to preserve information in the time domain.

Given a time series X, we identify its Q quantile bins and assign each x_i to the corresponding bins q_j ($j \in [1, Q]$). Thus we construct a $Q \times Q$ weighted adjacency matrix W by counting transitions among quantile bins in the manner of a first-order Markov chain along

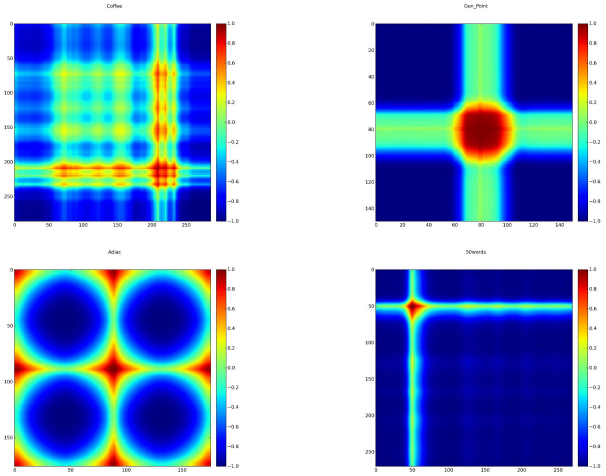

FIG. 4.2. Examples of GAF images on the 'Coffee', 'Gun-Point', 'Adiac' and '50Words' datasets.

the time axis. $w_{i,j}$ is given by the frequency with which a point in the quantile q_j is followed by a point in the quantile q_i. After normalization by $\sum_j w_{ij} = 1$, W is the Markov transition matrix. It is insensitive to the distribution of X and temporal dependency on time steps t_i. However, getting rid of the temporal dependency results in too much information loss in matrix W. To overcome this drawback, we define the Markov Transition Field (MTF) as follows:

$$M = \begin{bmatrix} w_{ij|x_1 \in q_i, x_1 \in q_j} & \cdots & w_{ij|x_1 \in q_i, x_n \in q_j} \\ w_{ij|x_2 \in q_i, x_1 \in q_j} & \cdots & w_{ij|x_2 \in q_i, x_n \in q_j} \\ \vdots & \ddots & \vdots \\ w_{ij|x_n \in q_i, x_1 \in q_j} & \cdots & w_{ij|x_n \in q_i, x_n \in q_j} \end{bmatrix} \tag{4.8}$$

We build a $Q \times Q$ Markov transition matrix (say, W) by dividing the data (magnitude) into Q quantile bins. The quantile bins that contain the data at time stamp i and j (temporal axis) are q_i and q_j ($q \in [1, Q]$). M_{ij} in MTF denotes the transition probability of $q_i \to q_j$. That is, we spread out matrix W which contains the transition probability on magnitude

axis into the MTF matrix by considering the temporal positions.

By assigning the probability from the quantile at time step i to the quantile at time step j at each pixel M_{ij}, the MTF M actually encodes the multi-span transition probabilities of the time series. $M_{i,j||i-j|=k}$ denotes the transition probability between the points with time interval k. For example, $M_{ij|j-i=1}$ illustrates the transition process along the time axis with a skip step. The main diagonal M_{ii}, which is a special case when $k = 0$ captures the probability from each quantile to itself (the self-transition probability) at time step i. To make the image size manageable and computation more efficient, we reduce the MTF size by averaging the pixels in each non-overlapping $m \times m$ patch with the blurring kernel $\{\frac{1}{m^2}\}_{m\times m}$. That is, we aggregate the transition probabilities in each subsequence of length m together. Figure 4.3 shows the procedure to encode time series to MTF.

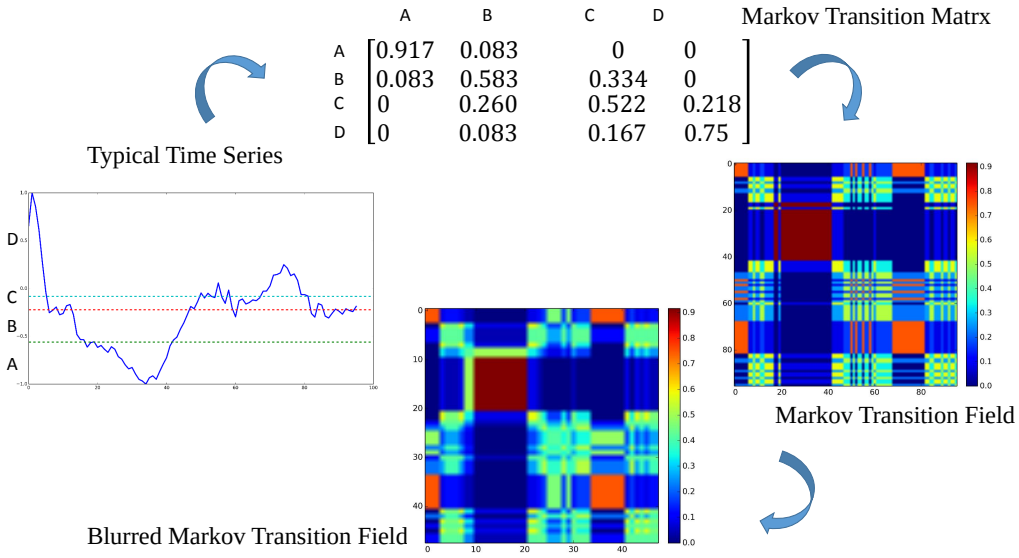

FIG. 4.3. Illustration of the proposed encoding map of Markov Transition Field. X is a sequence of typical time series in dataset 'ECG'. X is firstly discretized into Q quantile bins. Then we calculate its Markov Transition Matrix W and finally build its MTF with eq. (4.8). In addition, we reduce the image size from 96×96 to 48×48 by averaging the pixels in each non-overlapping 2×2 patch.

More full MTF images are shown in Figure 4.4.

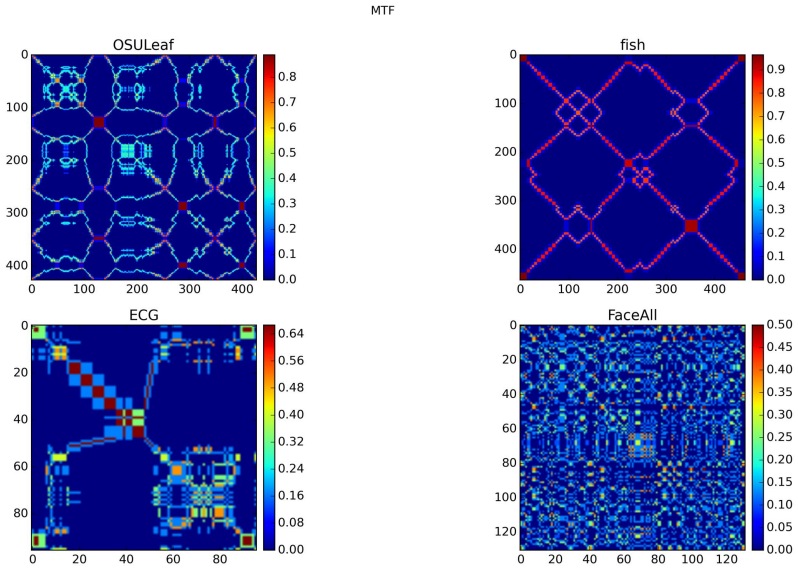

FIG. 4.4. Example of full MTF on 4 dataset (without blurring) with 16 quantile bins.

4.5 Classifying Time Series Using Deep Learning Architectures

In this section, we will explore the classification performance on time series images through using different deep learning (DL) architectures which include Convolutional Neural Networks (CNN), Tiled Convolutional Neural Networks (TCNN), Stacked Autoencoders (SAE) and Deep Belief Nets (DBN). It is also interesting to compare the performance and learned features of these DL methods on our non-natural images.

4.5.1 Tiled Convolutional Neural Networks

Tiled Convolutional Neural Networks (Ngiam *et al.* 2010) are a variation of Convolutional Neural Networks. They use tiles and multiple feature maps to learn invariant features. Tiles are parameterized by a tile size K to control the distance over which weights

are shared. By producing multiple feature maps, Tiled CNNs learn overcomplete representations through unsupervised pretraining with Topographic ICA (TICA).

A typical TICA network is actually a double-stage optimization procedure with square and square root nonlinearities in each stage, respectively. In the first stage, the weight matrix W is learned while the matrix V is hard-coded to represent the topographic structure of units. More precisely, given a sequence of inputs $\{x^h\}$, the activation of each unit in the second stage is $f_i(x^{(h)}; W, V) = \sqrt{\sum_{k=1}^{p} V_{ik} (\sum_{j=1}^{q} W_{kj} x_j^{(h)})^2}$. TICA learns the weight matrix W in the second stage by solving:

$$\begin{aligned} &\underset{W}{\text{minimize}} \quad \sum_{h=1}^{n} \sum_{i=1}^{p} f_i(x^{(h)}; W, V) \\ &\text{subject to} \quad WW^T = I \end{aligned} \tag{4.9}$$

$W \in \mathbb{R}^{p \times q}$ and $V \in \mathbb{R}^{p \times p}$ where p is the number of hidden units in a layer and q is the size of the input. V is a logical matrix ($V_{ij} = 1$ or 0) that encodes the topographic structure of the hidden units by a contiguous 3×3 block. The orthogonality constraint $WW^T = I$ provides diversity among learned features.

The pretraining algorithm (Algorithm. 1) is based on gradient descent on the TICA objective function in Equation. 4.9. The inner loop is a simple implementation of backtracking linesearch. The $orthogonalize\_localRF(W^new)$ function only orthogonalizes the weights that have completely overlapping receptive fields. Weight-tying is applied by averaging each set of tied weights. The algorithm is trained by batch projected gradient descent. Other unsupervised feature learning algorithms such as RBMs and autoencoders (Bengio *et al.* 2007) require more parameter tuning, especially during optimization. However, pretraining with TICA usually requires little tuning of optimization parameters, because the tractable objective function of TICA allows to monitor convergence easily.

Algorithm 1 Unsupervised pretraining with TICA.

Require: $\{x^{(t)}\}_{t=1}^{T}, v, s, W, V$ as input
Ensure: W as output
  repeat
    $f^{old} = \sum_{t=1}^{T}\sum_{i=1}^{m}\sqrt{\sum_{k=1}^{m}V_{ik}(\sum_{j=1}^{n}W_{kj}x_j^{(t)})^2}$, $g = \frac{f^{old}}{\partial W}$, $f^{new} = +\infty$, $\alpha = 1$
    while $f^{new} > f^{old}$ **do**
      $W^{new} = W - \alpha g$
      $W^{new} = Localize(W^{new}, s)$
      $W^{new} = tieWeights(W^{new}, k)$
      $W^{new} = orthogonalizeLocalRF(W^{new})$
      $W^{new} = tieWeights(W^{new}, k)$
      $f^{new} = \sum_{t=1}^{T}\sum_{i=1}^{m}\sqrt{\sum_{k=1}^{m}V_{ik}(\sum_{j=1}^{n}W_{kj}x_j^{(t)})^2}$
      $\alpha = 0.5\alpha$
    end while
    $W = W^{new}$
  until convergence

Neither GAF nor MTF images are natural images; they have no natural concepts such as "edges" and "angles". Thus, we propose to exploit the benefits of unsupervised pretraining with TICA to learn many diverse features with local orthogonality. In (Ngiam *et al.* 2010), the authors empirically demonstrate that tiled CNNs perform well with limited labeled data because the partial weight tying requires fewer parameters and reduces the need for a large amount of labeled data. Our data from the UCR Time Series Repository (Keogh *et al.* 2011) tends to have few instances (e.g., the "yoga" dataset has 300 labeled instance in the training set and 3000 unlabeled instance in the test set), so tiled CNNs are suitable for our learning task. Moreover, Tiled CNNs achieve good performance on large datasets (such as NORB and CIFAR). W

Typically, tiled CNNs are trained with two hyperparameters, the tiling size k and the number of feature maps l. In our experiments, we directly fixed the network structures without tuning these hyperparameters in loops, since the configuration is proved to be the

optimal in their work. Our experimental settings follow the default deep network structures and parameters in (Ngiam *et al.* 2010). Tiled CNNs with such configurations are reported to achieve the best performance on the NORB image classification benchmark. Although tuning the parameters will surely enhance performance, doing so may cloud our understanding of the power of the representation. Another consideration is computational efficiency. All of the experiments on the 12 datasets could be done in one day on a laptop with an Intel i7-3630QM CPU and 8GB of memory (our experimental platform). Thus, the results in this paper are a preliminary lower bound on the potential best performance. Thoroughly exploring network structures and parameters will be addressed in future work. The structure and parameters of the tiled CNN used in this paper are illustrated in Figure 4.5.

4.5.2 Experimental Settings and Data

We apply Tiled CNNs to classify time series using GASF, GADF and MTF representation on 20 datasets from UCR time series repository. More detailed statistics are summarized in Table 4.1. The datasets are pre-split into training and testing sets for experimental comparisons. For each dataset, the table gives its name, the number of classes, the number of training and test instances, and the length of the individual time series.

We apply Tiled CNNs to classify time series using GAF and MTF representations on 20 datasets from (Keogh *et al.* 2011) in different domains such as medicine, entomology, engineering, astronomy, signal processing, and others. The datasets are pre-split into training and testing sets to facilitate experimental comparisons. We compare the classification error rate of our GASF-GADF-MTF approach with previously published results of 6 best approaches proposed recently: Fast-Shapelets(Rakthanmanon & Keogh 2013), a 1NN classifier based on SAX with Bag-of-Patterns (SAX-BoP) (Lin, Khade, & Li 2012), a SAX based Vector Space Model (SAX-VSM)(Senin & Malinchik 2013), a classifier based on the

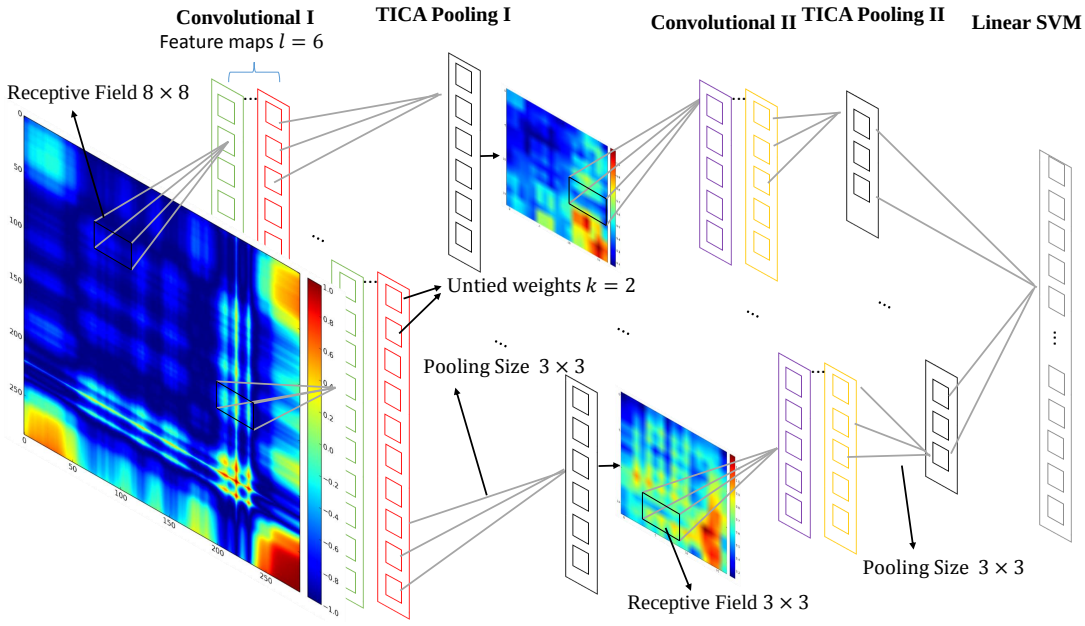

FIG. 4.5. Structure of the tiled convolutional neural networks. We fix the size of receptive fields to 8×8 in the first convolutional layer and 3×3 in the second convolutional layer. Each TICA pooling layer pools over a block of 3×3 input units in the previous layer without warping around the borders to optimize for sparsity of the pooling units. The number of pooling units in each map is exactly the same as the number of input units. The last layer is a linear SVM for classification. We construct this network by stacking two Tiled CNNs, each with 6 maps ($l = 6$) and tiling size $k = 1, 2, 3$.

Recurrence Patterns Compression Distance (RPCD) (Silva *et al.* 2013), a tree-based symbolic representation for multivariate time series (SMTS) (Baydogan & Runger 2014) and a SVM classifier based on a bag-of-features representation (TSBF) (Baydogan, Runger, & Tuv 2013).

In our experiments, the size of GAF image is regulated by the the number of PAA bins S_{GAF}. Given a time series X of size n, we divide the time series into S_{GAF} adjacent, non-overlapping windows along the time axis and extract the means of each bin. This enables us to construct the smaller GAF matrix $G_{S_{GAF} \times S_{GAF}}$. MTF requires the time series to be discretized into Q quantile bins to calculate the $Q \times Q$ Markov transition matrix, from which

Table 4.1. Statistics on 20 UCR datasets

	classes	train	test	length
50Words	50	450	455	270
Adiac	37	390	391	176
Beef	5	30	30	470
CBF	3	30	900	128
Coffee	2	28	28	286
ECG	2	100	100	96
Face(all)	14	560	1690	131
Face(four)	4	24	88	350
Fish	7	175	175	463
Gun-Point	2	50	150	150
Lighting-2	2	60	61	637
Lighting-7	7	70	73	319
OliveOil	4	30	30	570
OSUL.	6	200	242	427
SwedishL.	15	500	625	128
Synt.Cont.	6	300	300	60
Trace	4	100	100	275
TwoPat.	4	1000	4000	128
Wafer	2	1000	6174	152
Yoga	2	300	3000	426

we construct the raw MTF image $M_{n\times n}$ afterwards. Before classification, we shrink the MTF image size to $S_{MTF} \times S_{MTF}$ by the blurring kernel $\{\frac{1}{m^2}\}_{m\times m}$ where $m = \lceil \frac{n}{S_{MTF}} \rceil$. The Tiled CNN is trained with image size $\{S_{GAF}, S_{MTF}\} \in \{16, 24, 32, 40, 48\}$ and quantile size $Q \in \{8, 16, 32, 64\}$. At the last layer of the Tiled CNN, we use a linear soft margin SVM (Fan *et al.* 2008) and select C by 5-fold cross validation over $\{10^{-4}, 10^{-3}, \ldots, 10^{4}\}$ on the training set.

For each input of image size S_{GAF} or S_{MTF} and quantile size Q, we pretrain the Tiled CNN with the full unlabeled dataset (both training and test set) to learn the initial weights W through TICA. Then we train the SVM at the last layer by selecting the penalty factor C with cross validation. Finally, we classify the test set using the optimal hyperparame-

Table 4.2. Summary of error rates for 3 classic baselines, 6 recently published best results and our approach. The symbols $\triangleleft$, $*$, † and • represent datasets generated from human motions, figure shapes, synthetically predefined procedures and all remaining temporal signals, respectively. For our approach, the numbers in brackets are the optimal PAA size and quantile size.

Dataset	Random-Guess	Fast-Shapelet	SAX-BoP	SAX-VSM	RPCD	SMTS	TSBF	GASF-GADF-MTF
50words •	0.98	N/A	0.466	N/A	0.2264	0.289	**0.209**	0.301 (16, 32)
Adiac $*$	0.972	0.514	0.432	0.381	0.3836	0.248	**0.245**	0.373 (32, 48)
Beef •	0.8	0.447	0.433	0.33	0.3667	0.26	0.287	**0.233 (64, 40)**
CBF †	0.667	0.053	0.013	0.02	N/A	0.02	**0.009**	**0.009 (32, 24)**
Coffee •	0.5	0.068	0.036	**0**	**0**	0.029	0.004	**0 (64, 48)**
ECG •	0.5	0.227	0.15	0.14	0.14	0.159	0.145	**0.09 (8, 32)**
FaceAll $*$	0.75	0.411	0.219	0.207	**0.1905**	0.191	0.234	0.237 (8, 48)
FaceFour $*$	0.98	0.090	0.023	**0**	0.0568	0.165	0.051	0.068 (8, 16)
fish $*$	0.857	0.197	0.074	**0.017**	0.1257	0.147	0.08	0.114 (8, 40)
Gun_Point $\triangleleft$	0.5	0.061	0.027	0.007	**0**	0.011	0.011	0.08 (32, 32)
Lighting2 •	0.5	0.295	0.164	0.196	0.2459	0.269	0.257	**0.114 (16, 40)**
Lighting7 •	0.857	0.403	0.466	0.301	0.3562	**0.255**	0.262	0.260 (16, 48)
OliveOil •	0.75	0.213	0.133	0.1	0.1667	0.177	**0.09**	0.2 (8, 48)
OSULeaf $*$	0.833	0.359	0.256	**0.107**	0.3554	0.377	0.329	0.358 (16, 32)
SwedishLeaf $*$	0.933	0.269	0.198	0.01	0.0976	0.08	0.075	**0.065 (16, 48)**
synthetic control †	0.8338	0.081	0.037	0.251	N/A	0.025	0.008	**0.007 (64, 48)**
Trace †	0.75	0.002	**0**	**0**	N/A	**0**	0.02	**0 (64, 48)**
Two Patterns †	0.75	0.113	0.129	0.004	N/A	0.003	0.001	0.091 (64, 32)
wafer •	0.5	0.004	0.003	0.0006	0.0034	**0**	0.004	**0 (64, 16)**
yoga $*$	0.5	0.249	0.17	0.164	0.134	**0.094**	0.149	0.196 (8, 32)
#wins	0	0	1	5	3	4	4	**9**

ters $\{S, Q, C\}$ with the lowest error rate on the training set. If two or more models tie, we prefer the larger S and Q because larger S helps preserve more information through the PAA procedure and larger Q encodes the dynamic transition statistics with more detail. Our model selection approach provides generalization without being overly expensive computationally.

