

**DATA COMPRESSION OVER SEISMIC SENSOR
NETWORKS**

BY

HILAL HUDAN NUHA

A Dissertation Presented to the
DEANSHIP OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

DOCTOR OF PHILOSOPHY

In

ELECTRICAL ENGINEERING

DECEMBER 2018

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN- 31261, SAUDI ARABIA

DEANSHIP OF GRADUATE STUDIES

This thesis, written by Hilal Hudan Nuha under the direction of his thesis advisor and approved by his thesis committee, has been presented and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of **DOCTOR OF PHILOSOPHY IN ELECTRICAL ENGINEERING.**



Dr. Mohamed Mohandes
(Advisor)



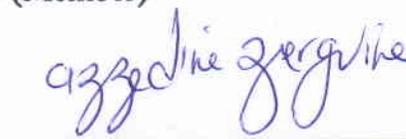
Dr. Mohamed Deriche
(Co-Advisor)



Dr. Faramarz Fekri
(Member)



Dr. Abdullatif Al-Shuhail
(Member)



Dr. Azzedine Zerguine
(Member)



Dr. Abdallah S. Al-Ahmari
Department Chairman



Dr. Salam A. Zummo
Dean of Graduate Studies



Date

21/4/19

© Hilal Hudan Nuha

2018

To my parents, my wife, and my children

ACKNOWLEDGMENTS

I would like to express the deepest appreciation to my advisor Prof. M. Mohandes for introducing me to machine learning, for his patience, for life lessons he taught me. Also my co-advisor Dr. M. Deriche for introducing me to statistical signal processing.

I am indebted to Dr. Bo Liu for his research guidance during the course of this dissertation. Dr. Bo Liu is not listed an advisor but he is a mentor to me and let me tag along on many research works.

I thank my committee members, Prof. Abdullatif Al-Shuhail whose book helps me a lot, Prof. F. Fekri for the critical comments, Prof. A. Zerguine who introduces me to the concept of cost function

I would like to thank to CeGP fellows for their support during my research, Dr. Naveed Iqbal, Dr. Quraishi, and Dr. Asjad. who accompanied me in the center and Abdul Majid Lawal who struggled with me to survive from many course-works.

I would like to express my gratitude to Indonesian Community in Khobar and School of Computing (SoC) of Telkom University (Tel-U) for their supports during my study.

Lastly, I would like to acknowledge the financial support from the Deanship of Scientific Research (DSR) of KFUPM under project GTEC-1801.

TABLE OF CONTENTS

ACKNOWLEDGMENTS.....	III
TABLE OF CONTENTS.....	IV
LIST OF TABLES	VIII
LIST OF FIGURES	IX
LIST OF ABBREVIATIONS.....	XI
ABSTRACT.....	XII
ملخص الرسالة.....	XIV
CHAPTER 1 INTRODUCTION	1
1.1 BACKGROUND.....	2
1.2 PROBLEM STATEMENTS	4
1.2.1 RESEARCH OBJECTIVES.....	6
1.2.2 MAIN RESEARCH DIRECTIONS	6
1.3 MAJOR CONTRIBUTIONS	7
1.4 THESIS ORGANIZATION	8
CHAPTER 2 LITERATURE REVIEW	10
2.1 SEISMIC EXPLORATION.....	13
2.2 SEISMIC DATA FORMAT	18
2.2.1 UNSIGNED INTEGER	19

2.2.2	TWO'S COMPLEMENT INTEGER.....	20
2.2.3	IBM FLOATING POINT	20
2.2.4	IEEE FLOATING POINT	21
2.3	SEISMIC DATA COMPRESSION CATEGORIES.....	22
2.3.1	SAMPLING BASED COMPRESSION.....	23
2.3.2	QUANTIZATION BASED COMPRESSION	24
2.3.3	RUN LENGTH ENCODING	26
2.3.4	VARIABLE-LENGTH CODES	27
2.3.5	DICTIONARY BASED COMPRESSION	29
2.3.6	PREDICTION BASED COMPRESSION	30
2.3.7	TRANSFORMATION BASED COMPRESSION.....	31
2.3.8	MACHINE LEARNING AND OTHER TECHNIQUES	38
2.4	METRICS USED IN EVALUATING DATA COMPRESSION TECHNIQUES	39
2.4.1	COMPRESSION RATIO	39
2.4.2	RECONSTRUCTION QUALITY FOR LOSSY COMPRESSION	40
2.5	CONCLUSION AND REMARKS.....	43
 CHAPTER 3 DISTRIBUTED PRINCIPAL COMPONENT ANALYSIS.....		44
3.1	PRINCIPAL COMPONENT ANALYSIS.....	45
3.2	MIXTURE MODEL BASED DPCA	46
3.3	APPLICATION OF THE DPCA ON THE SEISMIC SENSOR NETWORK	50
3.3.1	APPLYING THE DPCA ON TWO FUNDAMENTAL TOPOLOGIES	50
3.3.2	PRACTICAL SEISMIC SENSOR NETWORK	53
3.3.3	COMPRESSION RATIO ANALYSIS	56
3.3.4	EXPERIMENTAL RESULTS	57
3.4	CONCLUDING REMARKS	62

CHAPTER 4 MODEL BASED SEISMIC DATA COMPRESSION.....	63
4.1 SEISMIC TRACE MODELLING	63
4.1.1 SINGLE EDSW ESTIMATION	64
4.1.2 MULTIPLE EDSWS ESTIMATION	67
4.1.3 RESIDUAL ENCODING	70
4.2 EXPERIMENTAL RESULTS	72
4.2.1 PARAMETER ESTIMATION	72
4.2.2 ENCODING THE RESIDUALS	78
4.2.3 COMPARISON WITH DCT.....	79
4.3 CONCLUDING REMARKS	84
CHAPTER 5 MACHINE LEARNING APPROACHES	86
5.1 INTRODUCTION.....	86
5.2 RESTRICTED BOLTZMANN MACHINE FOR AUTO ENCODER.....	88
5.2.1 METHODOLOGY	88
5.2.2 EXPERIMENTAL RESULTS	95
5.2.3 CONCLUDING REMARKS FOR DATA COMPRESSION USING RBM.....	108
5.3 PREDICTION BASED COMPRESSION USING DEEP LEARNING	109
5.3.1 METHODOLOGIES.....	109
5.3.2 EXPERIMENTAL RESULTS	115
5.3.3 CONCLUDING REMARKS ON THE PREDICTION BASED COMPRESSION	120
5.4 DATA COMPRESSION USING STACKED AUTO-ENCODER WITH EXTREME LEARNING MACHINE.....	121
5.4.1 STACKED AUTO ENCODER AND EXTREME LEARNING MACHINE.....	122
5.4.2 PROPOSED MODEL AND LEARNING METHOD	131
5.4.3 EXPERIMENTAL RESULTS.....	134
5.4.4 CONCLUDING REMARKS FOR COMPRESSION USING THE SAE-ELM.....	143

CHAPTER 6 CONCLUSIONS AND FUTURE WORKS.....	145
6.1 CONCLUSIONS	145
6.2 FUTURE RESEARCH DIRECTIONS	147
REFERENCES	149
VITAE	163

LIST OF TABLES

TABLE 4.1: DCT AND EDSW RESIDUAL ENERGY	81
TABLE 5.1: TRAINING DURATION.....	100
TABLE 5.2: COMPRESSION AND DECOMPRESSION TIME OF EAST TEXAS DATASET	101
TABLE 5.3: EXPERIMENT RESULT WITH $cr = 10:1$	101
TABLE 5.4: COMPRESSION AND DECOMPRESSION TIME OF UTAM DATASET	105
TABLE 5.5 SNR OF THE LPC AND THE DNN	119
TABLE 5.6 COMPRESSION RATIO OF THE DNN	120
TABLE 5.7 PERFORMANCE OF SAE-ELM WITH DIFFERENT ARCHITECTURES AND COMPRESSION RATIOS	137
TABLE 5.8 TRAINING DURATION COMPARISON OF THE SAE-ELM USING EAST TEXAS TESTING DATA FOR 10:1 COMPRESSION RATIO	143

LIST OF FIGURES

FIGURE 1.1 OIL SUPPLY AND DEMAND.....	1
FIGURE 1.2 A TYPICAL SEISMIC ACQUISITION.....	3
FIGURE 1.3 TRENDS IN SEISMIC DATA ACQUISITION (A) PRESTACK DENSITY (B) NUMBER OF SHOTS (C) DATA VOLUME.....	3
FIGURE 2.1: TOTAL NUMBER OF YEAR-WISE PUBLICATIONS SINCE 1974 (SCOPUS).....	12
FIGURE 2.2: CMP (A) REGULAR REFLECTOR (B) DIPPING REFLECTOR.....	14
FIGURE 2.3: TYPICAL RECEIVER LAYOUT.....	15
FIGURE 2.4: SWATH SHOOTING	16
FIGURE 2.5 428XL LAND SEISMIC ACQUISITION SYSTEM	17
FIGURE 2.6: SEG-Y FILE FORMAT.....	19
FIGURE 2.7: 4 BYTE HEXADECIMAL EXPONENT DATA.....	20
FIGURE 2.8: IEEE FLOATING POINT (A) 4 BYTE (B) 8 BYTE	21
FIGURE 2.9: COMPRESSION SCHEME CATEGORIES	23
FIGURE 2.10 BELL BASIS FOR LOCAL SINE/COSINE.....	37
FIGURE 3.1: SENSOR LAYOUT MODEL (A) STAR (B) SEQUENTIAL.....	51
FIGURE 3.2: PRACTICAL SEISMIC SENSOR NETWORKS.....	54
FIGURE 3.3: THE NORMALIZED TRACES OF (EAST TEXAS USA DATABASE)	57
FIGURE 3.4: THE NCEs OF THE LPCA AND THE DPCA BY USING 10 SENSORS AND 18 SHOTS.....	59
FIGURE 3.5: THE NCEs OF THE LPCA AND THE DPCA BY USING 33 SENSORS AND 18 SHOTS.....	60
FIGURE 3.6: CUMULATIVE ENERGY COMPARISON OF (EAST TEXAS USA DATABASE).....	60
FIGURE 3.7: TRACE RECONSTRUCTION (A) ORIGINAL (B) 99.9% OF NCE (C) 99% OF NCE.	61
FIGURE 4.1: SINGLE EDSW ESTIMATION WITH PSO	67
FIGURE 4.2: MULTIPLE EDSW ESTIMATION	70
FIGURE 4.3: AN ORIGINAL NORMALIZED SEISMIC TRACE SAMPLE.....	72
FIGURE 4.4: THE MODEL COMPONENTS s1, s2, s3, s4, s5 OF A TRACE SAMPLE	73
FIGURE 4.5: THE RECONSTRUCTED SAMPLE TRACE WITH M = 1, 2, 3, 4, 5	74
FIGURE 4.6: THE RESIDUAL OF MODELLING SAMPLE TRACE WITH M = 1, 2, 3, 4, 5	75
FIGURE 4.7: ANOTHER SAMPLE TRACE RECONSTRUCTION.....	76
FIGURE 4.8: THE ENERGY OF THE NORMALIZED RESIDUALS FOR SAMPLE TRACE.....	77
FIGURE 4.9: NORMALIZED RESIDUAL ENERGY COMPARISON AS FUNCTION OF COMPRESSION RATIO.....	78
FIGURE 4.10: THE RECONSTRUCTION QUALITY WITH RESIDUAL QUANTIZATION.....	79
FIGURE 4.11: THE RECONSTRUCTION QUALITY WITH RESIDUAL QUANTIZATION OF DCT AND EDSW	83
FIGURE 4.12: THE RECONSTRUCTION QUALITY WITH RESIDUAL QUANTIZATION OF THE PROPOSED METHOD J = 50, 100, 150, 200	84
FIGURE 5.1: AUTO-ENCODER WITH A SINGLE HIDDEN LAYER	90
FIGURE 5.2: A RESTRICTED BOLTZMANN MACHINE	91
FIGURE 5.3: SEISMIC ACQUISITION SYSTEM (A) GEOPHONES AND SHOTS LAYOUT (B) FIRST SHOT (C) SECOND SHOT (D) FINAL (18TH) SHOT	97
FIGURE 5.4: SAMPLE OF THE FEATURES LEARNED.....	98
FIGURE 5.5: SAMPLE OF PRINCIPAL COMPONENTS OF THE DATA	98
FIGURE 5.6: LEARNING RATE DURING SUPERVISED TRAINING	99
FIGURE 5.7: PSNR OF TRACES FROM EACH SHOT OF EAST TEXAS DATA.....	102