4.5.3 Results and Discussion

We use Tiled CNNs to classify the single GASF, GADF and MTF images as well as the compound GASF-GADF-MTF images on 20 datasets. For the sake of space, we do not show the full results on single-channel images. Generally, our approach is not prone to overfitting by the relatively small difference between training and test set errors. One exception is the Olive Oil dataset with the MTF approach where the test error is significantly higher.

In addition to the risk of potential overfitting, we found that MTF has generally higher error rates than GAFs. This is most likely because of the uncertainty in the inverse map of MTF. Note that the encoding function from $-1/1$ rescaled time series to GAFs and MTF are both surjections. The map functions of GAFs and MTF will each produce only one image with fixed S and Q for each given time series X . Because they are both surjective mapping functions, the inverse image of both mapping functions is not fixed. However, the mapping function of GAFs on $0/1$ rescaled time series are bijective. As shown in a later section, we can reconstruct the raw time series from the diagonal of GASF, but it is very hard to even roughly recover the signal from MTF. Even for $-1/1$ rescaled data, the GAFs have smaller uncertainty in the inverse image of their mapping function because such randomness only comes from the ambiguity of $\cos(\phi)$ when $\phi \in [0, 2\pi]$. MTF, on the other hand, has a much larger inverse image space, which results in large variations when we try to recover the signal. Although MTF encodes the transition dynamics which are important features of time series, such features alone seem not to be sufficient for recognition/classification tasks.

Note that at each pixel, G_{ij} denotes the superstition/difference of the directions at t_i and t_j, M_{ij} is the transition probability from the quantile at t_i to the quantile at t_j. GAF encodes static information while MTF depicts information about dynamics. From this

point of view, we consider them as three "orthogonal" channels, like different colors in the RGB image space. Thus, we can combine GAFs and MTF images of the same size (i.e. $S_{GAFs} = S_{MTF}$) to construct a triple-channel image (GASF-GADF-MTF). It combines both the static and dynamic statistics embedded in the raw time series, and we posit that it will be able to enhance classification performance. In the experiments below, we pretrain and tune the Tiled CNN on the compound GASF-GADF-MTF images. Then, we report the classification error rate on test sets. In Table 4.2, the Tiled CNN classifiers on GASF-GADF-MTF images achieved significantly competitive results with 9 other state-of-the-art time series classification approaches.

4.6 Image Recovery on GASF for Time Series Imputation with Denoised Auto-encoder

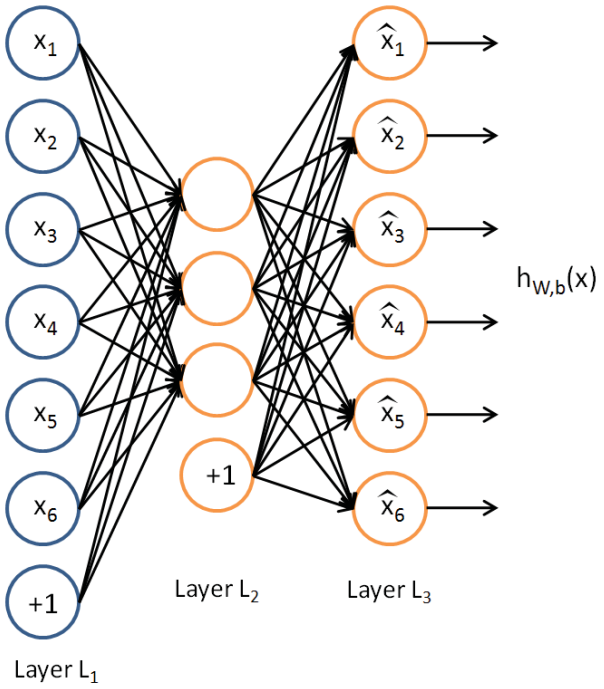

FIG. 4.6. Network structure of Auto-encoder.

We use Denoised AutoEncoders (DAE) to learn a generative model of time seris on GASF. DAE is a variation of a standard AutoEncoder (Figure 4.6). An Autoencoder neural network is an unsupervised learning algorithm that applies backpropagation, setting the target values to be equal to the inputs.

The Auto-encoder tries to learn a function $h_{W,b}(x) \approx x$. In other words, it is trying to learn an approximation to the identity function, so as to output $\hat{x}$ that is similar to x. The identity function seems a particularly trivial function to be trying to learn; but by placing constraints on the network, such as by limiting the number of hidden units, we can discover interesting structure in the data. If the input were completely random say, each x_i comes from an IID Gaussian independent of the other features, then this compression task would be very difficult. But if there is structure in the data, for example, if some of the input features are correlated, then this algorithm will be able to discover these correlations (Ng 2011).

The idea behind Denoising Auto-encoders (DAE) is simple and similar to AE. In order to force the hidden layer to discover more robust features and prevent it from simply learning the identity, we train the Auto-encoder to reconstruct the input from a corrupted version of it. The DAE is a stochastic version of the auto-encoder. Intuitively, a DAE does two things: try to encode the input (preserve the information about the input), and try to undo the effect of a corruption process stochastically applied to the input of the AE. The latter can only be done by capturing the statistical dependencies between the inputs. The DAE can be understood from different perspectives including the manifold learning perspective, stochastic operator perspective, bottom-up information theoretic perspective and top-down generative model perspective (Vincent *et al.* 2008). The stochastic corruption process in this section randomly sets some of the inputs (as many as half of them) to zero (salt-and-pepper noise). Hence the DAE is trying to predict the corrupted (i.e., missing) values from the uncorrupted (i.e., non-missing) values, for randomly selected subsets of

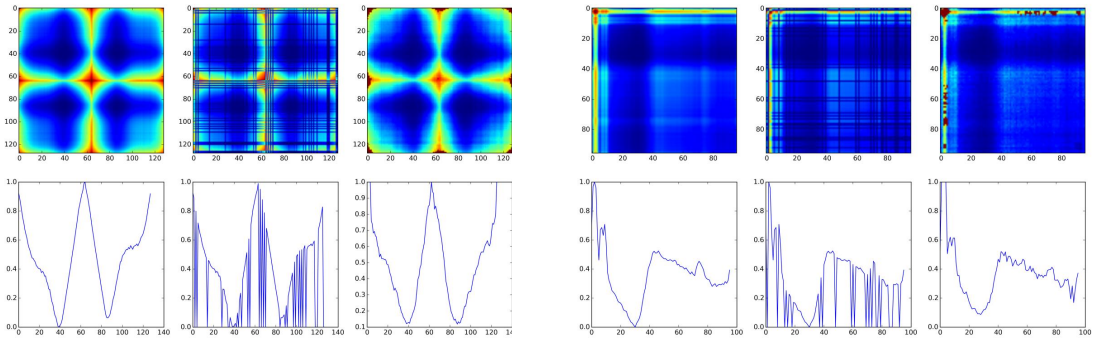

FIG. 4.7. Pipeline of time series imputation by image recovery. Raw GASF $\rightarrow$ "broken" GASF $\rightarrow$ recovered GASF (top), Raw time series $\rightarrow$ corrupted time series with missing value $\rightarrow$ predicted time series (bottom) on dataset "SwedishLeaf" (left) and "ECG" (right).

missing patterns. Note how being able to predict any subset of variables from the rest is a sufficient condition for completely capturing the joint distribution between a set of variables.

As previously mentioned, the mapping functions from $-1/1$ rescaled time series to GAFs are surjections. The uncertainty among the inverse images come from the ambiguity of the $\cos(\phi)$ when $\phi \in [0, 2\pi]$. However the mapping functions of $0/1$ rescaled time series are bijections. The main diagonal of GASF, i.e. $\{G_{ii}\} = \{\cos(2\phi_i)\}$ allows us to precisely reconstruct the original time series by

$$\cos(\phi) = \sqrt{\frac{\cos(2\phi) + 1}{2}} \qquad \phi \in [0, \frac{\pi}{2}] \tag{4.10}$$

Thus, we can predict missing values among time series through recovering the "broken" GASF images. During training, we manually add "salt-and-pepper" noise (i.e., randomly set a number of points to 0) to the raw time series and transform the data to GASF images. Then a single layer Denoised Auto-encoder (DA) is fully trained as a generative model to reconstruct GASF images. Note that at the input layer, we do not add noise again to the "broken" GASF images. A Sigmoid function helps to learn the nonlinear features at

the hidden layer. At the last layer we compute the Mean Square Error (MSE) between the original and "broken" GASF images as the loss function to evaluate fitting performance. To train the models simple batch gradient descent is applied to back propagate the inference loss. For testing, after we corrupt the time series and transform the noisy data to "broken" GASF, the trained DA helps recover the image, on which we extract the main diagonal to reconstruct the recovered time series. To compare the imputation performance, we also test standard DA with the raw time series data as input to recover the missing values (Figure. 4.7).

4.6.1 Experiment Setting

For the DA models we use batch gradient descent with a batch size of 20. Optimization iterations run until the MSE changed less than a threshold of 10^{-3} for GASF and 10^{-5} for raw time series. A single hidden layer has 500 hidden neurons with sigmoid functions. We choose four dataset of different types from the UCR time series repository for the imputation task: "Gun Point" (human motion), "CBF" (synthetic data), "SwedishLeaf" (figure shapes) and "ECG" (other remaining temporal signals). To explore if the statistical dependency learned by the DA can be generalized to unknown data, we use the above four datasets and the "Adiac" dataset together to train the DA to impute two totally unknown test datasets, "Two Patterns" and "wafer" (We name these synthetic miscellaneous datasets "7 Misc"). To add randomness to the input of DA, we randomly set 20% of the raw data among a specific time series to be zero (salt-and-pepper noise). Our experiments for imputation are implemented with Theano (Bastien *et al.* 2012). To control for the random initialization of the parameters and the randomness induced by gradient descent, we repeated every experiment 10 times and report the average MSE.

Table 4.3. MSE of imputation on time series using raw data and GASF images.

Dataset	Full MSE		Interpolation MSE	
	Raw	GASF	Raw	GASF
ECG	0.01001	0.01148	0.02301	0.01196
CBF	0.02009	0.03520	0.04116	0.03119
Gun Point	0.00693	0.00894	0.01069	0.00841
SwedishLeaf	0.00606	0.00889	0.01117	0.00981
7 Misc	0.06134	0.10130	0.10998	0.07077

4.6.2 Results and Discussion

In Table 4.3, "Full MSE" means the MSE between the complete recovered and original sequence and "Imputation MSE" means the MSE of only the unknown points among each time series. Interestingly, DA on the raw data perform well on the whole sequence, generally, but there is a gap between the full MSE and imputation MSE. That is, DA on raw time series can fit the known data much better than predicting the unknown data (like overfitting). Predicting the missing value using GASF always achieves slightly higher full MSE but the imputation MSE is reduced by 12.18%-48.02%. We can observe that the difference between the full MSE and imputation MSE is much smaller on GASF than on the raw data. Interpolation with GASF has more stable performance than on the raw data.

Why does predicting missing values using GASF have more stable performance than using raw time series? Actually, the transformation maps of GAFs are generally equivalent to a kernel trick. By defining the inner product $k(x_i, x_j)$, we achieve data augmentation by increasing the dimensionality of the raw data. By preserving the temporal and spatial information in GASF images, the DA utilizes both temporal and spatial dependencies by considering the missing points as well as their relations to other data that has been explicitly encoded in the GASF images. Because the entire sequence, instead of a short subsequence, helps predict the missing value, the performance is more stable as the full MSE and impu-

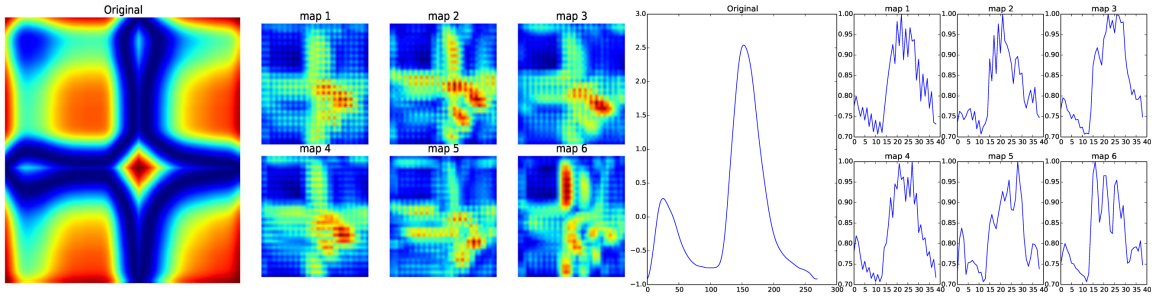

FIG. 4.8. (a) Original GASF and its six learned feature maps before the SVM layer in Tiled CNNs (left). (b) Raw time series and its reconstructions from the main diagonal of six feature maps on '50Words' dataset (right).

tation MSE are close.

4.7 Analysis on Features and Weights Learned by Tiled CNNs and DA

In contrast to the cases in which the CNNs is applied in natural image recognition tasks, neither GAFs nor MTF have natural interpretations of visual concepts like "edges" or "angles". In this section we analyze the features and weights learned through Tiled CNNs to explain why our approach works.

Figure 4.8 illustrates the reconstruction results from six feature maps learned through the Tiled CNNs on GASF (by Eqn 4.10). The Tiled CNNs extracts the color patch, which is essentially a moving average that enhances several receptive fields within the nonlinear units by different trained weights. It is not a simple moving average but the synthetic integration by considering the 2D temporal dependencies among different time intervals, which is a benefit from the Gramian matrix structure that helps preserve the temporal information. By observing the orthogonal reconstruction from each layer of the feature maps, we can clearly observe that the tiled CNNs can extract the multi-frequency dependencies through the convolution and pooling architecture on the GAF and MTF images to preserve the trend while addressing more details in different subphases. The high-leveled feature maps

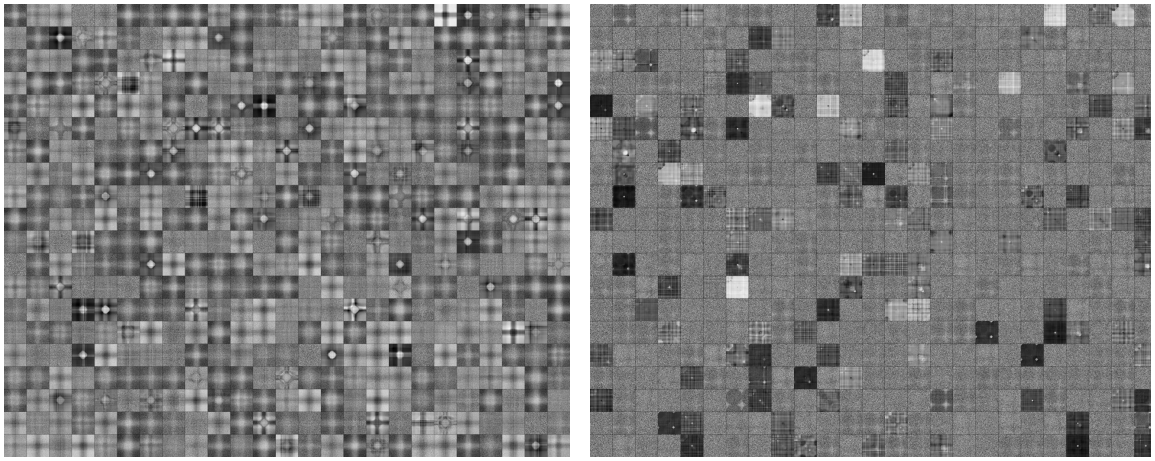

FIG. 4.9. All 500 filters learned by DA on the "Gun Point" (left) and "7 Misc" (right) dataset.

learned by the Tiled CNN are equivalent to a multi-frequency approximator of the original curve. Our experiments also demonstrates the learned weight matrix W with the constraint $WW^T = I$, which makes effective use of local orthogonality. The TICA pretraining provides the built-in advantage that the function w.r.t the parameter space is not likely to be ill-conditioned as $WW^T = 1$. The weight matrix W is quasi-orthogonal and approaching 0 without large magnitude. This implies that the condition number of W approaches 1 and helps the system to be well-conditioned.

As for imputation, because the GASF images have no concept of "angle" and "edge", DA actually learned different prototypes of the GASF images (Table 4.9). We find that there is significant noise in the filters on the "7 Misc" dataset because the training set is relatively small to better learn different filters. Actually, all the noisy filters with no patterns work like one Gaussian noise filter.

4.8 Experiments on Trajectory Data

We have demonstrated the effectiveness of GAF and MTF the benchmark time series datasets as diverse as shape, physiological surveillance and industry from the UCR time series repository. In this section we describe an application of our approaches to classify spatial-temporal trajectory data. The trajectory data is complex because patterns of movement are often driven by unperceived goals and constrained by an unknown environment.

To compare our results with other benchmark approaches including the seminal work from (Lee *et al.* 2008), we run experiments on two benchmark datasets, the animal movement dataset (Animal) and the hurricane track dataset (Hurricane) (Figure 4.10). Both datasets have trajectories of unequal length. For the "Animal" dataset, the x and y coordinates are extracted from animal movements observed in June 1995. It is divided into three classes by species: elk, deer, and cattle, as shown in Figure 15. The numbers of trajectories (points) are 38 (7117), 30 (4333), and 34 (3540), respectively. In the "Hurricane" dataset, the latitude and longitude are extracted from Atlantic hurricanes for the years 1950 through 2006. The Saffir-Simpson scale classifies hurricanes into categories 1-5 by intensity. A high category number indicates high intensity. Categories 2 and 3 are chosen for two classes. The numbers of trajectories (points) are 61 (2459) and 72 (3126), respectively. Both datasets are pre-split into two parts for training (80%) and testing (20%). Figure 4.10 shows the overview of the trajectory data. Table 4.4 provides the classes, training size, testing size, minimum length and maximum length of the trajectory data.

Table 4.4. Summary statistics of two trajectory datasets.

Dataset	Classes	Training Size	Testing Size	Min Length	Max Length
Animal Tracking	3	80	18	10	291
Hurricane	2	112	21	11	108

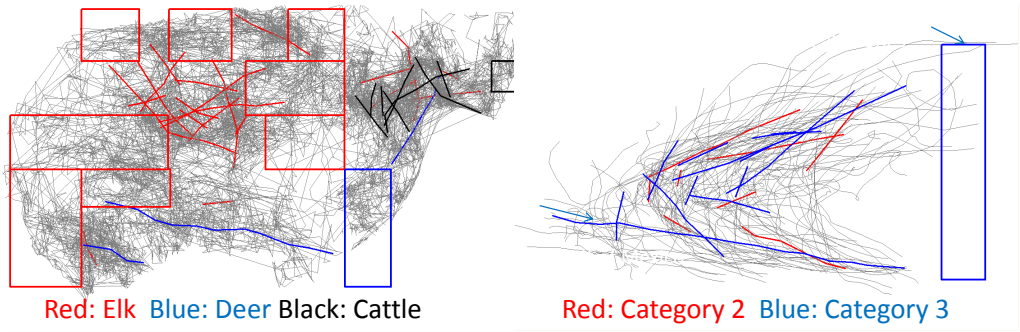

FIG. 4.10. Overview of the trajectory and the RB-TB features learnt in (a). Animal tracking data (left) and (b). Hurricane data (right)

4.8.1 Hilbert Space Filling Curves

Spatial-temporal trajectory data is commonly multi-dimensional. We use Hilbert Space Filling Curves (SFC) to transform the trajectory into time series while preserving the spatial-temporal information.

Space filling has been studied by the mathematicians since the late 19th century when the first graphical representation was proposed by David Hilbert in 1891 (Hilbert 1891). Space filling curves provide a linear mapping from the multi-dimensional space to the 1-dimensional space. This mapping can be thought of as dividing D-dimensional space into D-dimensional hypercubes with a line passing through each hypercube. Recently, filling curve based approaches have shown to be able to preserve locality between objects in the multidimensional space in the linear space, and thus have been applied to different tasks like clustering (Moon *et al.* 2001), high dimensional outlier detection (Angiulli & Pizzuti 2005), and trajectory query (Ding, Trajcevski, & Scheuermann 2008) and classification (Gandhi & Oates 2015). Figure 4.11 (a) shows SFC examples of order $\{1,2,3,4,5,6\}$.

Basically, the SFC of order 1 divides the square into 4 area. For the Hilbert curve with order 2, each sub-area of the curve with order 1 is further divided into 4 sub-areas. This

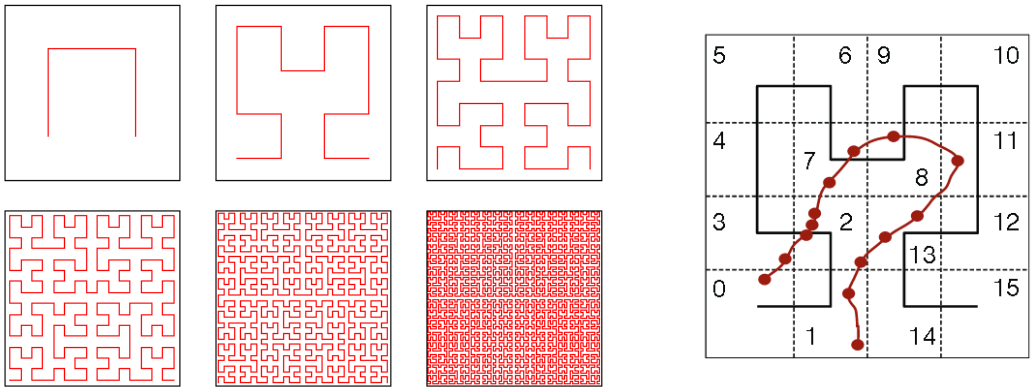

FIG. 4.11. (a). Hilbert space filling curve of order {1,2,3,4,5,6} in 2-dimensional space (left) (b). An example of the transformation from 2-dimensional trajectory to 1-dimensional time series using HSCF of order 2 (right).

process goes on as the order of the SFC increases. It is clear that the number of sub-areas in 2 dimensional SFC is 4^{order}. To convert 2-dimensional data points to 1-dimensional points, each sub-area is integer numbered from 0 to $4^{order} - 1$ starting from the lower left corner as 0 to the lower right corner. All other sub-areas are numbered in order of occurrence of the corresponding vertex as shown in Figure 4.11 (b) when order = 2. It also shows the example transformation process from a 2D trajectory to a sequence of scalars (time series). The final time series generated after SFC transformation is T = [0, 3, 2, 2, 2, 7, 7, 8, 11, 13, 13, 2, 1, 1].

We map the trajectory points by the visiting order of the SFC embedded in the trajectory manifold space to the index sequence by the recorded times. The produced time series can be used for classification using our algorithm. This adds another hyperparameter called the SFC order, which decides the granularity of the space filling curve.

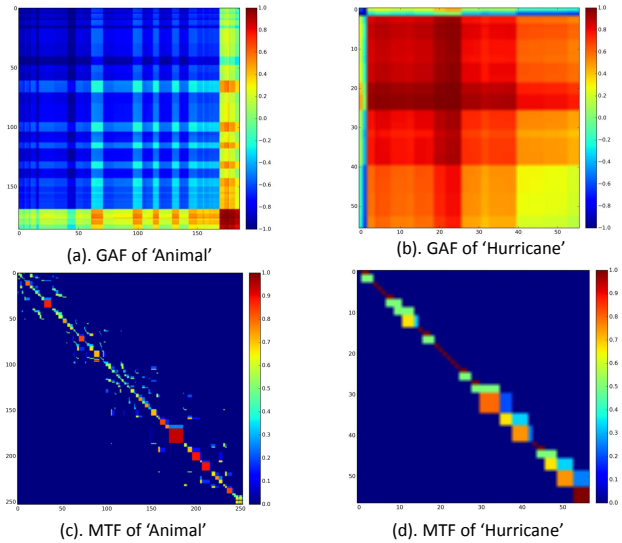

FIG. 4.12. Examples of GAF and MTF images generated from the time series on 'Animal' and 'Hurricane' datasets. The time series is produced using SFC from raw 2D trajectory.

4.8.2 Experiment Settings

The parameter settings are the same as the previous experiments on UCR datasets. The optimal SFC order is selected together with other parameters through 5-fold cross validation from $\{3,4,5,6,7,8,9,10\}$.

Note that both trajectory datasets have quite small sample size with varying length. When the trajectory length (as well as the time series length produced by SFC) is smaller than image size S, we uniformly duplicate each point in the time series in temporal order to stretch the sequence to length S. If the difference between the length of a time series and S is smaller than the original time series length, the interpolation strategy changes to random duplication instead of following the temporal order.

4.8.3 Results and Discussion

Both 'Animal' and 'Hurricane' datasets have been used in previous research (Lee *et al.* 2008; Gandhi & Oates 2015) to achieve state-of-the-art classification accuracy. Traclass give two algorithms, trajectory-based (TB-only) and region-based + trajectory-based (RB-TB) approaches based on features used for classification on these datastes. They carefully designed a hierarchy of features by partitioning trajectories and exploring two types of clustering. In (Gandhi & Oates 2015), the author used SFC transformation to linearly map the trajectory data to time series and classified the sequences based on symbolic discretization with the multiple normal distribution assumption.

After transforming the 2D trajectory data to time series using SFC, we generate the corresponding GAF and MTF images as shown in Figure 4.12. However, we found significant overfitting with CNNs even using 5-fold cross validation. This is probably because both the sample size and the time series length of the trajectory datasets are too small to avoid overfitting in neural networks. Previous work has discussed overfitting during cross validation and proposed potential techniques to address this problem (Ng 1997; Prechelt 1998). Here, we applied a simple and straight-forward hyperparameter selection approach to reduce classifier variance. For a given set of hyperparameter $\{S, Q, SFC_{order}\}$, after cross validation with different C values of the linear SVM, we compute the mean and standard deviation to get the 3σ lower bound over all C by

$$score_{3\sigma} = mean(Accuracy) - 3 \times STD(Accuracy) \tag{4.11}$$

By selecting the other hyperparameters $\{S, Q, SFC - order\}$ with the best statistical lower bound on the classifier performance over C, the optimal hyperparameters have lower variance while preserving lower bias. Using this hyperparameter selection approach, the

Table 4.5. Classification accuracy for TB-Only, RB-TB methods, multiple normal distribution based symbolic distance (NDist) and our algorithm (%).

Dataset	TB-Only	RB-TB	NDist	GAF-MTF
Animal Tracking	50	83.3	83.3	72.2
Hurricane	65.4	73.1	52.3	71.42

classification results are reported in Table 4.5.

We perform better than the TB-Only method on both datasets and almost as good as the RB-TB method on the 'Hurricane' dataset. However, both RB-TB and NDist methods outperform ours on the 'Animal' dataset. As shown in Figure 4.10, both region and trajectory based features are useful for classification. For the 'Hurricane' dataset, direction based features are more useful than region based features. Direction based features are quite easy to capture using our approach as the GAF is actually calculating the pairwise direction fields on each points in the trajectory data. For the 'Animal' dataset, region is very important as shown in Figure 4.10 (a). Elk, deer and cattle are almost separable just using location as their regions are clearly located at the left, right top and right bottom, respectively. When transforming the trajectory data into time series using SFC, two close regions might be mapped to different sub-areas with different SFC indexes. When the indexes of two close regions are also near, this can be handled by CNNs with its capability to capture the small shifting-invariance features. However, CNNs are not good at discriminating similar images with large shifting from each other. Thus, when the region information is preserved by the manner of shifting the specific patterns largely in the time series produced by SFC, CNNs might have difficulty capturing the region information.

Although our approach does not overtake other benchmark methods on both trajectory datasets, we provide a more general framework to encode the spatial-temporal patterns for classification tasks. Instead of complicated hand-tuned features, our approach can be

applied to a variety of time series and trajectory data. When the region of the trajectory is not significantly important or the direction feature dominates, our general methods work quite well. On large datasets where the volume of time series/trajectory data is big, our deep neural network based approach will greatly benefit from the large sample size in both feature learning and classification tasks.

4.9 Conclusion

This chapter described an off-line approach to spatially encode the temporal patterns for classification using convolutional neural networks. We created a pipeline for converting trajectory and time series data into novel representations, GAF and MTF images, and extracted high-level features from these using CNNs. The features were subsequently used for classification. We demonstrated that our approach yields competitive results when compared to state-of-the-art methods by searching a relatively small parameter space. We found that GAF-MTF multi-channel images are scalable to larger numbers of quasi-orthogonal features that yield more comprehensive images. Our analysis of high-level features learned from CNNs suggested Tiled CNNs work like multi-frequency moving averages that benefit from the 2D temporal dependency that is preserved by the Gramian matrix.

Chapter 5

SIGNIFICANT STATISTICS — ADAPTIVE NORMALIZED RISK-AVERTING TRAINING FOR NEURAL NETWORKS

5.1 Introduction

In last section, we investigated modeling temporal data through imaging its static and dynamic temporal correlations to learn the feature sets adaptively with the deep learning approaches. The significant non-linearity in DNNs make the training hard with a high chance to get stuck with local optima and plateaus. Meanwhile, the gradient vanishing/explosion problem exist when not using the Rectified Linear Unit, which can be alleviated by unsupervised pretraining techniques(Erhan *et al.* 2010). Some recent works point out that in high dimensional space (e.g. deep neural nets), the convexity is not needed because the local optima is good enough (Dauphin *et al.* 2014; Choromanska *et al.* 2014). However, all above techniques and works do not really demystify the non-convex optimization problem in DNNs and Recurrent Neural Networks (RNNs).

In this and next section, we will introduce a exponential-form based error estimator, whose optimality and robustness can be adjusted and achieved by tuning a index λ.

This new estimator provides another perspective to debunk the non-convexity in DNNs and RNNs.

5.2 Background

Deep neural networks (DNNs) are attracting attention largely due to their impressive empirical performance in image and speech recognition tasks. (Krizhevsky, Sutskever, & Hinton 2012; Hinton *et al.* 2012; Hannun *et al.* 2014). While Convolutional Networks (ConvNets) are the de facto state-of-the-art for visual recognition, Deep Belief Networks (DBN), Deep Boltzmann Machines (DBM) and Stacked Auto-encoders (SA) provide insights as generative models to learn the full generating distribution of input data. (Vincent *et al.* 2010; Hinton, Osindero, & Teh 2006; Salakhutdinov & Hinton 2009). Recently, researchers have investigated various techniques to improve the learning capacity of DNNs. Unsupervised pretraining using Restrict Boltzmann Machines (RBM), Denoised Autoencoders (DA) or Topographic ICA (TICA) has proved to be helpful for training DNNs with better weight initialization (Ngiam *et al.* 2010; Coates & Ng 2011). Specific types of deep network structures such as Network in Network (NIN) (Min Lin 2014) and Deeply Supervised Nets (DSN) (Lee *et al.* 2014) enhance the local and global modeling to enhance the feature learned by hidden layers. Rectified Linear Unit (ReLU) and variants are proposed as the optimal activation functions to better interpret hidden features (Nair & Hinton 2010; He *et al.* 2015). Various regularization techniques such as dropout (Srivastava *et al.* 2014) with Maxout (Goodfellow *et al.* 2013b) are proposed to regulate the DNNs to be less prone to overfitting.

Neural network models always lead to a non-convex optimization problem. The optimization algorithm impacts the quality of the local minimum because it is hard to find a global minimum or estimate how far a particular local minimum is from the best possible

solution. The most standard approach to optimize DNNs is Stochastic Gradient Descent (SGD). There are many variants of SGD and researchers and practitioners typically choose a particular variant empirically. While nearly all DNNs optimization algorithms in popular use are gradient-based, recent work has shown that more advanced second-order methods such as L-BFGS and Saddle-Free Newton (SFN) approaches can yield better results for DNN tasks (Ngiam *et al.* 2011; Dauphin *et al.* 2014). Second order derivatives can be addressed by hardware extensions (GPUs or clusters) or batch methods when dealing with massive data, SGD still provides a robust default choice for optimizing DNNs.

Instead of modifying the network structure or optimization techniques for DNNs, we focused on designing a new error function to convexify the error space. The convexification approach has been studied in the optimization community for decades, but has never been seriously applied within deep learning. Two well-known methods are the graduated non-convexity method (Blake & Zisserman 1987) and the LiuFloudas convexification method (Liu & Floudas 1993). LiuFloudas convexification can be applied to optimization problems where the error criterion is twice continuously differentiable, although determining the weight α of the added quadratic function for convexifying the error criterion involves significant computation when dealing with massive data and parameters.

Following the same name employed for deriving robust controllers and filters (Speyer, Deyst, & Jacobson 1974), a new type of Risk-Averting Error (RAE) is proposed theoretically for solving non-convex optimization problems (Lo 2010). Empirically, with the proposal of Normalized Risk-Averting Error (NRAE) and the Gradual Deconvexification method (GDC), this error criterion is proved to be competitive with the standard mean square error (MSE) in single layer and two-layer neural networks for solving data fitting and classification problems (Gui, Lo, & Peng 2014; Lo, Gui, & Peng 2012). Our work will consistently follow the paradigm of Lo's work. Interestingly, SimNets, a generalization of ConvNets that was recently proposed in (Cohen & Shashua 2014), uses the MEX

operator (whose name stands for Maximum-minimum-Expectation Collapsing Smooth) as an activation function to generalize ReLU activation and max pooling. We notice that the MEX operator with L_2 units has exactly the *same* mathematical form with NRAE. However, NRAE is still hard to optimize in practice due to plateaus and the unstable error space caused by the fixed large convexity index. GDC alleviates these problems but its performance is limited and suffers from the slow learning speed. Instead of fixing the convexity index λ, Adaptive Normalized Risk-Averting Training (ANRAT) optimizes NRAE by tuning λ adaptively using gradient descent. We give theoretical proofs of its optimal properties against the standard L_p-norm error. Our experiments on MNIST and CIFAR-10 with different deep/shallow neural nets demonstrate the effectiveness empirically. Being an optimization algorithm, our approach are not supposed to deal specifically with the problem of over-fitting, however we show that this can be handled by the usual methods of regularization such as weight decay or dropout.

In this section, we generalize RAE and NRAE to L_P norm and provide theoretical proofs on the convexity properties. We prove that NRAE is at least as good as MSE in solving Non-convex optimization problems when $|\lambda| \geq 1$. We develop an Adaptive Normalized Risk-Averting Training approach (ANRAT) to automatically tune the sensitive index λ through standard gradient descent. By analyzing the derivatives on λ, we show that our approach is analogous to Deconvexification by GDC while possessing more flexibility and fast convergence speed. Empirically, our approaches are able to overcome the underfitting problem encountered when training DNN more effectively than the pretraining + fine-tuning. Being an optimization algorithm, our approach are not supposed to deal specifically with the problem of over-fitting, however we show that this can be handled by the usual methods of regularization such as weight decay or dropout. Our experiments demonstrate competitive results with the state-of-the-art on several visual recognition tasks. On MNIST dataset without pretraining, we can achieve 0.52% error rate using only ConvNets

(LeNet5) with weight decaym which is the best results ever reported with only standard ConvNets. Our preliminary experiments on CIFAR-10 dataset show that by using ConvNets and dropout with ANRAT methods, we can achieve 82.12% test accuracy, which is superior to the results using Cross Entropy (CE) and dropout (Zeiler & Fergus 2013) as well as competitive with the results achieved by unsupervised pretraining (Coates & Ng 2011). The experiments on Denoised Auto-encoder also demonstrate ANRAT helps in unsupervised feature learning.

5.3 Reformulation on Error Criterion - Significant Statistics

We begin with the definition of RAE for the L_p norm and the theoretical justifications on its convexity property. RAE is not suitable for real applications since it is not bounded. Instead, NRAE is bounded to overcome the register overflow in real implementations. We prove that NRAE is quasi-convex,and thus shares the same global and local optimum with RAE. Moreover, we show the lower-bound of its performance is as good as L_p-norm error when the convexity index satisfies a constraint, which theoretically supports the ANRAT method proposed in the next section.

5.3.1 Risk-averting Error Criterion

Given training samples $\{\boldsymbol{X}, y\} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2),$
$..., (\boldsymbol{x}_m, y_m)\}$, $f(\boldsymbol{x}_i, \boldsymbol{W})$ is the learning model with parameters W. The loss function of L_p-norm error is defined as:

$$l_p(f(\boldsymbol{x}_i, \boldsymbol{W}), y_i) = \frac{1}{m} \sum_{i=1}^{m} ||f(\boldsymbol{x}_i, \boldsymbol{W}) - y_i||^p \tag{5.1}$$

When $p = 2$, Eqn. 6.1 denotes to the standard Mean Square Error (MSE). The Risk-

Averting Error criterion (RAE) corresponding to the L_p-norm error is defined by

$$RAE_{p,q}(f(\boldsymbol{x}_i, \boldsymbol{W}), y_i) = \frac{1}{m}\sum_{i=1}^{m} e^{\lambda^q ||f(\boldsymbol{x}_i, \boldsymbol{W}) - y_i||^p} \tag{5.2}$$

λ is the convexity index. It controls the size of the convexity region.

For clarity, we write the matrix derivatives as function form. In our proof we use quadratic form.

The Jacobian matrix of Eqn. 6.1 is

$$J_{p,q}(\boldsymbol{W}) = \frac{1}{m} p\lambda^q \sum_{i=1}^{m} e^{\lambda^q ||f(\boldsymbol{x}_i, \boldsymbol{W}) - y_i||^p} ||f(\boldsymbol{x}_i, \boldsymbol{W}) - y_i||^{p-1} \frac{\partial f(\boldsymbol{x}_i, \boldsymbol{W})}{\partial \boldsymbol{W}} \tag{5.3}$$

The Hessian matrix of Eqn. 6.1 is

$$H_{p,q}(\boldsymbol{W}) = \frac{p}{m}\sum_{i=1}^{m} e^{\lambda^q (f(\boldsymbol{x}_i, \boldsymbol{W}) - y_i)^p} \tag{5.4}$$

$$\times\{(p-1)\lambda^q ||f(\boldsymbol{x}_i, \boldsymbol{W}) - y_i||^{p-2} \frac{\partial f(\boldsymbol{x}_i, \boldsymbol{W})}{\partial \boldsymbol{W}}^2 \tag{5.5}$$

$$+p\lambda^{2q}(y_i - f(\boldsymbol{x}_i, \boldsymbol{W}))^{2p-2} \frac{\partial f(\boldsymbol{x}_i, \boldsymbol{W})}{\partial \boldsymbol{W}}^2 \tag{5.6}$$

$$+\lambda^q (y_i - f(\boldsymbol{x}_i, \boldsymbol{W}))^{p-1} \frac{\partial f^2(\boldsymbol{x}_i, \boldsymbol{W})}{\partial \boldsymbol{W}^2}\} \tag{5.7}$$

Because RAE has the sum-exponential form, its Hessian matrix is tuned exactly by the convexity index λ^q. The following theorem indicates the relation between the convexity index and its convexity region.

Theorem 1 (Convexity)**.** *Given the Risk-Averting Error criterion $RAE_{p,q}$ ($p, q \in \mathcal{N}^+$), which is twice continuous differentiable. $J_{p,q}(\boldsymbol{W})$ and $H_{p,q}(\boldsymbol{W})$ are the corresponding Jacobian and Hessian matrix. As $\lambda \to \pm\infty$, the convexity region monotonically expands to the entire parameter space except for the subregion $S := \{W \in \mathcal{R}^n | rank(H_{p,q}(\boldsymbol{W})) < n, H_{p,q}(\boldsymbol{W} < 0)\}$.*

Proof. Assume $p > 0$ and $q > 0$, matrix 5.4 is positive semi-definite. Note that both $\frac{\partial f(\boldsymbol{x}_i, \boldsymbol{W})^2}{\partial \boldsymbol{W}}$ and $\lambda^{2q}(y_i - f(\boldsymbol{x}_i, \boldsymbol{W}))^{2p-2}$ are positive semi-definite, matrix 6.7 is semi-positive definite, but Eqn. 6.6 and 6.8 may be indefinite. Let $\alpha_i(\boldsymbol{W}) = f(\boldsymbol{x}_i, \boldsymbol{W}) - y_i$, We rewrite Eqn. 6.7 in quadratic form:

$$= p\lambda^{2q}\alpha_i(\boldsymbol{W})^T \frac{\partial f(\boldsymbol{x}_i, \boldsymbol{W})^T}{\partial \boldsymbol{W}} \cdot \frac{\partial f(\boldsymbol{x}_i, \boldsymbol{W})}{\partial \boldsymbol{W}} \alpha_i(\boldsymbol{W}) \tag{5.8}$$

$$= p\lambda^{2q}\alpha_i(\boldsymbol{W})^T Q \Lambda Q^T \alpha_i(\boldsymbol{W}) \tag{5.9}$$

$$= p\lambda^{2q} S(\boldsymbol{W})^T \Lambda S(\boldsymbol{W}) \tag{5.10}$$

$\Lambda = diag[\Lambda_1, \Lambda_2, ..., \Lambda_m]$. Note that when p is a even number, Eqn. 6.6 is also positive semi-definite.

If $S(\boldsymbol{W})$ if a full-rank matrix, then Eqn. 6.9 is positive definite. When $\lambda^q \to \pm\infty$, the eigenvalues Λ becomes dominant in the leading principal minors (as well as eigenvalues) of the Hessian matrix to make $H_{p,q}(\boldsymbol{W})$ monotonically increasing with λ^q. Assume $P_\lambda := \{W \in \mathcal{R}^n | H_{p,q}(\boldsymbol{W} > 0)\}$, we have $P_{\lambda_1} \in P_{\lambda_2}$ when $\lambda_1 < \lambda_2$. Considering Eqn. 6.6, 6.7 and 6.8 are all bounded, $\exists\ \psi(W)$, s.t. $H_{p,q}(\boldsymbol{W}) > 0$ when $|\lambda^q| > \psi(W)$.

When $S(\boldsymbol{W})$ is not a full-rank matrix, the determinant of all its $\binom{n}{k}$ submatrices is 0. Thus, in the subregion $S := \{W \in \mathcal{R}^n | rank(H_{p,q}(\boldsymbol{W})) < n, H_{p,q}(\boldsymbol{W} < 0)\}$, there is no parameters satisfied the convexity conditions ($\bigcup P_\lambda$). □

Please refer to the supplementary material for the proof. Intuitively, the use of the RAE was motivated by its emphasizing large individual deviations in approximating functions and optimizing parameters in an exponential manner, thereby avoiding such large individual deviations and achieving robust performances. Theoretically, Theorem 7 states that when the convexity index λ increases to infinity, the convexity region in the parameter space of RAE expands monotonically to the entire space except the intersection of a finite

number of lower dimensional sets. The number of sets increases rapidly as the number m of training samples increases. Roughly speaking, larger λ and m cause the size of the convexity region to grow larger respectively in the error space of RAE.

When $\lambda \to \infty$, the error space can be perfectly stretched to be strictly convex, thus avoid the local optimum to guarantee a global optimum. Although RAE works well in theory, it is not bounded and suffers from the exponential magnitude and arithmetic overflow when using gradient descent in implementations .

5.3.2 Normalized Risk-Averting Error Criterion

RAE ensures the convexity of the error space to find the global optimum. By using NRAE, we relax the global optimum problem by finding a better local optimum to meet a theoretically and practically reasonable trade-off in real applications.