FIGURE 5.8: PSNR OF THE DATA AS A FUNCTION OF NUMBER OF UNITS IN CODE LAYER	103
FIGURE 5.9: PSNR OF EACH SHOT OF UTAM AVENUE DATASET.	104
FIGURE 5.10: PSNR OF LPC AND THE PROPOSED METHOD.	107
FIGURE 5.11: RECONSTRUCTION SAMPLE	108
FIGURE 5.12 PREDICTION BASED METHOD (A) SENDER SIDE AND (B) RECEIVER SIDE	110
FIGURE 5.13 A DNN WITH MULTIPLE HIDDEN LAYERS	112
FIGURE 5.14 A SUBSET OF THE FEATURES LEARNED BY AN RBM ON A FULL-TRACES. THE SAMPLES ARE THE FIVE FEATURES WITH THE HIGHEST L_2 NORM.....	117
FIGURE 5.15 SIGNAL ENERGY DISTRIBUTION OVER THE FREQUENCY COMPONENTS.....	117
FIGURE 5.16 ORIGINAL DATA AND PREDICTION RESIDUAL DISTRIBUTION.....	118
FIGURE 5.17 RECONSTRUCTION SAMPLE (A) ORIGINAL (B) RECONSTRUCTED USING DNN (C) RECONSTRUCTION ERROR	120
FIGURE 5.18 A STACKED AUTO ENCODER	123
FIGURE 5.19 SAE-MBP WITH (A) MULTIPLE BPs FOR PRE-TRAINING (B) FINAL BP FOR FINE-TUNING	125
FIGURE 5.20 SRBM-SBP WITH (A) MULTIPLE RBMS FOR PRE-TRAINING (B) STACKING AND A FINAL BP FOR FINE-TUNING..	126
FIGURE 5.21 AN AUTO-ENCODER ELM	128
FIGURE 5.22 SAE-ELM WITH (A) SEQUENTIAL TRAININGS FROM AE-ELM1 TO AE-ELML (B) STACKING THE AE-ELMS ARRANGING WEIGHTS $W_1 \dots W_L$ AND $\beta_1 \dots \beta_L$	131
FIGURE 5.23 SEISMIC ACQUISITION LAYOUT.....	134
FIGURE 5.24. WEIGHTS OF (A) BP (B) RBM-BP (B) (C) ELM	136
FIGURE 5.25. A RECONSTRUCTION SAMPLE WITH 10:1 COMPRESSION RATIO	141
FIGURE 5.26. SNR AS A FUNCTION OF COMPRESSION RATIOS FOR ALL METHODS	142

LIST OF ABBREVIATIONS

ANN	: Artificial Neural Networks
DCT	: Discrete Cosine Transform
DNN	: Deep Neural Networks
EDSW	: Exponentially Decaying Sinusoidal Wave
ELM	: Extreme Learning Machine
NN	: Neural Networks
PCA	: Principal Component Analysis
PSO	: Particle Swarm Optimization
RBM	: Restricted Boltzmann Machine
WHT	: Walsh Hadamard Transform

ABSTRACT

Full Name [HILAL HUDAN NUHA]
Thesis Title [DATA COMPRESSION OVER SEISMIC SENSOR NETWORKS]
Major Field [ELECTRICAL ENGINEERING]
Date of Degree [13 December 2018]

Seismic surveys produce substantial amounts of data. Such data needs to be sent regularly to main processing center. The transmission of this data using current infrastructure is space and computation intensive. This thesis proposes a suit of compression techniques to reduce the data volume while maintaining near lossless quality. More specifically, the approaches developed in this thesis include: Distributed Principal Component Analysis (DPCA), model based seismic signal compression, and deep machine learning.

PCA has traditionally been used as a robust dimension reduction technique. Here, PCA is reformulated under a distributed framework, in which the individual sensors do not need to transmit their own basis to the fusion center. Each sensor uses its own data to update the second order statistics accumulated from previous sensors and forward these to the next sensor in the network. The fusion center estimates the global basis and broadcasts them back to the individual sensors for projection. The proposed approach results in substantial reduction in data transmission across the network as compared to the local PCA. The analysis of experimental results using real seismic data show improved compression ratios and reduced computational load.