Given training samples $\{\boldsymbol{X}, y\} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), ..., (\boldsymbol{x}_m, y_m)\}$, $f(\boldsymbol{x}_i, \boldsymbol{W})$ is the learning model with parameters W. The Normalized Risk-Averting Error Criterion (NRAE) corresponding to the L_p-norm error is defined as:

$$
\begin{aligned}
& NRAE_{p,q}(f(\boldsymbol{x}_i, \boldsymbol{W}), y_i) \\
&= \frac{1}{\lambda^q} \log RAE_{p,q}(f(\boldsymbol{x}_i, \boldsymbol{W}), y_i) \\
&= \frac{1}{\lambda^q} \log \frac{1}{m} \sum_{i=1}^{m} e^{\lambda^q ||f(\boldsymbol{x}_i, \boldsymbol{W}) - y_i||^p}
\end{aligned}
\tag{5.11}
$$

Theorem 2 (Bounded)**.** *$NRAE_{p,q}(f(\boldsymbol{x}_i, \boldsymbol{W}), y_i)$ is bounded. $NRAE_{p,q}(f(\boldsymbol{x}_i, \boldsymbol{W}), y_i) \to minimax$ if $\lambda^q \to \infty$, $NRAE_{p,q}(f(\boldsymbol{x}_i, \boldsymbol{W}), y_i) \to L_p$-norm error if $\lambda^q \to 0$.*

Proof. Let

$$\alpha_i(\boldsymbol{W}) = f(\boldsymbol{x}_i, \boldsymbol{W}) - y_i \tag{5.12}$$

$$\alpha_{max}(\boldsymbol{W}) = \max\{\alpha_i(\boldsymbol{W}), i = 1, ..., m\} \tag{5.13}$$

$$\beta_i(\boldsymbol{W}) = e^{\lambda^q(||\alpha_{max}(\boldsymbol{W})||^p - ||\alpha_i(\boldsymbol{W}))||^p} \tag{5.14}$$

Then we can write Eqn. 5.11 as

$$\begin{aligned} NRAE_{p,q}(f(\boldsymbol{x}_i, \boldsymbol{W}), y_i) &= \frac{1}{\lambda^q} \log \frac{1}{m} e^{\lambda^q ||\alpha_{max}||^p} \sum_{i=1}^{m} \beta_i(\boldsymbol{W}) \\ &= -\frac{1}{\lambda^q} \log m + ||\alpha_{max}||^p + \frac{1}{\lambda^q} \log \sum_{i=1}^{m} \beta_i(\boldsymbol{W}) \qquad (5.15) \\ &\leq -\frac{1}{\lambda^q} \log m + ||\alpha_{max}||^p + \frac{1}{\lambda^q} \log m \cdot e^{\lambda^q ||\alpha_{max}||^p} \\ &= 2||\alpha_{max}||^p \qquad (5.16) \end{aligned}$$

□

The proof is provided in the supplemental materials. Briefly, NRAE is bounded by functions independent of λ and no overflow occurs for $\lambda \gg 1$. The following theorem states the quasi-convexity of NRAE.

Theorem 3 (Quasi-convexity)**.** *Given a parameter space* $\{W \in \mathcal{R}^n\}$*, Assume* $\exists\, \psi(W)$*, s.t.* $H_{p,q}(\boldsymbol{W}) > 0$ *when* $|\lambda^q| > \psi(W)$ *to guarantee the convexity of* $RAE_{p,q}(f(\boldsymbol{x}_i, \boldsymbol{W}), y_i)$*. Then,* $NRAE_{p,q}(f(\boldsymbol{x}_i, \boldsymbol{W}), y_i)$ *is* ***quasi-convex*** *and share the same local and global optimum with* $RAE_{p,q}(f(\boldsymbol{x}_i, \boldsymbol{W}), y_i)$*.*

Proof. If $RAE_{p,q}(f(\boldsymbol{x}_i, \boldsymbol{W}), y_i)$ is convex, it is quasi-convex. $\log$ function is monotonically increasing, so the composition $\log RAE_{p,q}(f(\boldsymbol{x}_i, \boldsymbol{W}), y_i)$ is quasi-convex. [1]

[1] Because the function f defined by $f(x) = g(U(x))$ is quasi-convex if the function U is quasiconvex and

$\log$ is a strictly monotone function and $NRAE_{p,q}(f(\boldsymbol{x}_i, \boldsymbol{W}), y_i)$ is quasi-convex, so it shares the same local and global minimizer with $RAE_{p,q}(f(\boldsymbol{x}_i, \boldsymbol{W}), y_i)$. □

The convexity region of NRAE is consistent with RAE. To interpret this statement in another perspective, the $\log$ function is a strictly monotone function. Even if RAE is not strictly convex, NRAE still shares the same local and global optimum with RAE. If we define the mapping function $f : RAE \rightarrow NRAE$, it is easy to see that f is bijective and continuous. Its inverse map f^{-1} is also continuous, so that f is an open mapping. Thus, it is easy to prove that the mapping function f is a homeomorphism to preserve all the topological properties of the given space.

The above theorems state the consistent relations among NRAE, RAE and MSE. It is proven that the greater the convexity index λ, the larger is the convex region is. Intuitively, increasing λ creates tunnels for a local-search minimization procedure to travel through to a good local optimum. However, we care about the justification on the advantage of NRAE against MSE. Theorem 9 provides the theoretical justification for the performance lower-bound of NRAE.

Theorem 4 (Lower-bound)**.** *Given training samples* $\{\boldsymbol{X}, y\} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), ..., (\boldsymbol{x}_m, y_m)\}$ *and the model* $f(\boldsymbol{x}_i, \boldsymbol{W})$ *with parameters* W*. If* $\lambda^q \geq 1$*,* $p, q \in \mathcal{N}^+$ *and* $p \geq 2$*, then both* $RAE_{p,q}(f(\boldsymbol{x}_i, \boldsymbol{W}), y_i)$ *and* $NRAE_{p,q}(f(\boldsymbol{x}_i, \boldsymbol{W}), y_i)$ *always have the higher chance to find a better local optimum than the standard* L_p*-norm error due to the expansion of the convexity region.*

Proof. Let $h_p(\boldsymbol{W})$ denotes the Hessian matrix of standard L_p-norm error (Eqn. 6.1), note

---

the function g is increasing.

$\alpha_i(\boldsymbol{W}) = f(\boldsymbol{x}_i, \boldsymbol{W}) - y_i$ we have

$$\begin{aligned} h_p(\boldsymbol{W}) &= \frac{p}{m}\sum_{i=1}^{m}\{(p-1)\alpha_i(\boldsymbol{W})^{p-2}\frac{\partial f(\boldsymbol{x}_i, \boldsymbol{W})^2}{\partial \boldsymbol{W}} \\ &+ \alpha_i(\boldsymbol{W})^{p-1}\frac{\partial f^2(\boldsymbol{x}_i, \boldsymbol{W})}{\partial \boldsymbol{W}^2}\} \end{aligned} \tag{5.17}$$

Since $\lambda^q \geq 1$, let $diag_{eig}$ denotes the diagonal matrix of the eigenvalues from SVD decomposition. $\succeq$ here means 'element-wise greater'. When $A \succeq B$, each element in A is greater than B. Then we have

$$\begin{aligned} diag_{eig}[H_{p,q}(\boldsymbol{W})] &\succeq diag_{eig}[h_p(\boldsymbol{W}) + \\ &\quad \frac{p^2}{m}\sum_{i=1}^{m}||\alpha_i(\boldsymbol{W})||^{2p-2}\frac{\partial f(\boldsymbol{x}_i, \boldsymbol{W})}{\partial \boldsymbol{W}}^2\}] \\ &\succeq diag_{eig}[h_p(\boldsymbol{W})] \end{aligned} \tag{5.18}$$

This indicates that the $RAE_{p,q}(f(\boldsymbol{x}_i, \boldsymbol{W}), y_i)$ always has larger convexity regions than the standard L_p-norm error to better enable escape of local minima. Because $NRAE_{p,q}(f(\boldsymbol{x}_i, \boldsymbol{W}), y_i)$ is quasi-convex, sharing the same local and global optimum with $RAE_{p,q}(f(\boldsymbol{x}_i, \boldsymbol{W}), y_i)$, the above conclusions are still valid. □

Roughly speaking, NRAE always has a larger convexity region than the standard L_p-norm error in terms of their Hessian matrix when $\lambda \geq 1$. This property guarantees the higher probability to escape poor local optima using NRAE. In the worst case, NRAE will perform as good as standard L_p-norm error if the convexity region shrinks as λ decreases or the local search deviates from the "tunnel" of convex regions.

More specifically, $NRAE_{p,q}(f(\boldsymbol{x}_i, \boldsymbol{W}), y_i)$

- approaches the standard L_p-norm error as $\lambda^q \to 0$.

- approaches the minimax error criterion $inf_W \alpha_{max}(\boldsymbol{W})$ as $\lambda^q \to \infty$.

Intuitively, when $\lambda^q \to \infty$, we can roughly draw the second conclusion based on Eqn. 6.3. When $\lambda^q \to 0$, we have [2]

$$\begin{aligned} NRAE_{p,q}(f(\boldsymbol{x}_i, \boldsymbol{W}), y_i) &= \frac{1}{\lambda^q} \log \frac{1}{m} \sum_{i=1}^{m} e^{\lambda^q ||f(\boldsymbol{x}_i, \boldsymbol{W}) - y_i||^p} \\ &= \frac{1}{\lambda^q} \log(1 + \frac{1}{m} \sum_{i=1}^{m} \lambda^q ||f(\boldsymbol{x}_i, \boldsymbol{W}) - y_i||^p + O(\lambda^{2q})) \\ &= \frac{1}{m} \sum_{i=1}^{m} ||f(\boldsymbol{x}_i, \boldsymbol{W}) - y_i||^p \end{aligned} \tag{5.19}$$

Please refer to the supplemental materials for the proofs. More rigid proofs that can be generalized to L_p-norm error are also given in (Lo 2010). In SimNets, the authors also include quite similar discussions about the robustness with respect to L_p-norm error (Cohen & Shashua 2014).

5.4 Learning Methods

We propose a novel learning method to training DNNs with NRAE, called the Adaptive Normalized Risk-Avering Training (ANRAT) approach. Instead of manually tuning λ like GDC (Lo, Gui, & Peng 2012), we learn λ adaptively in error backpropagation by considering λ as a parameter instead of a hyperparameter. The learning procedure is standard batch SGD. We show it works quite well in theory and practice.

The loss function of ANRAT is

$$l(W, \lambda) = \frac{1}{\lambda^q} \log \frac{1}{m} \sum_{i=1}^{m} e^{\lambda^q ||f(\boldsymbol{x}_i, \boldsymbol{W}) - y_i||^p} + a||\lambda||^{-r} \tag{5.20}$$

[2] Consider the rules $e^x = 1 + x + \frac{x^2}{2} + \cdots (x \to 0)$ and $\log x = x - \frac{x^2}{2} + \cdots (x \to 0)$

Except for NRAE, we use a penalty term $a||\lambda||^{-r}$ to control the changing rate of λ. While minimize the NRAE score, small λ is penalized to regulate the convexity region. a is a hyperparameter to control the penalty index. The first-order derivatives on weight and λ are

$$\frac{dl(W,\lambda)}{dW} = \frac{p\sum_{i=1}^{m} e^{\lambda^q \alpha_i(W)^{p-1}} \frac{\partial f(\boldsymbol{x}_i,\boldsymbol{W})}{\partial \boldsymbol{W}}}{\sum_{i=1}^{m} e^{\lambda^q \alpha_i(W)^{p-1}}} \tag{5.21}$$

$$\frac{dl(W,\lambda)}{d\lambda} = \frac{-q}{\lambda^{q+1}} \log \frac{1}{m} \sum_{i=1}^{m} e^{\lambda^q \alpha_i(W)^p} \tag{5.22}$$

$$+ \frac{q}{\lambda} \frac{\sum_{i=1}^{m} e^{\lambda^q \alpha_i(W)^p} \alpha_i(W)^p}{\sum_{i=1}^{m} e^{\lambda^q \alpha_i(W)^p}} \tag{5.23}$$

$$- ar\lambda^{-r-1} \tag{5.24}$$

We make a transformation on Eqn. 5.24 to better understand the gradient with respect to λ. Note that $k_i = \frac{e^{\lambda^q \alpha_i(W)^p}}{\sum_{i=1}^{m} e^{\lambda^q \alpha_i(W)^p}}$ is actually performing like a probability ($\sum_{i=1}^{m} k_i = 1$). Ignoring the penalty term, Eqn. 5.24 can be formulated as follows:

$$\begin{aligned} \frac{dl(W,\lambda)}{d\lambda} &= \frac{q}{\lambda}(\sum_{i=1}^{m} k_i \alpha_i(\boldsymbol{W})^p - NRAE) \\ &= \frac{q}{\lambda}(\mathbb{E}(\alpha(\boldsymbol{W})^p) - NRAE) \\ &\approx \frac{q}{\lambda}(L_P\text{-norm error} - NRAE) \end{aligned} \tag{5.25}$$

Note that as $\alpha_i(\boldsymbol{W})^p$ becomes smaller, the expectation on $\alpha_i(\boldsymbol{W})^p$ approaches the standard L_p-norm error. Thus, the gradient on λ is approximately the difference between NRAE and the standard L_p-norm error. Because large λ can incur plateaus to prevent NRAE from finding better optima using batch SGD (Lo, Gui, & Peng 2012), they need GDC to gradually deconvexify the NRAE to make the error space well shaped and stable. Through Eqn. 6.11, ANRAT solve this problem in a more flexible and adaptive manner.

When NRAE is larger, Eqn. 6.11 remains negative and makes λ increase to enlarge the convexity region, facilitating the search in the error space for better optima. When NRAE is smaller, the learned parameters are seemingly going through the optimal "tunnel" for better optima. Eqn. 6.11 becomes positive to decrease λ and helps NRAE not deviate far from the manifold of the standard L_p-norm error to make the error space stable without large plateaus. Thus, ANRAT adaptively adjusts the convexity index to find an optimal trade-off between better solutions and stability.

This training approach has more flexibility. The gradient on λ as the weighted difference between NRAE and the standard L_P-norm error, enables NRAE to approach the L_P-norm error by adjusting λ gradually. Intuitively, it keeps searching the error space near the manifold of the L_p-norm error to find better optima in a way of competing with and at the same time relying on the standard L_p-norm error space.

In Eqn. 5.20, the penalty weight a and index r control the convergence speed by penalizing small λ. Smaller a emphasizes tuning λ to allow faster convergence speed between NRAE and L_p-norm error. Larger a forces larger λ for a better chance to find a better local optimum but runs the risk of plateaus and deviating far from the stable error space. r regulates the magnitude of λ and its derivatives in gradient descent.

5.5 Experiments

We present the results from a series of experiments designed on the MNIST and CIFAR-10 datasets to test the effectiveness of ANRAT for visual recognition with DNNs. We did not explore the full hyperparameter in Eqn. 5.20. Instead we fix the hyperparameter at $p = 2$, $q = 2$ and $r = 1$ to mainly compare with MSE. So the final loss function of ANRAT we optimized is

$$l(W,\lambda) = \frac{1}{\lambda^2}\log\frac{1}{m}\sum_{i=1}^{m} e^{\lambda^2||f(\boldsymbol{x}_i,\boldsymbol{W})-y_i||^2} + a|\lambda|^{-1} \tag{5.26}$$

This loss function is minimized by batch SGD without complex methods, such as momentum, adaptive/hand tuned learning rates or tangent prop. The learning rate and penalty weight a are selected in $\{1, 0.5, 0.1\}$ and $\{1, 0.1, 0.001\}$ on validation sets respectively. The initial λ is fixed at 10. We use the hold-out validation set to select the best model, which is used to make predictions on the test set. All experiments are implemented quite easily in Python and Theano to obtain GPU acceleration (Bastien *et al.* 2012).

The MNIST dataset (LeCun *et al.* 1998) consists of hand written digits 0-9 which are 28x28 in size. There are 60,000 training images and 10,000 testing images in total. We use 10000 images in training set for validation to select the hyperparameters and report the performance on the test set. We test our method on this dataset without data augmentation.

The CIFAR-10 dataset (Krizhevsky & Hinton 2009) is composed of 10 classes of natural images. There are 50,000 training images in total and 10,000 testing images. Each image is an RGB image of size 32x32. For this dataset, we adapt pylearn2 (Goodfellow *et al.* 2013a) to apply the same global contrast normalization and ZCA whitening as was used by Goodfellow et. al (Goodfellow *et al.* 2013b). We use the last 10,000 images of the training set as validation data for hyperparameter selection and report the test accuracy.

5.6 Results and Discussion

5.6.1 Results on ConvNets

On the MNIST dataset we use the same structure of LeNet5 with two convolutional max-pooling layers but followed by only one fully connected layer and a densely connected

softmax layer. The first convolutional layer has 20 feature maps of size 5×5 and max-pooled by 2×2 non-overlapping windows. The second convolutional layer has 50 feature maps with the same convolutional and max-pooling size. The fully connected layer has 500 hidden units. An l_2 prior was used with the strength 0.05 in the Softmax layer. Trained by ANRAT, we can obtain a test set error of 0.52%, which is the best result we are aware of that does not use dropout on the pure ConvNets. We summarize the best published results on the standard MNIST dataset in Table 6.2.

The best performing neural networks for pure ConvNets that does not use dropout or unsupervised pretraining achieve an error of about 0.69% (Ngiam *et al.* 2011). They demonstrated this performance with L-BFGS. Using dropout, ReLU and a response normalization layer, the error reduces to 0.55% (Zeiler & Fergus 2013). Prior to that, Jarrett et. al showed by increasing the size of the network and using unsupervised pretraining, they can obtain a better result at 0.53% (Jarrett *et al.* 2009). Previous state of the art is 0.47% (Goodfellow *et al.* 2013b) for a single model without on the original MNIST dataset. Using batch SGD to optimize either CE or MSE on the ConvNets descried above, we can get an error rate at 0.93%. Replacing the training methods with ANRAT using batch SGD leads to a sharply decreased validation error of 0.66% with a test error at 0.52%. With dropout and ReLU the error rate drops to 0.39%, which is the same with the best results without averaging or data augmentation (Table 6.2) but we only use standard Convnets and simple experimental settings.

Fig. 5.1 (a) shows the progression of training, validation and test errors over 160 training epochs. The errors trained on MSE plateau as it can not train the ConvNets sufficiently and seems like underfit. Using ANRAT, the validation and test errors remain decreasing

[3](1)(Mairal *et al.* 2014);(2)(Lee *et al.* 2014);(3)(LeCun *et al.* 1998);(4)(Ranzato *et al.* 2007);(5)(Ngiam *et al.* 2011);(6)(Ranzato *et al.* 2007);(7)(Poultney *et al.* 2006);(8)(Zeiler & Fergus 2013);(9)(Jarrett *et al.* 2009)

Method[3]	Error %
Convolutional Kernel Networks + L-BFGS-B[(1)]	**0.39**
Deeply Supervised Nets + dropout[(2)]	**0.39**
ConvNets (Lenet-5)[(3)]	0.95
ConvNets + MSE/CE (this section)	0.93
large ConvNets, random feature[(4)]	0.89
ConvNets + L-BFGS[(5)]	0.69
large ConvNets, unsup pretraining[(6)]	0.62
ConvNets, unsup pretraining[(7)]	0.6
ConvNets + dropout[(8)]	0.55
large ConvNets, unsup pretraining[(9)]	0.53
ConvNets + ANRAT (This paper)	0.52
ConvNets + ANRAT + dropout (This paper)	**0.39**

Table 5.1. Test set misclassification rates of the best methods that utilized convolutional networks on the original MNIST dataset using single model.

along with the training error. During training, λ sharply decrease, regulating the tunnel of NRAE to approach the manifold of MSE. Afterward the penalty term becomes significant, force λ to grow gradually while expanding the convex region for higher chance to find the better optimum (Figure 5.1 (b)).

Using the same network architecture described above, we trained two ConvNets using MSE and ANRAT respectively and compare their performance. Fig. 5.1 (a) shows the progression of train, validation and test errors over 160 training epochs. The errors trained on MSE plateau as it can not train the ConvNets sufficiently and underfits. Using ANRAT, the validation and test errors remain decreasing along with the training error. During training, λ sharply decrease, regulating the tunnel of NRAE to approach the manifold of MSE. Afterward the penalty term becomes significant, force λ to grow gradually while expanding the convex region for higher chance to find the better optimum (Fig. 5.1 (b)).

Our next experiment is performed on the CIFAR-10 dataset. We observed significant overfitting using both MSE and ANRAT with the fixed learning rate and batch SGD, so

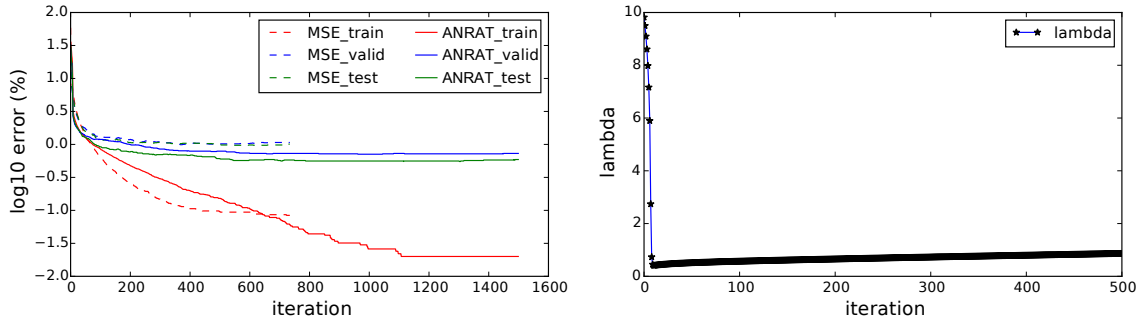

FIG. 5.1. (a). MNIST train, validation and test error rates throughout training with Batch SGD for MSE and ANRAT with l_2 priors. We cross-validated the learning rate and regularization weight on validation set for ConvNets (left). (b). The curve of λ throughout ANRAT training (right).

dropout is applied to prevent the co-adaption of weights and improve generalization. We use a similar network layout as in (Srivastava *et al.* 2014) but with only two convolutional max-pooling layers. The first convolutional layer has 96 feature maps of size 5×5 and max-pooled by 2×2 non-overlapping windows. The second convolutional layer has 128 feature maps with the same convolutional and max-pooling size. The fully connected layer has 500 hidden units. Dropout was applied to all the layers of the network with the probability of retaining a hidden unit being $p = (0.9, 0.75, 0.5, 0.5, 0.5)$ for the different layers of the network. Using batch SGD to optimize CE on the simple configuration of ConvNets + dropout, a test accuracy of 80.6 % is achieved (Krizhevsky, Sutskever, & Hinton 2012). We also reported the performance at 80.58% with MSE instead of CE with the similar network layout. Replacing the training methods with ANRAT using batch SGD gives a test accuracy of 85.15%. This is superior to the results obtained by MSE/CE and unsupervised pretraining. In Table. 5.2, our result with simple setting is shown to be competitive to those achieved by different ConvNet variants.

[4](1)(Zeiler & Fergus 2013);(2)(Srivastava *et al.* 2014);(3)(Goodfellow *et al.* 2013b);(4)(Zeiler & Fergus 2013);(5)(Coates & Ng 2011);(6)(Min Lin 2014);(7)(Lee *et al.* 2014)

Table 5.2. Test accuracy of the best methods that utilized convolutional framework on CIFAR-10 dataset without data augmentation.

Method[4]	Acc %
ConvNets + Stochastic pooling + dropout[(1)]	84.87
ConvNets + dropout +Bayesian hyperopt[(2)]	87.39
ConvNets + Maxout + dropout[(3)]	88.32
Convolutional NIN + dropout[(6)]	89.6
Deeply Supervised Nets + dropout[(7)]	**90.31**
ConvNets + MSE + dropout (this section)	80.58
ConvNets + CE + dropout[(4)]	80.6
ConvNets + VQ unsup pretraining[(5)]	82
ConvNets + ANRAT + dropout (This section)	**85.15**

5.6.2 Results on Multilayer Perceptron

While ConvNets leads the state of the art performance on visual recognition, general Multilayer Perceptron (MLP) is an important framework for generative and discriminative learning. Unsupervised layer-wise pretraining by RBM or auto-encoder demonstrate superb performance boosting on visual recognition on MLP.

On the MNIST dataset, MLPs with unsupervised pretraining has been well studied in recent years, so we select this dataset to compare ANRAT in shallow and deep MLPs with MSE/CE and unsupervised pretraining. For the shallow MLPs, we follow the network layout as in (Gui, Lo, & Peng 2014; LeCun *et al.* 1998) that has only one hidden layer with 300 neurons. We build the stacked architecture and deep network using the same architecture as (Larochelle *et al.* 2009) with 500, 500 and 2000 hidden units in the first, second and third layers, respectively. The training approach is purely batch SGD with no momentum or adaptive learning rate. No weight decay or other regularization technique is applied in our experiments.

Experiment results in Table. 5.3 show that the deep MLP classifier trained by the

ANRAT method has the lowest test error rate (1.45%) of benchmark MLP classifiers with MSE/CE under the same settings. It indicates that ANRAT has the ability to provide reasonable solutions with different initial weight vectors. This result is also better than deep MLP + supervised pretraining or Stacked Logistic Regression networks. We note that the deep MLP using unsupervised pretraining (auto-encoders or RBMs) remains to be the best with test error at 1.41% and 1.2%. Unsupervised pretraining is effective in initializing the weights to obtain a better local optimum. Compared with unsupervised pretraining + fine tuning, ANRAT sometimes still fall into the sightly worse local optima in this case. However, ANRAT is significantly better than MSE/CE without unsupervised pretraining.

Experimental results in Table. 5.3 show that the deep MLP classifier trained by ANRAT method has the lowest test error rate (1.55%) than benchmark MLP classifiers with MSE/CE with the same settings. It indicates that the ANRAT method has the ability to provide reasonable generalization results with different initial weight vectors. This result are also better than deep MLP + supervised pretraining or Stacked Logistic Regression network. However, we note that the deep MLP using unsupervised pretraining (auto-encoders or RBM) is remaining the best with the test error at 1.41% and 1.2%. Unsupervised pretraining is still effective to better initialize the weight to obtain a better local optimum. Compared with unsupervised pretraining + fine tuning, ANRAT sometimes still fall into the sightly worse local optimum in this case. However, ANRAT is significantly better than MSE/CE without unsupervised pretraining.

Interestingly, we do not observe significant advantages with ANRAT in shallow MLPs. Although in early literature, the error rate on shallow MLPs were reported as 4.7% (LeCun *et al.* 1998) and 2.7% with GDC (Gui, Lo, & Peng 2014), both recent papers using CE (Larochelle *et al.* 2009) and our own experiments with MSE can achieve error rate of 1.93% and 2.02%, respectively. Trained by ANRAT, we can have a test rate at 1.94%. This performance is slightly better than MSE, but it is statistically identical to the performance

obtained by CE. [5]. One possible reason is that in shallow networks which can be trained quite well by standard back propagation with normalized initializations, the local optimum achieved with MSE/CE is quite nearly a global optimum or good saddle point. Our result is also corresponding to the conclusion in (Dauphin *et al.* 2014), in which Dauphin et al. extend previous findings on networks with a single hidden layer to show theoretically and empirically that most badly suboptimal critical points are saddle points. Even with better convexity property, ANRAT is as good as MSE/CE in shallow MLPs. However, we find that the problem of poor local optimum becomes more manifest in deep networks. It is easier for ANRAT to find a way towards the better optimum near the manifold of MSE. For the sake of space, please refer to supplemental materials for the results on the shallow Denoised Auto-encoder. The conclusion is consistent that ANRAT performs better when attacking more difficult learning/fitting problems. While ANRAT is slightly better than CE/MSE + SGD on DA with uniform masking noise, it achieves a significant performance boost when Gaussian block masking noise is applied.Moreover, the exponential form of NRAE helps to alleviate vanishing of gradient also help training deep networks.

5.6.3 Results on Denoised Auto-encoder

Our last experiments focus on generative learning instead of discriminative learning. In (Vincent *et al.* 2010), Denoised Auto-encoder (DA) is well studied to show the ability to learn useful representations. We were considering if ANRAT is likely to lead to the learning of feature detectors that detect better structure in the input patterns with more sufficient training.

On MNIST dataset, we use a single-layer DA with the local masking noise at the

[5]in (Larochelle *et al.* 2009), the author do not report their network settings of the shallow MLP + CE, which may differ from 784-300-10.

[6](1)(Larochelle *et al.* 2009);(2)(LeCun *et al.* 1998);(3)(Gui, Lo, & Peng 2014)

Table 5.3. Test error rate of deep/shallow MLP with different training techniques.

Method[6]	Error %
Deep MLP + supervised pretraining$^{(1)}$	2.04
Stakced Logistic Regression Network$^{(1)}$	1.85
Stacked Auto-encoder Network$^{(1)}$	1.41
Stacked RBM Network$^{(1)}$	**1.2**
Shallow MLP + MSE$^{(2)}$	4.7
Shallow MLP + GDC$^{(3)}$	2.7 ± 0.03
Shallow MLP + MSE (this section)	2.02
Shallow MLP + ANRAT (this section)	1.94
Shallow MLP + CE$^{(1)}$	**1.93**
Deep MLP + CE$^{(1)}$	2.4
Deep MLP + MSE (this section)	1.91
Deep MLP + ANRAT (this section)	**1.45**

levels from 30% to 60%. That is, we randomly set the pixels to be 0 evenly. Another noise paradigm is Gaussian block noise. We randomly select centroids to span masking blocks in terms of each centroid with a Gaussian distribution. The noise level differs from 33.3% to 52.6% accordingly. Figure. 5.2 shows three examples of different corruption level. We train the DA using ANRAT and MSE with pure batch SGD with fixed learning rate among $\{1, 0.1, 0.01\}$ for 1000 epochs and report the best reconstruction MSE.

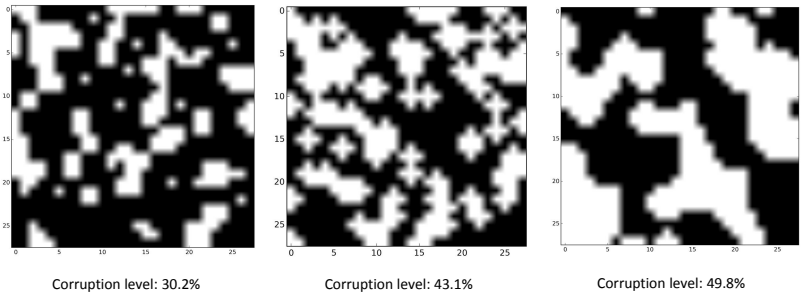

FIG. 5.2. Example maps of Gaussian block masking.

As in the shallow network, we found the advantage of ANRAT is statistically significant in DA but the difference is very small when using masking noise (Figure. 5.4). Although masking makes it harder to reconstruct original images, the strong spatial correlations are still existed between masked pixels and the "good" pixels nearby, thus helps DA to learn the generative distribution from the conditional probability with respect to the manifold of the added noise. The local optimum found by both SGD training with MSE and ANRAT is nearly optimal. When applying Gaussian block noise as shown in Figure. 5.2, masking blocks cover several patches instead of scattered points. It is much harder to learn the spatial correlation within the masking areas. On this tougher task, ANRAT performs much better than using batch SGD on MSE. In Figure. 5.4, the reconstruction MSE using ANRAT on each image is 1.078, which is 24.09% lower than 1.432 obtained by batch SGD on MSE.

Table 5.4. Reconstruction MSE of DA output trained by ANRAT and MSE error criterion at different masking level using random masking or Gaussian block masking. The mean and standard deviation over 20 times running are reported.

Corruption Level	MSE	ANRAT
30%	4.552 (0.006)	4.348 (0.005)
40%	5.697 (0.007)	5.57 (0.005)
50%	7.112 (0.005)	7.071 (0.007)
60%	9.245 (0.008)	9.1 (0.008)
Gaussian Isotropic Masking (33.3% - 52.6%)	1.432 (0.0002)	1.078 (0.009)

In Figure. 5.3 (a), the curve of training error using ANRAT penetrates across the curve trained by MSE after several epochs and is consistently staying lower. Note that NRAE and L_P norm can be compared since they are homeomorphism and in the same magnitude, but there is still a "gap" between them. So, λ is still forced to fluctuate, gradually grow larger to find more optimal tunnel towards the better optimum (Figure 5.3 (b)). This keeps ANRAT searching for better results near the manifold of MSE through a gradual convexing

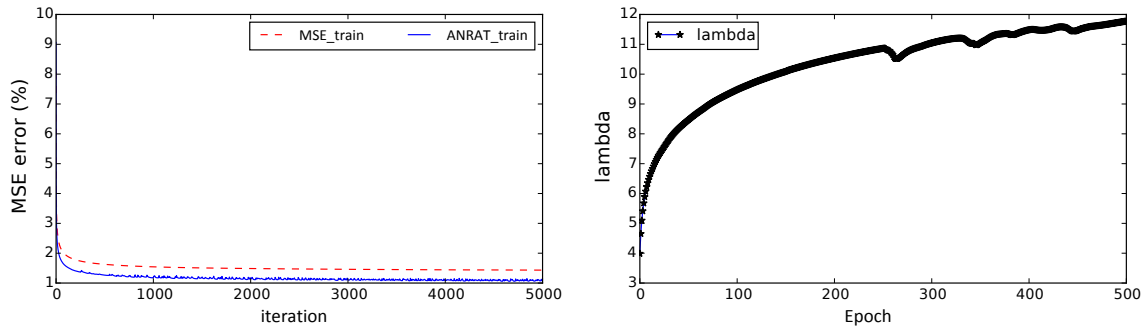

FIG. 5.3. (a). Reconstruction MSE throughout training by Batch SGD for MSE and ANRAT (left). Gaussian block noise is applied at input layer. (b). The curve of λ throughout ANRAT (right).

on its convexity region. In Figure. 5.4, DA using ANRAT learned a lightly more diverse features than SGD on MSE when uniform masking is applied. In the case of Gaussian block masking, DA learned more clear meaningful features with the help of ANRAT.

5.7 Conclusions and Outlook

In this section, we introduce a novel approach, Adaptive Normalized Risk-Averting Training (ANRAT), to help train deep neural networks. Theoretically, we prove the effectiveness of Normalized Risk-Averting Error on its arithmetic bound, global convexity and local convexity lower-bounded by standard L_p-norm error when convexity index $\lambda \geq 1$. By analyzing the gradient on λ, we explained the reason why using back propagation on λ works. The experiments on deep/shallow network layouts demonstrate comparable or better performance with the same experimental settings among pure ConvNets and MLP + batch SGD on MSE and CE (with or without dropout). Other than unsupervised pre-training, it provides a new perspective to address the non-convex optimization strategy in DNNs. Theoretically, we prove the effectiveness of Normalized Risk-Averting Error on its arithmetic bound, global convexity and local convexity lower-bounded by standard L_p-

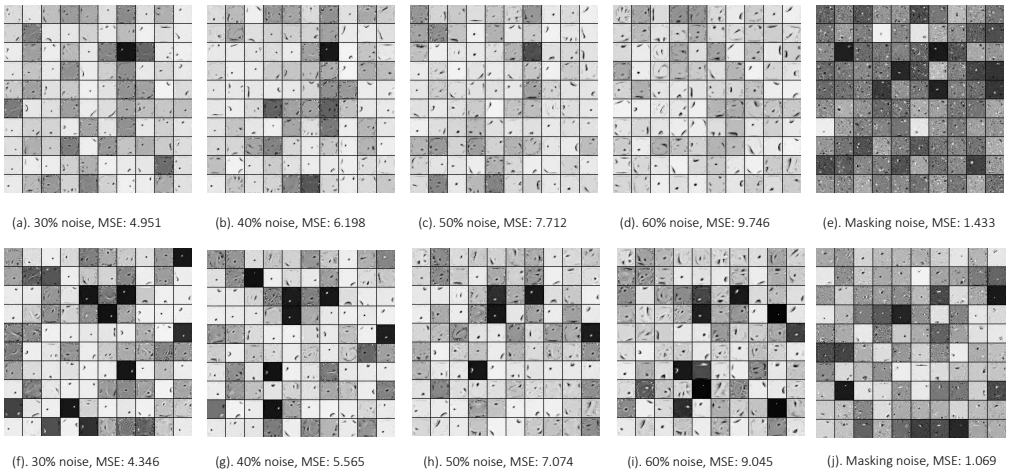

FIG. 5.4. 100 feature maps learned by Denoised Auto-encoder using ANRAT and batch SGD on MSE. Different corruption level of uniform masking and Gaussian block masking are applied at the input layer.

norm error when convexity index $\lambda \geq 1$. By analyzing the gradient on λ, we explained the reason why using back propagation on λ works.

Empirically, we compare ANRAT with batch SGD training on MSE/CE and unsupervised pretraining using pure ConvNets on MNIST and CIFAR-10 datasets. ANRAT achieves comparable or better performance with the same experiment settings among those use pure ConvNets + batch SGD and/or dropout. An overview of the comparisons among the best results obtained from different training methods and model variants is also provided to validate the performance of ANRAT is among the state of the art. Except for ConvNets, we evaluate ANRAT in shallow/deep multilayer perceptrons. Although the advantage of ANRAT is not significant against CE/MSE + SGD in shallow neural networks, it overtakes CE/MSE + SGD in deep neural networks and approaches to the performance achieved by unsupervised pretraining such as RBM and auto-encoder. To better understand ANRAT on generative models, experiments on shallow Denoised Auto-encoders are also performed

with uniform masking noise and Gaussian block masking noise at the input layer. While ANRAT is slightly better than CE/MSE + SGD on DA with the uniform masking noise, it achieves significant performance boosting when Gaussian block masking noise is applied.

Finally, while these early results are very encouraging, clearly further research is warranted to address the questions that arise from non-convex optimization in deep neural networks. It is preliminarily showed that in order to generalize to a wide array of tasks, unsupervised and semi-supervised learning using unlabeled data is crucial. One interesting future work is to take advantage of unsupervised/semi-supervised pretraining with the non-convex optimization methods to train deep neural networks by finding the nearly global optimum. Another crucial question is to guarantee the generalization capability by preventing overfitting. Finally, we are quite interested in generalizing our approach to recurrent neural networks. We leave as future work any performance improvement on benchmark datasets by considering the cutting-edge approach to improve training and generalization performance such as Bayesian hyperparameter optimization, network layout variants, Maxout or SFN, etc.

Chapter 6

ROBUST STATISTICS — ADAPTIVE NORMALIZED ANOMALY-AVERTING TRAINING FOR NEURAL NETWORKS

6.1 Introduction

So far, we only focused on the situations when $\lambda > 0$. However, when $\lambda < 0$ and is small enough, from Equation. 6.7 the Hessian matrix is still definite positive to guarantee the error function is convex. The credits should be given to Lo who firstly proposed such an idea in (Lo & Bassu 2001) and we consistently investigate Lo's early work with different naming systems. When $\lambda \to -\infty$, we called the error estimator Anomaly-Averting Estimator (AAE, in Lo's work, he called it Risk-seeking Error). AAE not only ensures the convexity but also retains high robustness to outliers. Note that when λ is largely negative, we still have the register underflow problem. The same normalized pipeline in the previous section also works on AAE. In this section, we will explore the optimality and robustness of Normalized Anomaly-Averting Estimator (NAAE) by pushing λ to $-\infty$. A more specific theoretical statements are provided in l_2 norm to clarify its connection with square loss. Robustness is widely required in control theory and dynamical systems design. Thus, we also evaluate NAAE on both function approximation problems and machine learning tasks.

6.2 Background

Function approximation has many applications in science and engineering, such as machine learning, pattern recognition, signal processing and control theory. As universal approximators (Hornik, Stinchcombe, & White 1989), neural networks (NNs) often deliver very good performance and have become the standard for several machine learning tasks given their recent success in various applications (Jaderberg *et al.* 2015; Lee *et al.* 2014; Szegedy *et al.* 2015; Karpathy & Fei-Fei 2015).

Error-free data are rarely provided in applications. Instead, data are usually contaminated by noise and outliers. Noise reflects inaccuracies in observations and the stochastic nature of the underlying process. NNs deal with such noise quite efficiently by optimizing the minimum squared error (MSE) or cross entropy (CE) between the observed and predicted values/labels with ℓ_1/ℓ_2 regularization, sparsity or adversary resistant learning (Sra, Nowozin, & Wright 2012). Recent developments in computer vision take advantage of manually added noise and distortion to further enhance the data (Krizhevsky, Sutskever, & Hinton 2012; Gan *et al.* 2015) or to capture reverse conditional probability within generative models (Bengio *et al.* 2013; Goodfellow *et al.* 2014). Outliers can be arbitrary, unbounded, and not from any specific distribution, reflecting sudden abnormal changes or a mixture of rare but different phenomena. It has been noted that outliers in routine data are infrequent but heavily impact even high dimensional data (Aggarwal & Yu 2001; Kriegel, Zimek, & others 2008). When fitting a model or learning an objective function with rare but significant outliers whose distribution and impact are both unknown, they need to be identified and eliminated to get rid of their effect. Otherwise, the model overfits or underfits easily if training procedure is sufficient or not. For example, in the image/video classification task, some images or videos may be corrupted unexpectedly due to the error of sensors or severe occlusions of objects. Such outliers can skew parameter estimation

severely and hence destroy the performance of the learned model. Alternatively, outliers are sometimes supposed to be examined closely, as they may be of interest themselves given the sample applications of intrusion or fraud detection.

The maximal likelihood principle is not robust against outliers among different approximation and learning models like logistic regression (LR) and (Feng *et al.* 2014) generalized linear models (Künsch, Stefanski, & Carroll 1989). High breakdown methods have been developed including least median of squares (Rousseeuw 1984), least trimmed squares (Rousseeuw 1984) and most recently trimmed inner product methods for linear regression. Several works have investigated multiple approaches to robustify LR (Pregibon 1982; Stefanski, Carroll, & Ruppert 1986; Tibshirani & Manning 2013). The majority of them are M-estimator based: minimizing a complicated and more robust loss function than the standard loss function (negative log-likelihood). A robust backpropagation algorithm is proposed to optimize a robust estimator derived from the M-estimator to train NNs (Chen & Jain 1994). (Liano 1996) used M-estimators to study the mechanism by which outliers affect the resulting NNs. The majority of the above work that is M-estimator based needs to predefine a threshold for determining the degree of contaminated data to achieve robustness to outliers. Besides, few of them discuss optimality while achieving the robustness. Here, by optimality, we refer to its ability to find a nearly global or better local optimum while achieving robustness to outliers by getting rid of large deviation among the data.

Our work is largely inspired by the exponentialized error criteria (Jacobson 1973; Whittle 1990) and its recent variations and modifies exponentialized error with two different training methods (Lo 2010; Gui, Lo, & Peng 2014; Wang, Oates, & Lo 2015). Our main contribution is to implement with our adaptive training approach and verify the idea that generalizing this estimator by pushing the robust-optimal (RO) index λ to $-\infty$ to obtain the robustness to the largely deviated outliers while preserving the optimality by the expansion of the convexity regions in the Hessian matrix. As a general error estimator, we

provide a quantitative analysis and validate its effectiveness on three function fitting and one visual recognition tasks.