In contrast to transform based techniques, a model based approach is developed. In this case, a seismic trace is modeled as a superposition of Exponentially Decaying Sinusoidal

Waves (EDSWs). Each EDSW is considered as a component in the model and is represented by few parameters. These parameters are estimated using a Particle Swarm Optimization (PSO) algorithm on a component-by-component basis until a certain level of significance (residual energy) is achieved. Once all the parameters are estimated, they are transmitted instead of the trace data, thus achieving a high compression ratio. To achieve near lossless compression, the residuals are encoded using fixed length code-word scalar quantization. However, high order of the model and the use of PSO lead to increased computational complexity.

In recent times, deep neural networks (DNN) have gained a lot of popularity over diverse applications. Here, a seismic data compression scheme is developed using a DNN model. The DNN is trained using the traditional Restricted Boltzmann machine (RBM) algorithm. Different DNN architectures were evaluated to obtain an appropriate configuration for the application of interest. DNN is capable of achieving a predetermined compression ratio by selecting the number of units in the middle hidden layer (code layer). To further improve the training time of the DNN, Extreme Learning Machine (ELM) is introduced instead of RBM. Furthermore, ELM also improves the robustness of the DNN without compromising the quality of the reconstructed traces.

The performances of the developed approaches are compared with several classical techniques including Discrete Cosine Transform (DCT) and the Linear Predictive Coding (LPC) models. Both synthetic and real seismic data sets were used to validate the obtained results. The developed approaches are shown to achieve better reconstruction quality with higher compression ratios than comparative methods.

ملخص الرسالة

الاسم الكامل : هلال هودان نوها

عنوان الرسالة : ضغط بيانات شبكات أجهزة استشعار المسح الجيوفيزيائي

التخصص : الهندسة الكهربائية

تاريخ الدرجة العلمية : 13 December 2018

عمليات المسح الجيوفيزيائي تنتج كميات كبيرة من البيانات التي يجب ان ترسل بانتظام إلى مركز المعالجة الرئيسي. إن نقل هذه البيانات باستخدام البنية التحتية الحالية يتطلب ذاكرة كبيرة وعمليات حسابية مكثفة. تقترح هذه الأطروحة مجموعة من تقنيات الضغط لتقليل حجم البيانات مع الحفاظ على الجودة بدون خسارة تقريباً. وبشكل أكثر تحديداً، تتضمن الطرق المطورة في هذه الرسالة ما يلي: تحليل المكون الرئيسي الموزع (DPCA) و ضغط الإشارة الجيوفيزيائية القائمة على نموذج ، والضغط باستخدام التعلم الآلي العميق.

لقد تم استخدام الـ PCA تقليدياً كأسلوب فعال لتقليل الأبعاد. هنا، يتم إعادة صياغة PCA في إطار موزع بحيث لا تحتاج أجهزة الاستشعار الفردية لنقل أساسها الخاص إلى مركز الدمج. يستخدم كل مستشعر بياناته الخاصة لتحديث إحصاءات من الدرجة الثانية مترابطة من أجهزة الاستشعار السابقة وإرسالها إلى المستشعر التالي في الشبكة. ويقوم مركز الدمج بحساب الأساس العام ويعيد بثه إلى كافة أجهزة الاستشعار الفردية للاستخدام في إسقاط بياناته. يؤدي النهج المقترح إلى انخفاض كبير في كمية البيانات المنقولة عبر الشبكة مقارنةً بـ PCA المحلية. يظهر تحليل النتائج التجريبية باستخدام بيانات مسح جيوفيزيائي حقيقية تحسن نسب الضغط وانخفاض الحمل الحسابي.

وعلى النقيض من التقنيات القائمة على التحويل، تم تطوير طريقة تعتمد على نموذج. في هذه الحالة، يتم نمذجة إشارة المسح من مستشعر على أنه تراكب لموجات جيبيّة متناقصة (EDSWs). يعتبر كل EDSW مكوناً في النموذج ويتم تمثيله بواسطة معاملات قليلة. يتم تقدير هذه المعلمات باستخدام خوارزمية تجسيم سرب الجسيمات (PSO) على أساس كل إشارة على حدة حتى يتحقق مستوى معين من الدقة (الطاقة المتبقية). وبمجرد أن يتم تقدير كل المعلمات، يتم نقلها بدلاً من بيانات إشارة المسح، وبالتالي يتم تحقيق نسبة ضغط عالية. ولتحقيق الضغط بدون فقدان معلومات، يتم ترميز المتبقي من الإشارة باستخدام ترميز بكلمة ترميز ثابتة. ولكن، يؤدي استخدام ترتيب عالي للنموذج واستخدام PSO إلى زيادة التعقيد الحسابي.

في الآونة الأخيرة ، اكتسبت الشبكات العصبية العميقة (DNN) ، الكثير من الشعبية على تطبيقات متنوعة. هنا، يتم تطوير نظام ضغط بيانات المسح الجيوفيزيائي باستخدام نموذج DNN. يتم تدريب DNN باستخدام خوارزمية آلة بولترمان التقليدية المقيدة (RBM). تم تقييم شبكات DNN المختلفة للحصول على التكوين المناسب للتطبيق الحالي. DNN قادرة على تحقيق نسبة ضغط محددة مسبقاً عن طريق اختيار عدد الوحدات في الطبقة الوسطى المخفية (طبقة الكود). لمزيد من تحسين وقت تدريب DNN ، تم تقديم آلة التعلم المتطرفة (ELM) بدلاً من RBM. علاوة على ذلك ، تعمل ELM أيضاً على تحسين وثوقية DNN دون المساس بجودة إشارة المسح الجيوفيزيائية التي أعيد بناؤها.