6.3 Normalized Anomaly-Avering Estimator

Given standardized training samples $\{\boldsymbol{X}, y\} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), ..., (\boldsymbol{x}_m, y_m)\}$, $f(\boldsymbol{x}_i, \boldsymbol{W})$ is the learned model with parameters W. The loss function of mean squared loss (MSE) is defined as:

$$l_2(f(\boldsymbol{x}_i, \boldsymbol{W}), y_i) = \frac{1}{m}\sum_{i=1}^{m}(f(\boldsymbol{x}_i, \boldsymbol{W}) - y_i)^2$$

As a modified exponentialized error of MSE, the Anomaly-Averting Estimator (AAE) is defined as

$$AAE = \frac{1}{m}\sum_{i=1}^{m} e^{\lambda(f(\boldsymbol{x}_i, \boldsymbol{W}) - y_i)^2} \quad (\lambda < 0) \tag{6.1}$$

The only difference between the AA estimator and RAE in (Lo 2010) is that λ stays negative, which makes it serve as the robust-optimal (RO) index to control both the degree of optimality and robustness to outliers. Note that when $\lambda \to -\infty$, it is not bounded and suffers from less stability, exponential magnitude and arithmetic overflow when using gradient descent in implementations. We define the Normalized Anomaly-Averting Estimator (NAAE) as:

$$\begin{aligned} & NAAE(f(\boldsymbol{x}_i, \boldsymbol{W}), y_i) \\ &= \frac{1}{\lambda}\log AAE(f(\boldsymbol{x}_i, \boldsymbol{W}), y_i) \\ &= \frac{1}{\lambda}\log\frac{1}{m}\sum_{i=1}^{m} e^{\lambda(f(\boldsymbol{x}_i, \boldsymbol{W}) - y_i)^2} (\lambda < 0) \end{aligned} \tag{6.2}$$

6.3.1 Robustness

NAAE is a bounded estimator. The bounds exert not only the stable property as an estimator, but also an interesting dynamics to control the degree of robustness.

Theorem 5. *$NAAE(f(\boldsymbol{x}_i, \boldsymbol{W}), y_i)$ is bounded between the minimal and maximal squared errors.*

Proof. Let

$$\begin{aligned} \alpha_i &= f(\boldsymbol{x}_i, \boldsymbol{W}) - y_i \\ \alpha_{min}^2 &= \min\{\alpha_i^2, i = 1, ..., m\} \\ \alpha_{max}^2 &= \max\{\alpha_i^2, i = 1, ..., m\} \\ \beta_i &= e^{\lambda(\alpha_i^2 - \alpha_{min}^2)} \end{aligned}$$

Then we can re-write Eqn. 6.2 as

$$\begin{aligned} & NAAE \\ &= \frac{1}{\lambda} \log \frac{1}{m} e^{\lambda \alpha_{min}^2} \sum_{i=1}^{m} \beta_i \\ &= -\frac{1}{\lambda} \log m + \alpha_{min}^2 + \frac{1}{\lambda} \log \sum_{i=1}^{m} \beta_i \end{aligned} \tag{6.3}$$

let $G = \alpha_{max}^2 - \alpha_{min}^2$ to be the maximal margin of the sample squared errors, considering $\lambda < 0$, $e^{\lambda G} \leq \beta_i \leq 1$, so

$$\frac{1}{\lambda} \log m \leq \frac{1}{\lambda} \log \sum_{i=1}^{m} \beta_i \leq \frac{1}{\lambda} \log m + G$$

We can derive the bounds of NAAE as

$$\alpha_{min}^2 \leq NAAE \leq \alpha_{max}^2$$

□

The above conclusions indicate that squared errors regulate the bounds of NAAE. The next theorem shows that λ controls NAAE to perform interactively with MSE and its relationship to the minimin operator.

Theorem 6. *when* $\lambda \to -\infty$, $NAAE(f(\boldsymbol{x}_i, \boldsymbol{W}), y_i)$ *approaches to an minimin operator; if* $\lambda \to 0$, $NAAE(f(\boldsymbol{x}_i, \boldsymbol{W}), y_i) \to$ ***MSE.***

Proof. Given $\lambda < 0$, we consider the objective function

$$W = \arg\min_{\boldsymbol{W}} NAAE(f(\boldsymbol{x}_i, \boldsymbol{W}), y_i)$$

Considering equation 6.3, it is easy to see that when $\lambda \to -\infty$, the objective function becomes

$$W = \arg\min_{\boldsymbol{W}} \alpha_{min}^2$$

So a large λ controls the NAAE to approach a minimin operator. Next we consider the

situation when $\lambda \to 0$ [1].

$$
\begin{aligned}
& NAAE(f(\boldsymbol{x}_i, \boldsymbol{W}), y_i) \\
&= \frac{1}{\lambda} \log \frac{1}{m} \sum_{i=1}^{m} e^{\lambda(f(\boldsymbol{x}_i, \boldsymbol{W}) - y_i)^2} \\
&= \frac{1}{\lambda} \log(1 + \frac{1}{m} \sum_{i=1}^{m} \lambda(f(\boldsymbol{x}_i, \boldsymbol{W}) - y_i)^2 + O(\lambda^2)) \\
&= \frac{1}{m} \sum_{i=1}^{m} (f(\boldsymbol{x}_i, \boldsymbol{W}) - y_i)^2
\end{aligned} \tag{6.4}
$$

□

More rigid proofs that can be generalized to L_p-norm error are given in (Lo 2010).

Equations 6.3 and 6.4 explain how λ controls the robustness level. When $\lambda \to 0$, NAAE approaches MSE with a breakdown point of 0%, meaning that a single observation can change it arbitrarily. Further, it is highly influenced by outliers. When $\lambda \to -\infty$, NAAE performs like a minimin operator to address only the minimal error. Outliers can be eliminated if they largely deviate from the objective space, or say manifold, so that NAAE concentrates on the smaller errors and ignores the impact of those large errors. But it is still vulnerable to noise and outliers that are much closer to the objective manifold than the majority of the clean data.

When λ changes between 0 and $-\infty$, the exponential sum of the squared error offsets the minimin operator to further address the situations when the outliers are close to the objective manifold. By adjusting λ, NAAE is able to handle both large deviations and false positive samples, which helps to achieve robustness to different type of outliers.

[1] Consider the rules $e^x = 1 + x + \frac{x^2}{2} + \cdots (x \to 0)$ and $\log x = x - \frac{x^2}{2} + \cdots (x \to 0)$

6.3.2 Optimality

To prove the optimality or NAAE, we start with AAE and generalize our conclusion to NAAE. For clarity, we write the matrix derivatives in functional form. In our proof we use the quadratic form. The Jacobian matrix of Eqn. 6.1 is

$$\begin{aligned} J(\boldsymbol{W}) = \quad & \tfrac{2\lambda}{m}\textstyle\sum_{i=1}^{m} e^{\lambda(f(\boldsymbol{x}_i,\boldsymbol{W})-y_i)^2} \\ \times \quad & (f(\boldsymbol{x}_i,\boldsymbol{W})-y_i)\tfrac{\partial f(\boldsymbol{x}_i,\boldsymbol{W})}{\partial \boldsymbol{W}} \quad (\lambda<0) \end{aligned} \tag{6.5}$$

The Hessian matrix of Eqn. 6.1 is

$$H(\boldsymbol{W}) = \tfrac{2}{m}\textstyle\sum_{i=1}^{m} e^{\lambda(f(\boldsymbol{x}_i,\boldsymbol{W})-y_i)^2}\{\lambda\tfrac{\partial f(\boldsymbol{x}_i,\boldsymbol{W})}{\partial \boldsymbol{W}}^2 \tag{6.6}$$

$$+ \quad 2\lambda^2(y_i - f(\boldsymbol{x}_i,\boldsymbol{W}))^2\tfrac{\partial f(\boldsymbol{x}_i,\boldsymbol{W})}{\partial \boldsymbol{W}}^2 \tag{6.7}$$

$$+ \quad \lambda(y_i - f(\boldsymbol{x}_i,\boldsymbol{W}))\tfrac{\partial f^2(\boldsymbol{x}_i,\boldsymbol{W})}{\partial \boldsymbol{W}^2}\} \tag{6.8}$$

We assume the training sample is standardized and the error space is a unit sphere $|\mathcal{R}^n \leq 1|$ (that is, $|y_i - f(\boldsymbol{x}_i,\boldsymbol{W}| < 1$). Because AAE has the sum-exponential form, its Hessian matrix is tuned exactly by the RO index λ. The following theorem indicates the relation between the convexity index and its convexity region.

Theorem 7. *Given the Anomaly-Averting Error criterion AAE, which is twice continuous differentiable. $J(\boldsymbol{W})$ and $H(\boldsymbol{W})$ are the corresponding Jacobian and Hessian matrix. As $\lambda \to \pm\infty$, the convexity region monotonically expands to the entire parameter space except for the subregion $S := \{W \in \mathcal{R}^n | rank(H(\boldsymbol{W})) < n, H(\boldsymbol{W} < 0)\}$.*

Proof. Both $\frac{\partial f(\boldsymbol{x}_i,\boldsymbol{W})^2}{\partial \boldsymbol{W}}$ and $\lambda^2(y_i - f(\boldsymbol{x}_i,\boldsymbol{W}))^2$ are positive semi-definite, matrix 6.7 is semi-positive definite, but Eqn. 6.6 and 6.8 may be indefinite. Let $\alpha_i(\boldsymbol{W}) = f(\boldsymbol{x}_i,\boldsymbol{W}) - y_i$,

We rewrite Eqn. 6.7 in quadratic form:

$$
\begin{aligned}
&= p\lambda^2 \alpha_i(\boldsymbol{W})^T \frac{\partial f(\boldsymbol{x}_i, \boldsymbol{W})^T}{\partial \boldsymbol{W}} \cdot \frac{\partial f(\boldsymbol{x}_i, \boldsymbol{W})}{\partial \boldsymbol{W}} \alpha_i(\boldsymbol{W}) \\
&= p\lambda^2 \alpha_i(\boldsymbol{W})^T Q \Lambda Q^T \alpha_i(\boldsymbol{W}) \\
&= p\lambda^2 S(\boldsymbol{W})^T \Lambda S(\boldsymbol{W}) \qquad (6.9)
\end{aligned}
$$

$\Lambda = diag[\Lambda_1, \Lambda_2, ..., \Lambda_m]$. 6.6 is positive semi-definite.

If $S(\boldsymbol{W})$ is a full-rank matrix, then Eqn. 6.9 is positive definite. When $\lambda \to \pm\infty$, the eigenvalues Λ becomes dominant in the leading principal minors (as well as eigenvalues) of the Hessian matrix to make $H(\boldsymbol{W})$ monotonically increasing with λ. Assume $P_\lambda := \{W \in \mathcal{R}^n | H_(\boldsymbol{W} > 0)\}$, we have $P_{\lambda_1} \in P_{\lambda_2}$ when $\lambda_1 < \lambda_2$. Considering Eqn. 6.6, 6.7 and 6.8 are all bounded, $\exists\, \psi(W)$, s.t. $H(\boldsymbol{W}) > 0$ when $|\lambda| > \psi(W)$.

When $S(\boldsymbol{W})$ is not a full-rank matrix, the determinant of all its $\binom{n}{k}$ submatrices is 0. Thus, in the subregion $S := \{W \in \mathcal{R}^n | rank(H(\boldsymbol{W})) < n, H(\boldsymbol{W} < 0)\}$, there is no parameters satisfied the convexity conditions ($\bigcup P_\lambda$). □

Theorem 7 states that when the RO index λ decrease to infinity, the convexity region in the parameter space of AAE expands monotonically to the entire space except the intersection of a finite number of lower dimensional sets. The number of sets increases rapidly as the number m of training samples increases. Roughly speaking, large $|\lambda|$ and m cause the size of the convexity region to grow larger in the error space of AAE.

When $\lambda \to -\infty$, the error space can be perfectly stretched to be strictly convex, thus avoiding the local optimum to find a global optimum. The following theorem states the quasi-convexity of NAAE.

Theorem 8. *Given a parameter space* $\{W \in \mathcal{R}^n\}$*, Assume* $\exists\, \psi(W)$*, s.t.* $H(\boldsymbol{W}) > 0$ *when* $|\lambda| > \psi(W)$ *to guarantee the convexity of* $AAE(f(\boldsymbol{x}_i, \boldsymbol{W}), y_i)$*. Then,* $NAAE(f(\boldsymbol{x}_i, \boldsymbol{W}), y_i)$

*is **quasi-convex** and share the same local and global optima with* $AAE(f(\boldsymbol{x}_i, \boldsymbol{W}), y_i)$.

Proof. If $AAE(f(\boldsymbol{x}_i, \boldsymbol{W}), y_i)$ is convex, it is quasi-convex. The $\log$ function is monotonically increasing, so the composition $\log AAE(f(\boldsymbol{x}_i, \boldsymbol{W}), y_i)$ is quasi-convex. [2]

$\log$ is a strictly monotone function and $NAAE(f(\boldsymbol{x}_i, \boldsymbol{W}), y_i)$ is quasi-convex, so it shares the same local and global optima with $AAE(f(\boldsymbol{x}_i, \boldsymbol{W}), y_i)$. □

The above theorem states that the convexity region of NAAE is consistent with AAE. To interpret this statement in another perspective, the $\log$ function is a strictly monotone function. Even if AAE is not strictly convex, NAAE still shares the same local and global optima with RAE. If we define the mapping function $f : AAE \rightarrow NAAE$, it is easy to see that f is bijective and continuous. Its inverse map f^{-1} is also continuous, so that f is an open mapping. Thus, it is easy to prove that the mapping function f is a homeomorphism to preserve all the topological properties of the given space.

The above theorems state the consistent relations among NAAE, AAE and MSE. It is proven that the greater the RO index $|\lambda|$, the larger the convex region is. Intuitively, increasing $|\lambda|$ creates tunnels for a local-search minimization procedure to travel through to a good local optimum. While NAAE preserves the robustness to MSE, theorem 9 provides insights into when NAAE is optimal than MSE.

Theorem 9. *Given training samples* $\{\boldsymbol{X}, y\} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), ..., (\boldsymbol{x}_m, y_m)\}$ *and the model* $f(\boldsymbol{x}_i, \boldsymbol{W})$ *with parameters* W. *Define the deviation* $\alpha_i = f(\boldsymbol{x}_i, \boldsymbol{W}) - y_i$, *when* $\alpha_i \rightarrow 1$ *and* $\lambda \leq -1$, *both* $AAE(f(\boldsymbol{x}_i, \boldsymbol{W}), y_i)$ *and* $NAAE(f(\boldsymbol{x}_i, \boldsymbol{W}), y_i)$ *always have a larger convexity region to commit a higher chance for the better local optima than MSE.*

[2] Because the function f defined by $f(x) = g(U(x))$ is quasi-convex if the function U is quasiconvex and the function g is increasing.

Proof. Let $h(\boldsymbol{W})$ denotes the Hessian matrix of MSE (Eqn. 6.1),

$$\begin{aligned} h(\boldsymbol{W}) &= \frac{2}{m}\sum_{i=1}^{m}\{\alpha_i(\boldsymbol{W})^2\frac{\partial f(\boldsymbol{x}_i,\boldsymbol{W})^2}{\partial \boldsymbol{W}} \\ &+ \alpha_i(\boldsymbol{W})\frac{\partial f^2(\boldsymbol{x}_i,\boldsymbol{W})}{\partial \boldsymbol{W}^2}\} \end{aligned} \tag{6.10}$$

Since $\lambda \leq -1$, let $diag_{eig}$ denote the diagonal matrix of the eigenvalues from SVD decomposition. $\succeq$ here means 'element-wise greater'. When $A \succeq B$, each element in A is greater than B. Then we have

$$\begin{aligned} &diag_{eig}[H(\boldsymbol{W})] \\ &\succeq \frac{2}{m}\{(2\lambda^2\alpha_i^2+\lambda)\frac{\partial f(\boldsymbol{x}_i,\boldsymbol{W})^2}{\partial \boldsymbol{W}} - \lambda\alpha_i\frac{f^2(\boldsymbol{x}_i,\boldsymbol{W})}{\partial \boldsymbol{W}^2}\} \\ &\succeq \alpha_i^2\frac{f(\boldsymbol{x}_i,\boldsymbol{W})^2}{\partial \boldsymbol{W}} + \alpha_i\frac{f^2(\boldsymbol{x}_i,\boldsymbol{W})}{\partial \boldsymbol{W}^2} \\ &\succeq diag_{eig}[h(\boldsymbol{W})] \end{aligned}$$

Briefly, when the standard deviation is large (approaches 1) and $\lambda \leq -1$, $AAE(f(\boldsymbol{x}_i,\boldsymbol{W}),y_i)$ always has larger convexity regions than MSE to better enable escape of local minima. Because $NAAE(f(\boldsymbol{x}_i,\boldsymbol{W}),y_i)$ is quasi-convex, sharing the same local and global optimum with $AAE(f(\boldsymbol{x}_i,\boldsymbol{W}),y_i)$, the above conclusions are still valid. □

The expansion of the convexity region in the Hessian matrix enables NAAE to find a better local optima than MSE. This property guarantees higher probability to escape poor local optima. In the worst case, NAAE will perform as good as MSE if the convexity region shrinks as the RO index λ increase to approach 0 or the local search deviates from the "tunnel" of convex regions.

6.3.3 Control Robustness and Optimality

NAAE performs robustly while preserving the optimality by the expansion of its convex region in its Hessian matrix, the RO index λ controls both the robustness and optimality of the estimator simultaneously. When λ is a large negative number, NAAE works like a minimin operator to avoid large deviations incurred by the outliers. Meanwhile, its Hessian matrix has a larger convex region to facilitate seeking a better optimum. If λ approaches 0, a compensation from the exponential sum of the squared errors offsets the impact of the minimin operator to further focus on the larger overview. This will help to address the small deviation from other noise. NAAE performs more like MSE to grant a smooth and stable optimization procedure.

6.4 Update Strategies

As λ controls the degree of both robustness and optimality, the initialization and update strategy impacts the learning performance. Generally, large λ is preferable at the start. Different update strategies lead to three major learning methods.

6.4.1 Fixed-λ

Using a fixed large λ to optimize the exponential estimator is reported in (Lo, Gui, & Peng 2012). For NAAE, a fixed λ consistently controls the estimator at a specific level of the offset towards the minimin operator. The weights are updated using gradient descent. It is simple and work well to approximate the lower dimensional functions with outliers. When attacking the learning tasks in higher dimensions such as visual recognition, the large plateau and unstable learning procedure can lead to a failure in training (Wang, Oates, & Lo 2015; Gui, Lo, & Peng 2014).

6.4.2 Gradual Deconvexfication

(Lo, Gui, & Peng 2013) proposed the Gradual Deconvexification (GDC) approach to alleviate the difficulty in finding a good value of λ for training. GDC starts with a very large λ while recording the values of the objective function before and after a pre-selected number of training epochs. If the performance converges, GDC flags the current training condition as stagnant and performs a deconvexification by reducing the current λ with a percentage r. After that, the training continues and repeats the deconvexification if necessary until a satisfied training error is achieved. If $|\lambda|$ keeps coming down to below 1, the training will set $\lambda = 0$, which actually converts to the training with MSE. All other weights are updated using gradient descent.

GDC works in a similar way as the decreasing-learning-rate approach. Training with GDC avoids fixing the initial λ during training, as the deconvexification gradually decreases λ to avoid bad optima and vulnerable extreme estimator and eventually obtain a reasonable optimum while achieving robustness. As reported in (Gui, Lo, & Peng 2014), GDC is much slower than using MSE alone. The update strategy relies on the per-defined convergence thresholds and dropping rate.

6.4.3 Adaptive Training

(Wang, Oates, & Lo 2015) proposed a novel learning method to training with RO index λ, called the Adaptive Normalized Risk-Avering Training approach (adaptive training). Instead of manually tuning λ like GDC , they learn λ adaptively through error backpropagation by considering λ as a parameter instead of a hyperparameter. The learning procedure is standard batch gradient descent.

$$\frac{dl(W,\lambda)}{d\lambda} \approx \frac{q}{\lambda}(L_P\text{-norm error} - E) \tag{6.11}$$

As shown in the above equation, the gradient on λ is approximately the difference between the exponentialized estimator E (e.g., NAAE in this paper) and the standard L_p-norm error. Considering $\lambda < 0$ as for NAAE. When E is larger, λ decreases to enlarge the convexity region, facilitating the search in the error space for better optima. When E is smaller, the learned parameters are seemingly going through the optimal "tunnel" for better optima. λ then increases and helps the weights not deviate far from the manifold of the standard L_p-norm error to make the error space stable without large plateaus. We note that adaptive training also adjusts λ to control the robustness level in a similarly 'smart' way. If E is larger than the L_p-norm error, λ decreases to further ignore more samples with large deviations to let the learning function gain a broader overview rather than focusing on the outliers. When E is small, outliers with large deviation are almost eliminated. λ increases to approach 0, making the learning procedure more accurate and stable by considering the effect of more training samples rather than the extreme value. The adaptive training approach has more flexibility. It keeps searching the error space near the manifold of the L_p-norm error to find better optima in a way of competing with, and at the same time relying on, the standard L_p-norm error space.

We consider the fixed λ approach and adaptive training as the update strategies for the RO index λ in our experiments. For Adaptive training, the final loss function is

$$l(W,\lambda) = \frac{1}{\lambda}\log\frac{1}{m}\sum_{i=1}^{m} e^{\lambda(f(\boldsymbol{x}_i,\boldsymbol{W})-y_i)^2} + a|\lambda|^{-1} \tag{6.12}$$

The penalty weight a controls the convergence speed by penalizing small $|\lambda|$. Smaller

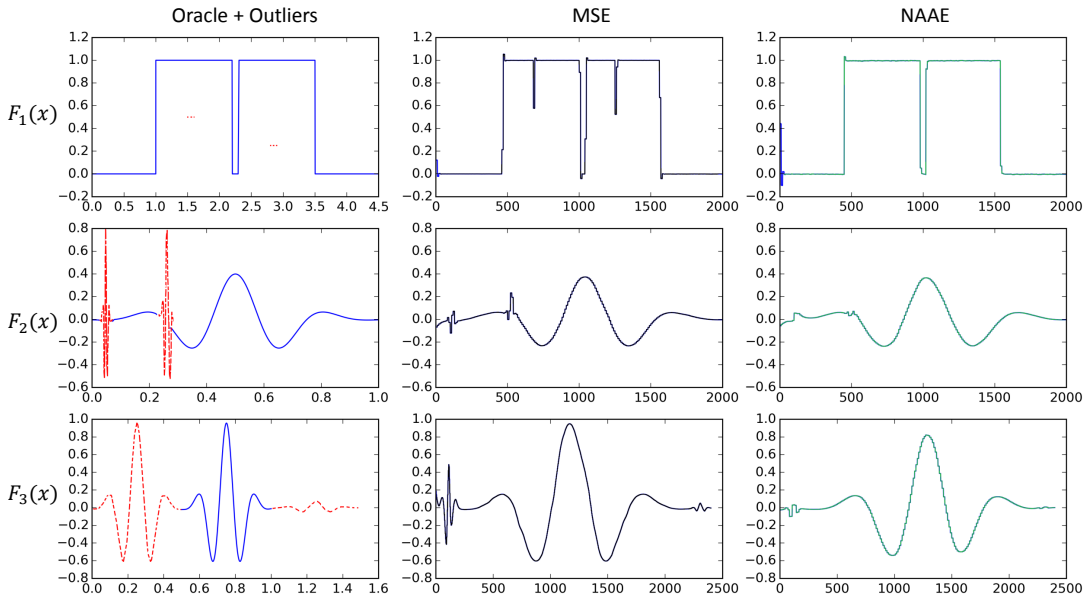

FIG. 6.1. Function approximation using MSE and NAAE on the training set. Three columns are (a). oracle function with outliers (L), (b). approximation results using MSE (M) and (c). NAAE (R). In column (a), the blue lines plot the oracle function, the red dots plot the outliers.

a emphasizes tuning λ to allow faster convergence speed between NAAE and MSE. Larger a forces larger $|\lambda|$ for a better chance to find a better optimum while skipping the outliers, but runs the risk of plateaus and deviating far from the stable error space.

6.5 Experiments and Analysis

As an error estimator, we test NAAE on the function approximation and digits recognition tasks with two different neural networks architectures, multiple layer perceptrons (MLPs) and convolutional networks (ConvNets), respectively. We limit all the experiments in very simple settings, using gradient descent with batch gradient descent and weight de-

cay or dropout (Srivastava *et al.* 2014).

6.5.1 Function Approximation

For function approximation tasks, we design three typical non-convex functions with different types of outliers to test the robustness and optimality achieved by NAAE.

1). The notch function is defined by

$$f_1(x) = \begin{cases} 0 & \text{if} \quad x \in [0, 1.0] \bigcup [2.2, 2.3] \bigcup [3.5, 4.5] \\ 1 & \text{otherwise} \end{cases}$$

The outliers in the middle range are generated by the function

$$f_1^*(x) = \begin{cases} 0.25 & \text{if} \quad x \in [2.8, 2.81] \\ 0.5 & \text{if} \quad x \in [1.5, 1.51] \end{cases}$$

where $x \in X = [0, 4.5]$. x_i are obtained by random sampling 2000 non-repeatable numbers from X with a uniform distribution, and the corresponding output values y_k are computed by $f(x)$ and $f^*(x)$. The overall function is $F_1(x) = f_1(x) + f_1^*(x)$. The training data with 2000 (x_i, y_i) pairs is chosen to perform the notch function approximation $f(x)$ with the corruption of the outliers $f^*(x)$. MLPs with 1:16:1 architectures are initiated to all training sessions.

2). The smooth function is defined by

$$f_2(x) = g(x, \frac{1}{6}, \frac{1}{2}, \frac{1}{6})$$

The outliers with very large derivations given by

$$f_2^*(x) = g(x, \frac{1}{64}, \frac{1}{4}, \frac{1}{128}) + g(x, \frac{1}{64}, \frac{1}{20}, \frac{1}{128})$$

Where $x \in X = [0, 1]$, g is defined as

$$g(x) = \frac{\alpha}{\sqrt{2\pi}\tau} \cos(\frac{(x-\mu)\pi}{\tau}) e^{-\frac{(x-\mu)^2}{2\tau^2}} \tag{6.13}$$

The overall function is $F_2(x) = f_2(x) + f_2^*(x)$. The input values x_i are selected by sampling 2000 numbers from a uniformly distributed grid on X. The corresponding output values y_i are computed by $f(x)$ and $f^*(x)$. The training data with 2000 (x_i, y_i) pairs is chosen to perform the smooth function approximation with two fine-feature intrusions. MLPs with 1:15:1 architectures are applied.

3). Define a smooth function by

$$F_3(x) = g(x, \frac{1}{5}, \frac{1}{4}, \frac{1}{12}) + g(x, \frac{1}{5}, \frac{3}{4}, \frac{1}{12}) + g(x, \frac{1}{64}, \frac{5}{4}, \frac{1}{12})$$

The outliers from the under-sampled points are given by the lower sampling frequency. $x \in X = [0, 1.5]$, the outliers x_i^* are collected by sampling 200 numbers from a uniform distributed grid on $[0, 0.5]$ and 200 numbers from a uniform distributed grid on $[1.0, 1.5]$. The smooth function is generated by 2000 numbers of points from a uniform distributed grid on $(0.5, 1.0)$. So the oracle function is $f_3(x) = F_3(x), x \in (0.5, 1)$ and the outlier generating function is $f_3(x) = F_3(x), x \in [0, 0.5] \bigcup [1, 1.5]$. The output y_i are computed by $F_3(x)$. The training data with 2400 (x_i, y_i) pairs is the function contaminated by the unevenly under-sampled segments. MLPs with 1:12:1 architecture are initiated to all training sessions.

The sample functions represent three typical sources of outliers (Figure 6.1 (a)). The

Table 6.1. Training and test MSE of approximation on the three sample functions. For NAAE, λ is updated by the fixed-λ and adaptive learning strategies. The average performance over 10 runs are reported.

	MSE		NAAE-fixed-λ		NAAE-adaptive learning	
	Training	Test	Training	Test	Training	Test
$F_1(x)$	0.02	0.022	0.021	0.019	0.021	**0.018**
$F_2(x)$	0.013	0.029	0.005	**0.005**	0.005	**0.005**
$F_3(x)$	0.012	0.037	0.045	**0.01**	0.045	**0.01**

outliers in $F_1(x)$ are in the range of the uncontaminated data but appearing at wrong positions. This sometimes occurs when the labels are normal but maybe corrupted or inaccurate. $F_2(x)$ is affected by outliers with significant large deviations (even exceeding the bounds), indicating the situations where the labels contain some unknown samples. $F_3(x)$ includes seemingly uncontaminated but shifted and under-sampled data. This may happen among heterogeneous observations from several experiments where the samples are highly sparse and biased, and the main function needs to get rid of their influence.

In the following experiments, we use both fixed λ and adaptive learning methods to update the RO index. λ starts at -10^3. NAAE is optimized with batch gradient descent. Batch size is fixed at 20. For each of the functions, we generate the test set consisting 2000 samples from the oracle function $f_i(x), (i \in 1, 2, 3)$ respectively.

The approximation results of 10 runs are summarized in Table 6.1. NAAE is more robust to these three types of outliers with both update strategies. We did not find significant differences between fixed-λ and adaptive learning on these experiments. Training with MSE always achieves lower training error but with a higher test error, which is also known as overfitting. NAAE eliminates the impact of outliers efficiently to achieve better test errors. It is interesting that when approximating the smooth function with fine-features ($F_2(x)$), NAAE achieves better training and test error simultaneously. This perhaps due to both the robustness and optimality of NAAE. Optimizing MSE is unable to get rid of the

outliers. The high non-convexity of the compound function also leads to the problem of local optimum. While ignoring the outliers, NAAE seeks the better local optima due to the expansion of the convex region to obtain better performance. The empirical breakdown point of NAAE on these three samples are 0.4%, 8.4% and 16.7%. The training results are shown in Figure 6.1.

6.5.2 Visual Digits Recognition

We further investigate NAAE in a typical learning problem, the digits recognition tasks on the MNIST dataset. The MNIST dataset (LeCun *et al.* 1998) consists of hand written digits 0-9 which are 28x28 in size. There are 60,000 training images and 10,000 testing images in total. We use 10,000 images in the training set for validation to select the hyperparameters and report the performance on the test set. We test our method on this dataset without data augmentation.

The NAAE function is minimized by batch gradient descent with momentum at 0.9. The learning rate and l_2 penalty are fixed at 0.5 and 0.05. The penalty weight a are selected in $\{1, 0.1, 0.001\}$ on validation sets respectively. The initial λ is fixed at -100. We use the hold-out validation set to select the best model, which is used to make predictions on the test set. All experiments are implemented quite easily in Python and Theano to obtain GPU acceleration (Bastien *et al.* 2012).

On the MNIST dataset we use the same structure of LeNet5 with two convolutional max-pooling layers but followed by only one fully connected layer and a densely connected softmax layer. The first convolutional layer has 20 feature maps of size 5×5 and max-pooled by 2×2 non-overlapping windows. The second convolutional layer has 50 feature maps with the same convolutional and max-pooling size. The fully connected layer has

[3](1)(Mairal *et al.* 2014);(2)(Lee *et al.* 2014); (3)(Zeiler & Fergus 2013);(4)(Jarrett *et al.* 2009)

Method[3]	Error %
Convolutional Kernel Networks$^{(1)}$	**0.39**
Deeply Supervised Nets + dropout$^{(2)}$	**0.39**
ConvNets + dropout$^{(3)}$	0.55
large ConvNets, unsup pretraining$^{(4)}$	0.53
ConvNets + NAAE (Ours)	0.53
ConvNets + NAAE + dropout (Ours)	**0.39**

Table 6.2. Test set misclassification rates of the best methods that utilized convolutional networks on the original MNIST dataset using single model.

500 hidden units. An l_2 prior was used with the strength 0.05 in the Softmax layer. Trained by ANRAT, we can obtain a test set error of 0.53%, which is the best result we are aware of that does not use dropout on the pure ConvNets. With dropout, our method achieve the same performance with the state-of-art at 0.39% error. We summarize the best published results on the standard MNIST dataset in Table 6.2.

The above results demonstrate both the optimality and robustness of NAAE. The improvements in MNIST are generally due to better regularization techniques with good optimization strategy. With simple experiment settings and network architectures, NAAE enables the simple ConvNets to optimally learn the models while getting rid of specific outlier samples to enhance the generalization capability.

To better evaluate the robustness of NAAE, we used a fraction of a permutation of the labels to add outliers among the samples but without fixed patterns like (Reed *et al.* 2014). The learning model is the same as the last experiment on the MNIST dataset. The noise fraction ranges from 0% to 35%. Figure 6.2 shows that our NAAE with the adaptive training method provides a significant benefit in the case of permuted labels. The consistently lower fitting MSE also indicates its optimality. As the noise level increases up to 30%, NAAE provides the benefit in this high-noise regime, but is only slightly better than or

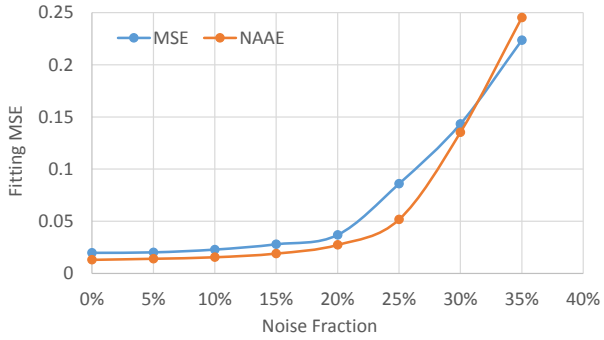

FIG. 6.2. Digit recognition error rates versus percent corrupted labels.

same with training using MSE overall. When the noise fraction is larger than 30%, NAAE perform even worse than MSE due to the failure of robustness to identify the outliers, while the convex region still expands to 'optimally' fit the noisy model, thus incurs overfitting.

6.6 Conclusions

We introduced the Normalized Anomaly-Averting Estimator by pushing the robust-optimal (RO) index λ to $-\infty$. It is robust to outliers due to its quasi-minimin functionality. The robustness is realized and controlled by its adaptive RO index without any predefined threshold. Its optimality is guaranteed by the expansion of the convexity region in its Hessian matrix to largely avoid the local optima. We provide both theoretical and empirical support for its robustness and optimality.

In future work, it may be promising to consider updating λ between $-\infty$ to ∞ to achieve robustness to both small noise and large outliers while preserving optimality, provide the risk and population risk bounds and extend our approach to other areas like aircraft and robotics control. It is also interesting to augment large-scale training for detection (e.g. ILSVRC and speech) with unlabeled and more weakly-labeled images/corpus.

Chapter 7

CONCLUSION AND FUTURE WORK

We introduced three different perspectives/frameworks to model temporal data (primarily time series data) and learn the representations. Motivated by the internal correlation embedded in time series and intrinsic property of BoP representations, we proposed time warping SAX to integrate the temporal correlation when building SAX words and BoP representations. Pooling SAX-BoP with Boosting approach suppose to solve classification problems on the multivariate vital signs time series. Instead of majority voting, the Boosting algorithm is applied to significantly improve the performance. When the data has strong internal temporal correlations, time-warping always tends to work better than the vanilla SAX. Pooling SAX-BoP with Boosting approach is motivated to solve the problem of representation modeling on multivariate time series. It demonstrates its effectiveness and efficiency compared with the current state-of-the-art approaches especially on the multivariate physiological data. Instead of majority voting, the Boosting algorithm is applied to significantly improve the performance. When the number of channel is smaller than 6, Pooling SAX-BoP is much worth to try.

Imaging time series is an off-line approach for spatially encoding the temporal patterns for classification with deep learning frameworks. We created a pipeline for converting trajectory and time series data into novel representations, GAF and MTF images,

and extracted high-level features from these using ConvNets. The features were subsequently used for classification. We demonstrated that our approach yields competitive results when compared to state-of-the-art methods by searching a relatively small parameter space. We found that GAF-MTF multi-channel images are scalable to larger numbers of quasi-orthogonal features that yield more comprehensive images. Our analysis of high-level features learned from ConvNets suggested Tiled ConvNets work like multi-frequency moving averages that benefit from the 2D temporal dependency that is preserved by the Gramian matrix. When the sample length is standard but the sample size is larges, imaging + deep learning demonstrates the supreme performance on a variety type of temporal data.

Modeling and feature learning using deep neural networks are widely explored recently. We introduce a set of novel error estimators with learning methods, NRAE and ANRAT to help train deep neural networks. Theoretically, we prove the effectiveness of Normalized Risk-Averting Error on its arithmetic bound, global convexity and local convexity lower-bounded by standard L_p-norm error when convexity index $\lambda \geq 1$. By analyzing the gradient on λ, we explained the reason why using back propagation on λ works. The experiments on deep/shallow network layouts demonstrate comparable or better performance with the same experimental settings among pure ConvNets and MLP + batch SGD on MSE and CE (with or without dropout). Other than unsupervised pretraining, it provides a new perspective to address the non-convex optimization strategy in DNNs.

NAAE is a variation of NRAE with the constraint $\lambda < 0$. That is, we push the robust-optimal (RO) index λ to $-\infty$. It is robust to outliers due to its quasi-minimin functionality. The robustness is realized and controlled by its adaptive RO index without any predefined threshold. Its optimality is guaranteed by the expansion of the convexity region in its Hessian matrix to largely avoid the local optima. We provide both theoretical and empirical support for its robustness and optimality. NRAE/NAAE with ANRAT are good alternatives to MSE/cross entropy in fitting and learning problems, especially for modeling temporal

data using neural nets in the end-to-end manner.

To summarize, symbolic approximation approaches discretize the temporal data as symbolic representation, facilitating to extract the bag-of-features for modeling temporal correlation. By symbolic approximation, a bunch of learning algorithm in natural language processing (NLP) can be adapted to temporal data mining. Imaging + deep learning framework enables the model to learn more complicated temporal dynamics. It bridges the research in computer vision and time series analysis, enabling the modeling and representation learning on temporal data to be benefit from the rapid development of deep learning based computer vision approaches. NRAE/NAAE and ANRAT provide another perspective to address the optimality and robustness with a unified framework for the non-convex optimization problem in deep learning.

As for the future work, we will explore the idea of imaging and symbolization for representation learning on the multivariate temporal data. How to learn the representations which will benefit temporal reasoning/inference is also a key topic. For the Non-convex optimization problem, more effort is worth to be spent on much large data set (e.g. ImageNet) with more complex vision models (e.g. GooleNet, VGG-Net) and recurrent models (e.g. Deep LSTM, Deep GRU) to better understand its potential on static and temporal data learning. Its robustness is also crucial to control theory.

REFERENCES

[1] Abdel-Hamid, O.; Mohamed, A.-r.; Jiang, H.; and Penn, G. 2012. Applying convolutional neural networks concepts to hybrid nn-hmm model for speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, 4277–4280. IEEE.

[2] Abdel-Hamid, O.; Deng, L.; and Yu, D. 2013. Exploring convolutional neural network structures and optimization techniques for speech recognition. In *INTERSPEECH*, 3366–3370.

[3] Aggarwal, C. C., and Yu, P. S. 2001. Outlier detection for high dimensional data. In *ACM Sigmod Record*, volume 30, 37–46. ACM.

[4] Agrawal, R.; Faloutsos, C.; and Swami, A. 1993. *Efficient similarity search in sequence databases*. Springer.

[5] An, J.; Chen, H.; Furuse, K.; Ohbo, N.; and Keogh, E. 2003. Grid-based indexing for large time series databases. In *Intelligent Data Engineering and Automated Learning*. Springer. 614–621.

[6] Angiulli, F., and Pizzuti, C. 2005. Outlier mining in large high-dimensional data sets. *Knowledge and Data Engineering, IEEE Transactions on* 17(2):203–215.

[7] Antunes, C. M., and Oliveira, A. L. 2001. Temporal data mining: An overview. In *KDD Workshop on Temporal Data Mining*, 1–13.

[8] Athitsos, V., and Sclaroff, S. 2005. Boosting nearest neighbor classifiers for multiclass recognition. In *Computer Vision and Pattern Recognition-Workshops, 2005. CVPR Workshops. IEEE Computer Society Conference on*, 45–45. IEEE.

[9] Bakshi, B. R., and Stephanopoulos, G. 1995. Reasoning in time: Modeling, analysis, and pattern recognition of temporal process trends. *Advances in Chemical Engineering* 22:485–548.

[10] Bandt, C., and Pompe, B. 2002. Permutation entropy: a natural complexity measure for time series. *Physical Review Letters* 88(17):174102.

[11] Bankó, Z., and Abonyi, J. 2012. Correlation based dynamic time warping of multivariate time series. *Expert Systems with Applications* 39(17):12814–12823.

[12] Bastien, F.; Lamblin, P.; Pascanu, R.; Bergstra, J.; Goodfellow, I. J.; Bergeron, A.; Bouchard, N.; and Bengio, Y. 2012. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop.

[13] Baydogan, M. G., and Runger, G. 2014. Learning a symbolic representation for multivariate time series classification. *Data Mining and Knowledge Discovery* 1–23.

[14] Baydogan, M. G.; Runger, G.; and Tuv, E. 2013. A bag-of-features framework to classify time series. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 35(11):2796–2802.

[15] Bengio, Y.; Lamblin, P.; Popovici, D.; Larochelle, H.; et al. 2007. Greedy layer-wise training of deep networks. *Advances in neural information processing systems* 19:153.

[16] Bengio, Y.; Yao, L.; Alain, G.; and Vincent, P. 2013. Generalized denoising auto-encoders as generative models. In *Advances in Neural Information Processing Systems*, 899–907.

[17] Blake, A., and Zisserman, A. 1987. *Visual reconstruction*, volume 2. MIT press Cambridge.

[18] Box, G. E.; Jenkins, G. M.; and Reinsel, G. C. 2013. *Time series analysis: forecasting and control*. John Wiley & Sons.

[19] Camerra, A.; Palpanas, T.; Shieh, J.; and Keogh, E. 2010. isax 2.0: Indexing and mining one billion time series. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, 58–67. IEEE.

[20] Campanharo, A. S.; Sirer, M. I.; Malmgren, R. D.; Ramos, F. M.; and Amaral, L. A. N. 2011. Duality between time series and networks. *PloS one* 6(8):e23378.

[21] Chan, F.-P.; Fu, A.-C.; and Yu, C. 2003. Haar wavelets for efficient similarity search of time-series: with and without time warping. *Knowledge and Data Engineering, IEEE Transactions on* 15(3):686–705.

[22] Chen, D. S., and Jain, R. C. 1994. A robust backpropagation learning algorithm for function approximation. *Neural Networks, IEEE Transactions on* 5(3):467–479.

[23] Choromanska, A.; Henaff, M.; Mathieu, M.; Arous, G. B.; and LeCun, Y. 2014. The loss surface of multilayer networks. *arXiv preprint arXiv:1412.0233*.

[24] Clark, S. P. 1990. Estimating the fractal dimension of chaotic time series. *Lincoln Laboratory Journal* 3(1).

[25] Coates, A., and Ng, A. Y. 2011. Selecting receptive fields in deep networks. In *Advances in Neural Information Processing Systems*, 2528–2536.

[26] Cohen, N., and Shashua, A. 2014. Simnets: A generalization of convolutional networks. *arXiv preprint arXiv:1410.0781*.

[27] Dauphin, Y. N.; Pascanu, R.; Gulcehre, C.; Cho, K.; Ganguli, S.; and Bengio, Y. 2014. Identifying and attacking the saddle point problem in high-dimensional non-

convex optimization. In *Advances in Neural Information Processing Systems*, 2933–2941.

[28] Daw, C. S.; Finney, C. E. A.; and Tracy, E. R. 2003. A review of symbolic analysis of experimental data. *Review of Scientific Instruments* 74(2):915–930.

[29] Deng, L.; Abdel-Hamid, O.; and Yu, D. 2013. A deep convolutional neural network using heterogeneous pooling for trading acoustic invariance with phonetic confusion. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, 6669–6673. IEEE.

[30] Deng, L.; Li, J.; Huang, J.-T.; Yao, K.; Yu, D.; Seide, F.; Seltzer, M.; Zweig, G.; He, X.; Williams, J.; et al. 2013. Recent advances in deep learning for speech research at microsoft. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, 8604–8608. IEEE.