تم مقارنة أداء الطرق المطورة في هذه الأطروحة مع العديد من التقنيات الكلاسيكية بما في ذلك تحويل كوسين المنفصل (DCT) والنماذج الخطية للتشفير (LPC). استخدمت كل من بيانات المسح الصطناعية والحقيقية للتحقق من صحة النتائج التي تم الحصول عليها. تظهر الطرق المطورة أنها تحقق جودة أفضل للإشارات المستردة مع نسب ضغط أعلى من الطرق المقارنة.

CHAPTER 1

INTRODUCTION

World yearly energy consumption is forecasted to be over 600 Quadrillion Btu on 2020 [1]. Oil and other fossil-fuel based energy sources still dominate the energy consumption by more than 70% of world consumption [2]. However, the oil demands tend to outnumber the discovered oil volumes. Figure 1.1 confirms this trend showing that discovered oil volume additions have lagged behind the oil demand since the early 1980 [3]. This trend forced new oilfield discoveries to become of undeniable importance. To discover new oilfields, seismic surveys must be performed to locate oil reservoirs. However, seismic surveys produce huge amount of data to maintain accuracy and precision since inaccurate drilling is very costly.

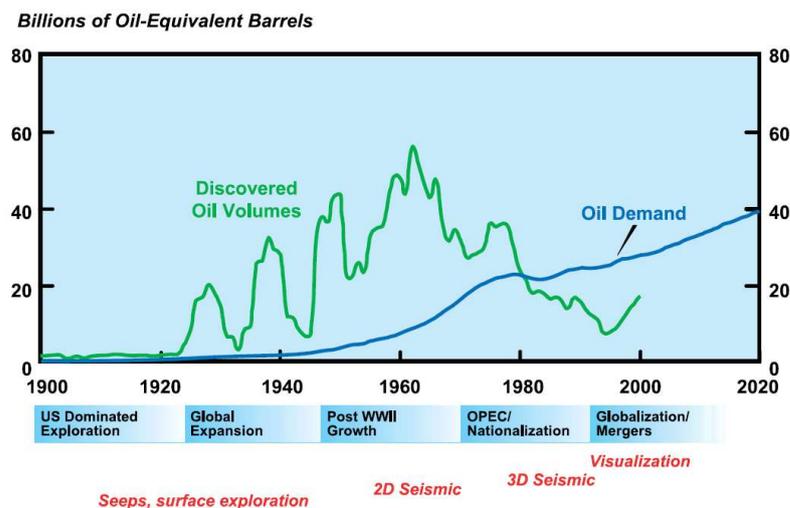


Figure 1.1 Oil supply and demand

This thesis proposes a suit of approaches to solve the problem of dealing with the enormous seismic data volume by using data compression techniques. In the next section we provide a brief introduction of the seismic data acquisition process and its effects on the total seismic data volume. Problem statements and the proposed solutions are then presented in the end of chapter.

1.1 Background

The seismic reflection is the most widely used and well-known geophysical technique over many surveying schemes. Seismic data is processed to show the details of geological structures on scales from the top tens of meters of the Earth crust to its inner core [4] [5]. Seismic exploration of the earth's crust is used extensively not only on land but also offshore to identify oil and gas deposits in subterranean earth formations [6]. The fundamental of geophysical survey is to apply a source, such as an explosion and the generated signals then propagate through Earth layers. When the generated signals encounter an interface between two layers with different acoustic impedances, the signals are reflected or refracted. At the surface, the reflected signals are then recorded by an array of sensors, transmitted to data center, and stored for processing and interpretation. In practice, adding more sensors leads to a higher quality of the received data [7].

The goal of seismic exploration is to characterize the hydrocarbon subsurface geological features. Those features such as salt domes, reefs, deltaic sands, etc. are naturally three-dimensional bodies. The seismic exploration techniques commonly involved the generation of shot using an acoustic wave from an explosion or a vibroseis truck as

illustrated in Figure 1.2. The generated wave travels down to the earth and is reflected the by the sub-surface layers back to surface where arrays of sensors or geophones record the reflected waves. The recorded reflection waves are called as seismic traces. The seismic traces are then transmitted by the sensors to a fusion center where the traces are stored.

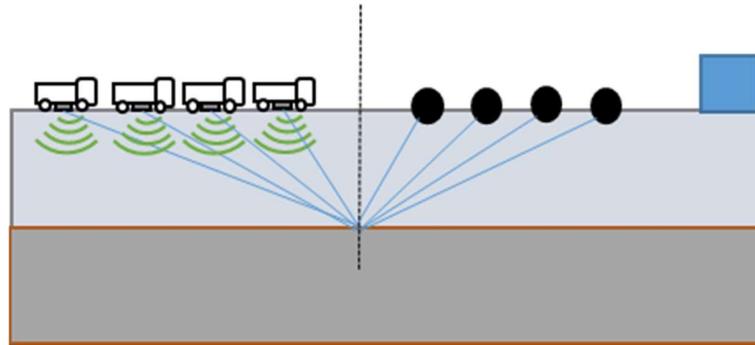


Figure 1.2 A typical seismic acquisition

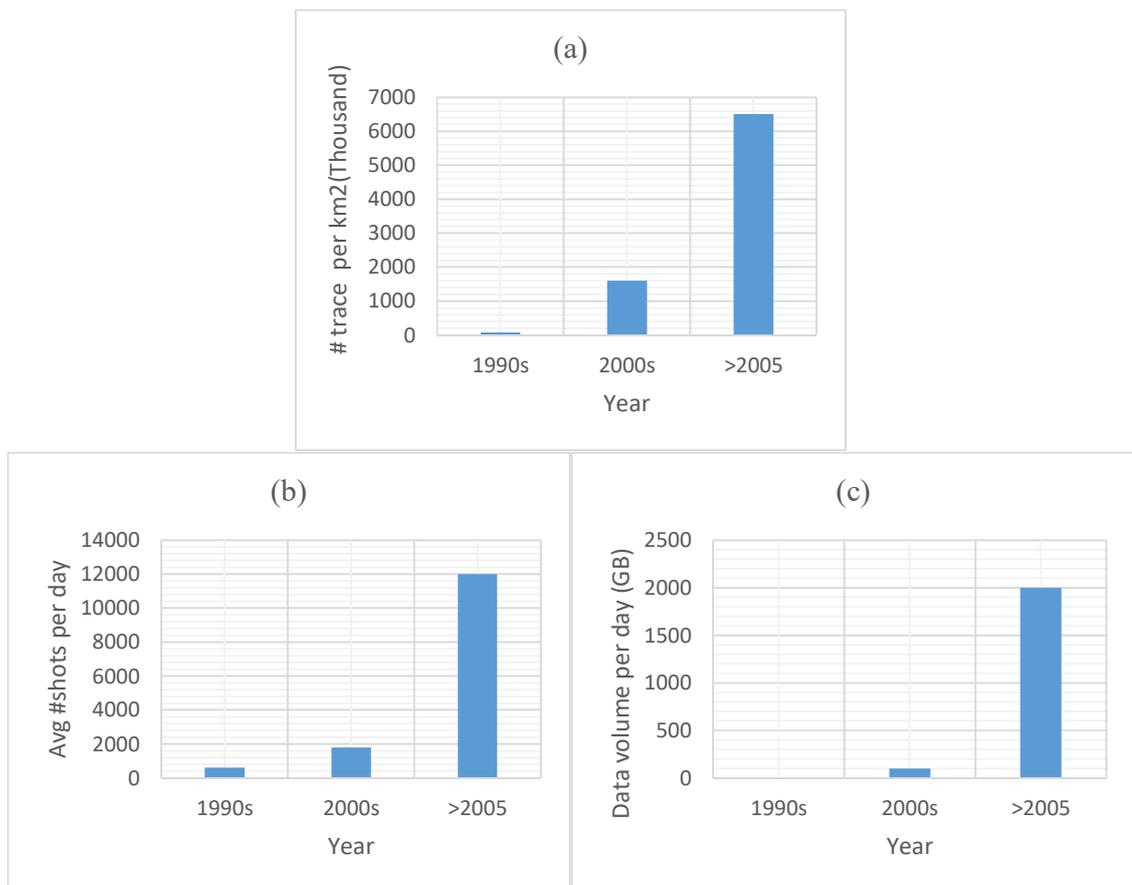


Figure 1.3 Trends in seismic data acquisition (a) data density (b) number of shots (c) data volume

In a typical seismic acquisition survey, in which data is acquired by generating a vibration source at one location and recording at a receiver the resulting vibrations in the earth can reach 100 terabytes [8]. That number increases in the range of multiple hundreds of terabytes due to with acquisition using more advanced surveys that involved more sensors instead of a traditional one, and emerging new techniques in data acquisition such as wide azimuth and full azimuth acquisitions as shown by Figure 1.3 [9]. Such seismic data requires very large storage facilities and a very high cost for transmission. The seismic data volume also demands considerable transmission bandwidth to transfer the data from a seismic survey location to a central processing location. Also for many situations where real time data processing is required, data compression is the key since it is possible only if the data is transmitted very quickly. Therefore, seismic data compression is desirable in order to save storage space and transmission time. It is worth noticing that the original data contains high redundancy, which motivated this work to develop compression algorithms reducing the data size by taking advantage of the redundancy while maintaining the reconstruction quality.

1.2 Problem Statements

Following the discussion above, we list below the foremost main problems discussed in this thesis:

1. Seismic surveys produce large data size. Most seismic data are saved as 32 bit integer or floating point and thousands of recordings are collected by each sensor. A modern seismic survey involves thousands sensors therefore data

produced by a typical seismic survey may exceed 10 TB. This may cause delay in further processing since data to be transmitted are huge. Current technology, especially in remote area with no network infrastructure, uses physical device to store the data and deliver the data physically. Even with existing network of 100Mbps, transmission of 10TB data requires days to be completed therefore compression techniques are required to minimize data transmission duration.

2. Compression of seismic data introduces quality degradation. Lossy compression techniques introduce distortion in the reconstructed seismic trace, which results in inaccurate sub-surface seismic images. Inaccurate seismic images are very risky for petroleum reservoir explorations as these surveys are used in the interpretation stage. Compression techniques must be designed to maintain high signal reconstruction quality.
3. Communication cost between sensors and fusion center can be very high. Sensors record large amounts of shot reflections that must be transmitted to the fusion center. To exploit temporal correlation in the data collected by the sensor, the sensor must accumulate the data before compressing them. Compression techniques produce an overhead in the form of a dictionary, a codebook, or a projection basis generated from the collected data. This obviously can generate a high communication cost, which needs to be considered in developing compression scheme.

1.2.1 Research Objectives

To solve the problems mentioned above, we formulate the following research objectives in this thesis:

1. Developing robust techniques for seismic data compression. By the end of this research, a suit of techniques for seismic data compression are be developed and tested on real data
2. Enhancing the developed techniques to maintain the quality of the reconstructed data. High seismic signal quality must be maintained since small errors can lead to unpredictable effects in the interpretation stage.
3. Developing a communication scheme for reducing the communication cost between the sensors and the fusion center. By reducing the size of the data, transmission time becomes acceptable. Compression techniques, however, usually introduce several overheads to be transmitted, therefore efficient communication mechanisms are a necessity.