[31] Deng, L.; Hinton, G.; and Kingsbury, B. 2013. New types of deep neural network learning for speech recognition and related applications: An overview. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, 8599–8603. IEEE.

[32] Ding, H.; Trajcevski, G.; and Scheuermann, P. 2008. Efficient maintenance of continuous queries for trajectories. *GeoInformatica* 12(3):255–288.

[33] Donner, R. V.; Zou, Y.; Donges, J. F.; Marwan, N.; and Kurths, J. 2010. Recurrence networksa novel paradigm for nonlinear time series analysis. *New Journal of Physics* 12(3):033025.

[34] Donner, R. V.; Small, M.; Donges, J. F.; Marwan, N.; Zou, Y.; Xiang, R.; and Kurths,

J. 2011. Recurrence-based time series analysis by means of complex network methods. *International Journal of Bifurcation and Chaos* 21(04):1019–1046.

[35] Erhan, D.; Bengio, Y.; Courville, A.; Manzagol, P.-A.; Vincent, P.; and Bengio, S. 2010. Why does unsupervised pre-training help deep learning? *The Journal of Machine Learning Research* 11:625–660.

[36] Faloutsos, C.; Ranganathan, M.; and Manolopoulos, Y. 1994. *Fast subsequence matching in time-series databases*, volume 23. ACM.

[37] Fan, R.-E.; Chang, K.-W.; Hsieh, C.-J.; Wang, X.-R.; and Lin, C.-J. 2008. Liblinear: A library for large linear classification. *The Journal of Machine Learning Research* 9:1871–1874.

[38] Feng, J.; Xu, H.; Mannor, S.; and Yan, S. 2014. Robust logistic regression and classification. In *Advances in Neural Information Processing Systems*, 253–261.

[39] Flash, T., and Hochner, B. 2005. Motor primitives in vertebrates and invertebrates. *Current opinion in neurobiology* 15(6):660–666.

[40] Freund, Y., and Schapire, R. E. 1995. A desicion-theoretic generalization of on-line learning and an application to boosting. In *Computational learning theory*, 23–37. Springer.

[41] Freund, Y.; Schapire, R. E.; et al. 1996. Experiments with a new boosting algorithm. In *ICML*, volume 96, 148–156.

[42] Gan, Z.; Henao, R.; Carlson, D.; and Carin, L. 2015. Learning deep sigmoid belief networks with data augmentation. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, 268–276.

[43] Gandhi, S. R., and Oates, T. 2015. A generative model for time series based on multiple normal distributions. *Master thesis*.

[44] Goodfellow, I. J.; Warde-Farley, D.; Lamblin, P.; Dumoulin, V.; Mirza, M.; Pascanu, R.; Bergstra, J.; Bastien, F.; and Bengio, Y. 2013a. Pylearn2: a machine learning research library. *arXiv preprint arXiv:1308.4214*.

[45] Goodfellow, I. J.; Warde-Farley, D.; Mirza, M.; Courville, A.; and Bengio, Y. 2013b. Maxout networks. *arXiv preprint arXiv:1302.4389*.

[46] Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, 2672–2680.

[47] Gui, Y.; Lo, J. T.-H.; and Peng, Y. 2014. A pairwise algorithm for training multilayer perceptrons with the normalized risk-averting error criterion. In *Neural Networks (IJCNN), 2014 International Joint Conference on*, 358–365. IEEE.

[48] Hannun, A.; Case, C.; Casper, J.; Catanzaro, B.; Diamos, G.; Elsen, E.; Prenger, R.; Satheesh, S.; Sengupta, S.; Coates, A.; et al. 2014. Deepspeech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567*.

[49] He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *arXiv preprint arXiv:1502.01852*.

[50] Hermansky, H. 1990. Perceptual linear predictive (plp) analysis of speech. *the Journal of the Acoustical Society of America* 87(4):1738–1752.

[51] Hilbert, D. 1891. Ueber die stetige abbildung einer line auf ein flächenstück. *Mathematische Annalen* 38(3):459–460.

[52] Hinton, G.; Deng, L.; Yu, D.; Dahl, G. E.; Mohamed, A.-r.; Jaitly, N.; Senior, A.; Vanhoucke, V.; Nguyen, P.; Sainath, T. N.; et al. 2012. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *Signal Processing Magazine, IEEE* 29(6):82–97.

[53] Hinton, G.; Osindero, S.; and Teh, Y.-W. 2006. A fast learning algorithm for deep belief nets. *Neural computation* 18(7):1527–1554.

[54] Hornik, K.; Stinchcombe, M.; and White, H. 1989. Multilayer feedforward networks are universal approximators. *Neural networks* 2(5):359–366.

[55] Hubel, D. H., and Wiesel, T. N. 1962. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology* 160(1):106.

[56] Jacobson, D. H. 1973. Optimal stochastic linear systems with exponential performance criteria and their relation to deterministic differential games. *Automatic Control, IEEE Transactions on* 18(2):124–131.

[57] Jaderberg, M.; Simonyan, K.; Zisserman, A.; and Kavukcuoglu, K. 2015. Spatial transformer networks. *arXiv preprint arXiv:1506.02025*.

[58] Jarrett, K.; Kavukcuoglu, K.; Ranzato, M.; and LeCun, Y. 2009. What is the best multi-stage architecture for object recognition? In *Computer Vision, 2009 IEEE 12th International Conference on*, 2146–2153. IEEE.

[59] Kalpakis, K.; Gada, D.; and Puttagunta, V. 2001. Distance measures for effective clustering of arima time-series. In *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, 273–280. IEEE.

[60] Karpathy, A., and Fei-Fei, L. 2015. Deep visual-semantic alignments for generating

image descriptions. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[61] Kavukcuoglu, K.; Sermanet, P.; Boureau, Y.-L.; Gregor, K.; Mathieu, M.; and Cun, Y. L. 2010. Learning convolutional feature hierarchies for visual recognition. In *Advances in neural information processing systems*, 1090–1098.

[62] Keogh, E. J., and Pazzani, M. J. 1998. An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback. In *KDD*, volume 98, 239–243.

[63] Keogh, E. J., and Pazzani, M. J. 2000. Scaling up dynamic time warping for datamining applications. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, 285–289. ACM.

[64] Keogh, E.; Chakrabarti, K.; Pazzani, M.; and Mehrotra, S. 2001. Dimensionality reduction for fast similarity search in large time series databases. *Knowledge and information Systems* 3(3):263–286.

[65] Keogh, E.; Xi, X.; Wei, L.; and Ratanamahatana, C. A. 2006. The ucr time series classification/clustering homepage. *URL= http://www. cs. ucr. edu/˜ eamonn/time_series_data*.

[66] Keogh, E.; Xi, X.; Wei, L.; and Ratanamahatana, C. A. 2011. The ucr time series classification/clustering homepage. *URL= http://www. cs. ucr. edu/˜ eamonn/time_series_data*.

[67] Keogh, E.; Lonardi, S.; and Ratanamahatana, C. A. 2004. Towards parameter-free data mining. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, 206–215. ACM.

[68] Kononenko, I. 2001. Machine learning for medical diagnosis: history, state of the art and perspective. *Artificial Intelligence in medicine* 23(1):89–109.

[69] Korn, F.; Jagadish, H. V.; and Faloutsos, C. 1997. Efficiently supporting ad hoc queries in large datasets of time sequences. *ACM SIGMOD Record* 26(2):289–300.

[70] Kriegel, H.-P.; Zimek, A.; et al. 2008. Angle-based outlier detection in high-dimensional data. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, 444–452. ACM.

[71] Krizhevsky, A., and Hinton, G. 2009. Learning multiple layers of features from tiny images. *Computer Science Department, University of Toronto, Tech. Rep* 1(4):7.

[72] Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 1097–1105.

[73] Kumar, N.; Lolla, V. N.; Keogh, E. J.; Lonardi, S.; and Ratanamahatana, C. A. 2005. Time-series bitmaps: a practical visualization tool for working with large time series databases. In *SDM*, 531–535. SIAM.

[74] Künsch, H. R.; Stefanski, L. A.; and Carroll, R. J. 1989. Conditionally unbiased bounded-influence estimation in general regression models, with applications to generalized linear models. *Journal of the American Statistical Association* 84(406):460–466.

[75] Längkvist, M.; Karlsson, L.; and Loutfi, A. 2014. A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern Recognition Letters* 42:11–24.

[76] Larochelle, H.; Bengio, Y.; Louradour, J.; and Lamblin, P. 2009. Exploring strategies for training deep neural networks. *The Journal of Machine Learning Research* 10:1–40.

[77] Lawrence, S.; Giles, C. L.; Tsoi, A. C.; and Back, A. D. 1997. Face recognition: A convolutional neural-network approach. *Neural Networks, IEEE Transactions on* 8(1):98–113.

[78] LeCun, Y., and Bengio, Y. 1995. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks* 3361.

[79] LeCun, Y.; Boser, B.; Denker, J. S.; Henderson, D.; Howard, R. E.; Hubbard, W.; and Jackel, L. D. 1989. Backpropagation applied to handwritten zip code recognition. *Neural computation* 1(4):541–551.

[80] LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11):2278–2324.

[81] LeCun, Y.; Kavukcuoglu, K.; and Farabet, C. 2010. Convolutional networks and applications in vision. In *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, 253–256. IEEE.

[82] Lee, J.-G.; Han, J.; Li, X.; and Gonzalez, H. 2008. Traclass: trajectory classification using hierarchical region-based and trajectory-based clustering. *Proceedings of the VLDB Endowment* 1(1):1081–1094.

[83] Lee, C.-Y.; Xie, S.; Gallagher, P.; Zhang, Z.; and Tu, Z. 2014. Deeply-supervised nets. *arXiv preprint arXiv:1409.5185*.

[84] Leggetter, C. J., and Woodland, P. C. 1995. Maximum likelihood linear regression for speaker adaptation of continuous density hidden markov models. *Computer Speech & Language* 9(2):171–185.

[85] Liano, K. 1996. Robust error measure for supervised neural network learning with outliers. *Neural Networks, IEEE Transactions on* 7(1):246–250.

[86] Lin, J.; Keogh, E.; Lonardi, S.; and Chiu, B. 2003. A symbolic representation of time series, with implications for streaming algorithms. In *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, 2–11. ACM.

[87] Lin J, Williamson S, B. K. D. D. 2012. *Pattern recognition in time series*. Chapman & Hall, To appear.

[88] Lin, J.; Khade, R.; and Li, Y. 2012. Rotation-invariant similarity in time series using bag-of-patterns representation. *Journal of Intelligent Information Systems* 39(2):287–315.

[89] Liu, W., and Floudas, C. A. 1993. A remark on the gop algorithm for global optimization. *Journal of Global Optimization* 3(4):519–521.

[90] Lo, J. T., and Bassu, D. 2001. Training multilayer perceptrons in the presence of measurement outliers. In *Neural Networks, 2001. Proceedings. IJCNN'01. International Joint Conference on*, volume 3, 2030–2035. IEEE.

[91] Lo, J. T.-H.; Gui, Y.; and Peng, Y. 2012. Overcoming the local-minimum problem in training multilayer perceptrons with the nrae training method. In *Advances in Neural Networks–ISNN 2012*. Springer. 440–447.

[92] Lo, J. T.-H.; Gui, Y.; and Peng, Y. 2013. Overcoming the local-minimum problem in training multilayer perceptrons by gradual deconvexification. In *Neural Networks (IJCNN), The 2013 International Joint Conference on*, 1–6. IEEE.

[93] Lo, J. T.-H. 2010. Convexification for data fitting. *Journal of global optimization* 46(2):307–315.

[94] Lu, G.; Yang, F.; Taylor, J.; and Stein, J. 2009. A comparison of photoplethysmography and ecg recording to analyse heart rate variability in healthy subjects. *Journal of medical engineering & technology* 33(8):634–641.

[95] Mairal, J.; Koniusz, P.; Harchaoui, Z.; and Schmid, C. 2014. Convolutional kernel networks. In *Advances in Neural Information Processing Systems*, 2627–2635.

[96] Manly, B. F. 2006. *Randomization, bootstrap and Monte Carlo methods in biology*, volume 70. CRC Press.

[97] Martens, J., and Grosse, R. 2015. Optimizing neural networks with kronecker-factored approximate curvature. *arXiv preprint arXiv:1503.05671*.

[98] Martens, J. 2010. Deep learning via hessian-free optimization. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 735–742.

[99] Min Lin, Qiang Chen, S. Y. 2014. Network in network. *arXiv preprint arXiv:1312.4400v3*.

[100] Mohamed, A.-r.; Dahl, G. E.; and Hinton, G. 2012. Acoustic modeling using deep belief networks. *Audio, Speech, and Language Processing, IEEE Transactions on* 20(1):14–22.

[101] Moon, B.; Jagadish, H. V.; Faloutsos, C.; and Saltz, J. H. 2001. Analysis of the clustering properties of the hilbert space-filling curve. *Knowledge and Data Engineering, IEEE Transactions on* 13(1):124–141.

[102] Mörchen, F., and Ultsch, A. 2006. Finding persisting states for knowledge discovery in time series. In *From Data and Information Analysis to Knowledge Engineering*. Springer. 278–285.

[103] Nair, V., and Hinton, G. E. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 807–814.

[104] Nanopoulos, A.; Alcock, R.; and Manolopoulos, Y. 2001. Feature-based classification of time-series data. *International Journal of Computer Research* 10(3).

[105] Ng, A. Y. 1997. Preventing" overfitting" of cross-validation data. In *ICML*, volume 97, 245–253.

[106] Ng, A. 2011. Sparse autoencoder. *CS294A Lecture notes* 72.

[107] Ngiam, J.; Chen, Z.; Chia, D.; Koh, P. W.; Le, Q. V.; and Ng, A. Y. 2010. Tiled convolutional neural networks. In *Advances in Neural Information Processing Systems*, 1279–1287.

[108] Ngiam, J.; Coates, A.; Lahiri, A.; Prochnow, B.; Le, Q. V.; and Ng, A. Y. 2011. On optimization methods for deep learning. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, 265–272.

[109] Oates, T.; Mackenzie, C. F.; Stansbury, L. G.; Aarabi, B.; Stein, D. M.; and Hu, P. F. 2012a. Predicting patient outcomes from a few hours of high resolution vital signs data. In *Machine Learning and Applications (ICMLA), 2012 11th International Conference on*, volume 2, 192–197. IEEE.

[110] Oates, T.; Mackenzie, C. F.; Stein, D. M.; Stansbury, L. G.; Dubose, J.; Aarabi, B.; and Hu, P. F. 2012b. Exploiting representational diversity for time series classification. In *Machine Learning and Applications (ICMLA), 2012 11th International Conference on*, volume 2, 538–544. IEEE.

[111] Ordonez, P.; Oates, T.; and Desjardins, M. 2012. *Multivariate time-series analysis of physiological and clinical data*. University of Maryland at Baltimore County.

[112] Panuccio, A.; Bicego, M.; and Murino, V. 2002. A hidden markov model-based approach to sequential data clustering. In *Structural, Syntactic, and Statistical Pattern Recognition*. Springer. 734–743.

[113] Papadimitriou, S., and Yu, P. 2006. Optimal multi-scale patterns in time series streams. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, 647–658. ACM.

[114] Perng, C.-S.; Wang, H.; Zhang, S. R.; and Parker, D. S. 2000. Landmarks: a new model for similarity-based pattern querying in time series databases. In *Data Engineering, 2000. Proceedings. 16th International Conference on*, 33–42. IEEE.

[115] Poultney, C.; Chopra, S.; Cun, Y. L.; et al. 2006. Efficient learning of sparse rerepresentations with an energy-based model. In *Advances in neural information processing systems*, 1137–1144.

[116] Prechelt, L. 1998. Automatic early stopping using cross validation: quantifying the criteria. *Neural Networks* 11(4):761–767.

[117] Pregibon, D. 1982. Resistant fits for some commonly used logistic models with medical applications. *Biometrics* 485–498.

[118] Pukelsheim, F. 1994. The three sigma rule. *The American Statistician* 48(2):88–91.

[119] Rakthanmanon, T., and Keogh, E. 2013. Fast shapelets: A scalable algorithm for discovering time series shapelets. In *Proceedings of the thirteenth SIAM conference on data mining (SDM)*. SIAM.

[120] Ranzato, M.; Huang, F. J.; Boureau, Y.-L.; and LeCun, Y. 2007. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, 1–8. IEEE.

[121] Ratanamahatana, C.; Keogh, E.; Bagnall, A. J.; and Lonardi, S. 2005. A novel bit level time series representation with implication of similarity search and clustering. In *Advances in Knowledge Discovery and Data Mining*. Springer. 771–777.

[122] Ravi Kanth, K.; Agrawal, D.; and Singh, A. 1998. Dimensionality reduction for similarity searching in dynamic databases. In *ACM SIGMOD Record*, volume 27, 166–176. ACM.

[123] Reed, S.; Lee, H.; Anguelov, D.; Szegedy, C.; Erhan, D.; and Rabinovich, A. 2014. Training deep neural networks on noisy labels with bootstrapping. *arXiv preprint arXiv:1412.6596*.

[124] Reynolds, D. A., and Rose, R. C. 1995. Robust text-independent speaker identification using gaussian mixture speaker models. *Speech and Audio Processing, IEEE Transactions on* 3(1):72–83.

[125] Riedl, M.; Müller, A.; and Wessel, N. 2013. Practical considerations of permutation entropy. *The European Physical Journal Special Topics* 222(2):249–262.

[126] Rousseeuw, P. J. 1984. Least median of squares regression. *Journal of the American statistical association* 79(388):871–880.

[127] Salakhutdinov, R., and Hinton, G. E. 2009. Deep boltzmann machines. In *International Conference on Artificial Intelligence and Statistics*, 448–455.

[128] Schapire, R. 2013. Explaining adaboost. In Schlkopf, B.; Luo, Z.; and Vovk, V., eds., *Empirical Inference*. Springer Berlin Heidelberg. 37–52.

[129] Sebastiani, P.; Ramoni, M.; Cohen, P.; Warwick, J.; and Davis, J. 1999. Discovering dynamics using bayesian clustering. In *Advances in intelligent data analysis*. Springer. 199–209.

[130] Senin, P., and Malinchik, S. 2013. Sax-vsm: Interpretable time series classification using sax and vector space model. In *Data Mining (ICDM), 2013 IEEE 13th International Conference on*, 1175–1180. IEEE.

[131] Silva, D. F.; Souza, V.; De, M.; and Batista, G. E. 2013. Time series classification using compression distance of recurrence plots. In *Data Mining (ICDM), 2013 IEEE 13th International Conference on*, 687–696. IEEE.

[132] Speyer, J. L.; Deyst, J.; and Jacobson, D. 1974. Optimization of stochastic linear systems with additive measurement and process noise using exponential performance criteria. *Automatic Control, IEEE Transactions on* 19(4):358–366.

[133] Sra, S.; Nowozin, S.; and Wright, S. J. 2012. *Optimization for machine learning*. Mit Press.

[134] Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15(1):1929–1958.

[135] Stefanski, L. A.; Carroll, R. J.; and Ruppert, D. 1986. Optimally hounded score functions for generalized linear models with applications to logistic regression. *Biometrika* 73(2):413–424.

[136] Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; and Rabinovich, A. 2015. Going deeper with convolutions.

[137] Taylor, G. W. 2009. *Composable, distributed-state models for high-dimensional time series*. Ph.D. Dissertation, University of Toronto.

[138] Tibshirani, J., and Manning, C. D. 2013. Robust logistic regression using shift parameters. *CoRR*.

[139] Vincent, P.; Larochelle, H.; Bengio, Y.; and Manzagol, P.-A. 2008. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, 1096–1103. ACM.

[140] Vincent, P.; Larochelle, H.; Lajoie, I.; Bengio, Y.; and Manzagol, P.-A. 2010. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *The Journal of Machine Learning Research* 11:3371–3408.

[141] Wang, J.; Liu, P.; She, M. F.; Nahavandi, S.; and Kouzani, A. 2013. Bag-of-words representation for biomedical time series classification. *Biomedical Signal Processing and Control* 8(6):634–644.

[142] Wang, Z.; Oates, T.; and Lo, J. 2015. Adaptive normalized risk-averting training for deep neural networks. *arXiv preprint arXiv:1506.02690*.

[143] Weng, X., and Shen, J. 2008. Classification of multivariate time series using locality preserving projections. *Knowledge-Based Systems* 21(7):581–587.

[144] Whittle, P. 1990. *Risk-sensitive optimal control*. John Wiley & Son Ltd.

[145] Xie, J., and Yan, W. 2007. Pattern-based characterization of time series. *International Journal of Information and Systems Science* 3(3):479–491.

[146] Ye, L., and Keogh, E. 2009. Time series shapelets: a new primitive for data mining. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, 947–956. ACM.

[147] Yi, B.-K., and Faloutsos, C. 2000. Fast time sequence indexing for arbitrary lp norms. VLDB.

[148] Yu, C.; Liu, Z.; McKenna, T.; Reisner, A. T.; and Reifman, J. 2006. A method for automatic identification of reliable heart rates calculated from ecg and ppg waveforms. *Journal of the American Medical Informatics Association* 13(3):309–320.

[149] Zeiler, M. D., and Fergus, R. 2013. Stochastic pooling for regularization of deep convolutional neural networks. *arXiv preprint arXiv:1301.3557*.

[150] Zheng, Y.; Liu, Q.; Chen, E.; Ge, Y.; and Zhao, J. L. 2014. Time series classification using multi-channels deep convolutional neural networks. In *Web-Age Information Management*. Springer. 298–310.

[151] Zhou, F.; Torre, F.; and Hodgins, J. K. 2008. Aligned cluster analysis for temporal segmentation of human motion. In *Automatic Face & Gesture Recognition, 2008. FG'08. 8th IEEE International Conference on*, 1–7. IEEE.