1.2.2 Main Research Directions

The main research directions in this thesis are listed below:

1. Develop a Distributed PCA (DPCA) technique for seismic data compression. DPCA offers an orthogonal linear transform basis for all sensors. A communication scheme is developed to enable the fusion center to construct global Principal Components (PCs) from the statistics of all sensors. A single set of global PCs is used for all sensor instead of multiple sets of PCs from each sensor to reduce the overhead.

2. Develop a model based approach to reduce data redundancy. Modelling seismic data as a mathematical model leads to a smaller data representation. Seismic data will be represented by a small number of parameters that preserve the most important information from seismic signals. Evolutionary computing techniques are applied to estimate the parameter of the model. Estimating the model parameters requires huge computational efforts since search space of each parameter can be very large. Random noise may also prevent the estimation process to achieve a good optimum.
3. Develop Deep Neural Networks (DNN) for seismic data compression. The DNN offers generalization capability and is able to capture the important features from training data that will be useful for compression. This method is capable to exploit the non-linearity of the data that cannot be extracted by linear transforms.
4. Developed methods will be evaluated in terms of compression ratio and reconstruction quality. Each method may have its own advantages and disadvantages. The developed techniques are also compared to existing techniques for the sake of benchmarking.

1.3 Major Contributions

The main contributions of this thesis are:

1. Development of a distributed principal component analysis for seismic data compression.
2. Development of a model based techniques for seismic data compression.

3. Development of an infield seismic data compression scheme using restricted Boltzmann machine.
4. Development of a near lossless prediction based seismic data compression using DNN.
5. Development of fast learning scheme for DNN using Extreme Learning Machine (ELM) for seismic data compression.

1.4 Thesis Organization

The rest of this thesis is organized as follows:

Chapter 2 provides an extensive literature review of existing approaches for seismic data compression, data formats, metrics for compression ratios, and reconstruction quality measures.

Chapter 3 provides the development of the distributed PCA for seismic data compression, mixture model derivation, its implementation for seismic sensor networks, and comparison with the other linear transformation techniques.

Chapter 4 provides the development of the model based seismic data compression algorithm, model derivation, parameter estimation using Particle Swarm Optimization (PSO), residual quantization and encoding, and performance comparison with the Discrete Cosine Transform (DCT).

Chapter 5 provides development of machine learning approaches for seismic data compression. Different types of DNN are developed using RBM for seismic data

compression. In addition, a new architecture of DNN is developed that can be trained in a short time using ELM.

CHAPTER 2

LITERATURE REVIEW

For many years, it has been common to perform seismic exploration operation to discover new sources of oil, gas, and other minerals. The techniques for seismic exploration typically involve the generation of an acoustic wave at the surface of the earth or the ocean. The generated wave travels downwardly into the earth and is reflected upwardly from the sub-surface layers of interest towards the surface of the earth where its return is recorded by arrays of sensors. The sensors typically record analog electrical signals which are converted to digital form and are stored. Seismic data processing is then performed to produce an image of the sub-surface structure of the earth which can be interpreted by geophysicists to identify accurate locations of oil or gas reservoirs as well as other valuable minerals [10].

Along with the rising demand for oil and gas, it is desirable to produce higher quality of images, which in turn requires the acquisition of more seismic data. The emergence of more advanced seismic data acquisition techniques have led to the production of data sets capable of burdening network capacities of the latest communication technologies. The development of networked and distributed computing environment has aggravated most problems of dealing with seismic data that can approach or even exceed petabyte levels [11]. Obviously, it is desired to reduce the volume of data to be stored and transmitted as

long as this can be achieved without loss of significant accuracy [12]. For that purpose, data compression is considered as an important solution to efficiently handle large seismic data volume.

Technically, the term data compression can be defined as a method to reduce the number of bits per sample required to represent a specific amount of information from a digital signal. Despite being synonymous, there is a clear distinction between data and information. In fact, data are the mediums that convey information. Therefore, the same amount of information can be expressed in various amounts of data. Some data contain irrelevant information or simply repeat already known information. Those kinds of data are thus said to contain redundancy [13].

Depending on the purpose, the amount of relevant information contained in the data typically varies from application to application. In other words, data with relevant information for one application may be less relevant for another application. Data compression can be achieved by reducing the redundancy or exploiting the irrelevancy. Lossless data compression is achieved by reducing the redundancy without taking advantage of reducing the irrelevancy such that the compression does not introduce any distortion. To the contrary, a distortion occurs when irrelevancy is exploited and redundancy is reduced hence this is called lossy data compression.

Due to the removal of irrelevancy, lossy methods generally achieve higher compression ratios than that of lossless methods. In practice, for numerical data, a compression ratio greater than 2:1 requires a lossy method. To the contrary, compression seismic trace headers that contain text data can be easily compressed with significantly higher

compression ratios than 2:1 by lossless methods [14]. This is due to the fact that seismic trace headers are mostly text files that contain frequently repeated symbols and patterns of symbols while the numerical data in a seismic trace data is far less structured.

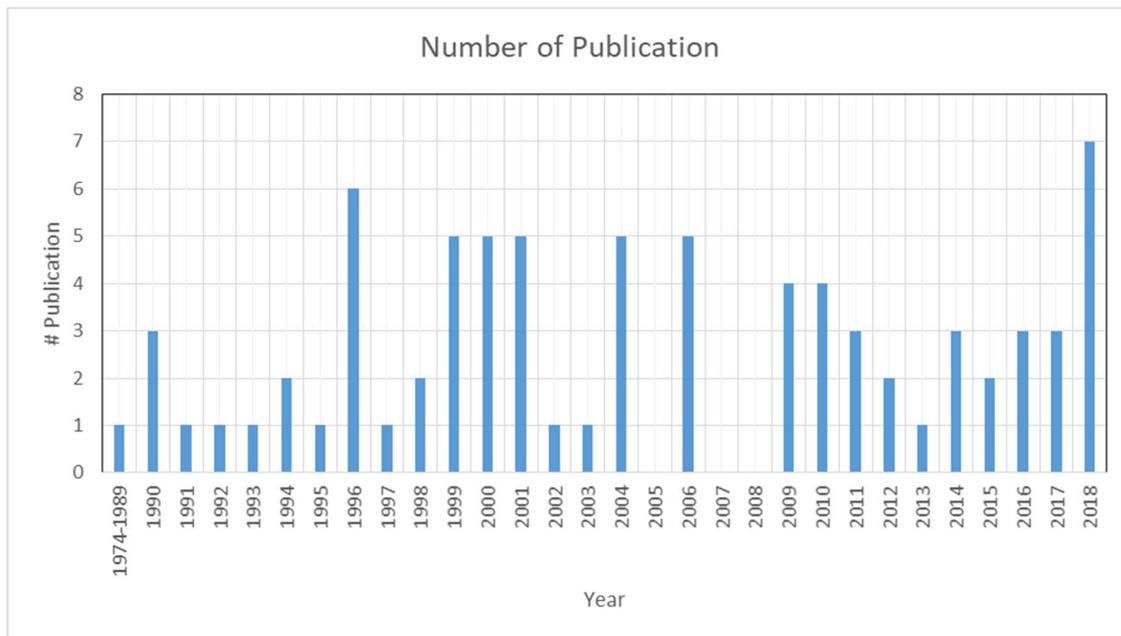


Figure 2.1: Total number of year-wise publications since 1974 (Scopus).

In lossy compression, the fundamental problem is to achieve the minimum level of distortion for a given compression ratio, or equivalently, to achieve a given acceptable level of distortion with the maximum compression ratio. This problem is formulated by the rate-distortion function [15].

Compression ratios can be classified into low and high. Seismic data trace are commonly represented by 32 bits per sample (single precision floating point number format) to maintain the accuracy of the data. Therefore, for seismic data, it is said that bit rates lower than 2 bits per sample are low bit rates that lead to high compression ratios, while high bit-rates start from 2 bits per sample leading to low compression ratios. Therefore, a

compression ratio of 16:1 is the threshold between low and high compression ratios of seismic data [13].

The idea of seismic data compression was initiated in 1974 [16], but the idea did not gain much attentions from researchers and practitioners. Figure 2.1 shows the barcharts of year wise publications since 1974 for different types of seismic data compression. The data is collected from Scopus using a combination of search keyword seismic data compression (i.e. seismic, data compression). Overall, we notice major increase in early 1990 but the trend does not show any annual increase in this field. However, this area is still active as new geophysical acquisition techniques like 3D and 4D schemes arise along with rapidly growing computing power in the geophones or sensors. This chapter attempts to summarize the efforts from past decades of seismic data compression scheme by presenting a survey of the literature in the area.

This chapter is organized as follows: Section 2.1 provides a brief introduction to seismic exploration. Section 2.2 describes the typical format for seismic data. Section 2.3 provides the different classes of seismic data compression techniques with their advantages and disadvantages. We then provide metrics used in the seismic data compression in section 2.4. Finally, we conclude this chapter in section 2.5.

2.1 Seismic Exploration

The goal of seismic exploration is to characterize the hydrocarbon subsurface geological features. Those features such as salt domes, reefs, faults, deltaic sands, etc. are naturally three-dimensional bodies. A two dimensional (2D) seismic section, which is a cross-section of a three-dimensional (3D) seismic section, describes cross-section subsurface

properties until certain depth of a line of sensor. 3D surveys provide better and more detailed subsurface image that lead to more accurate interpretation. One of the benefits of 3D surveys is on the case where the reflector layer is dipping as shown in Figure 2.1.

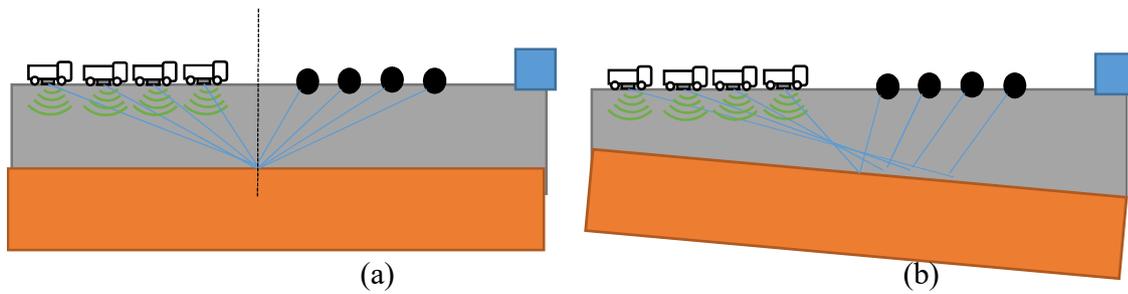


Figure 2.2: CMP (a) regular reflector (b) dipping reflector

Unlike 2D surveys, which are allocated with small number of sensor array lines and a large line spacing up to 1 km, 3D surveys use large number of sensor array lines with small line spacing up to 25m. Geophones are allocated around 25m away from other. A few hundred thousand to hundred million of traces are collected during 3D survey where over ten thousands of shots are generated every day.

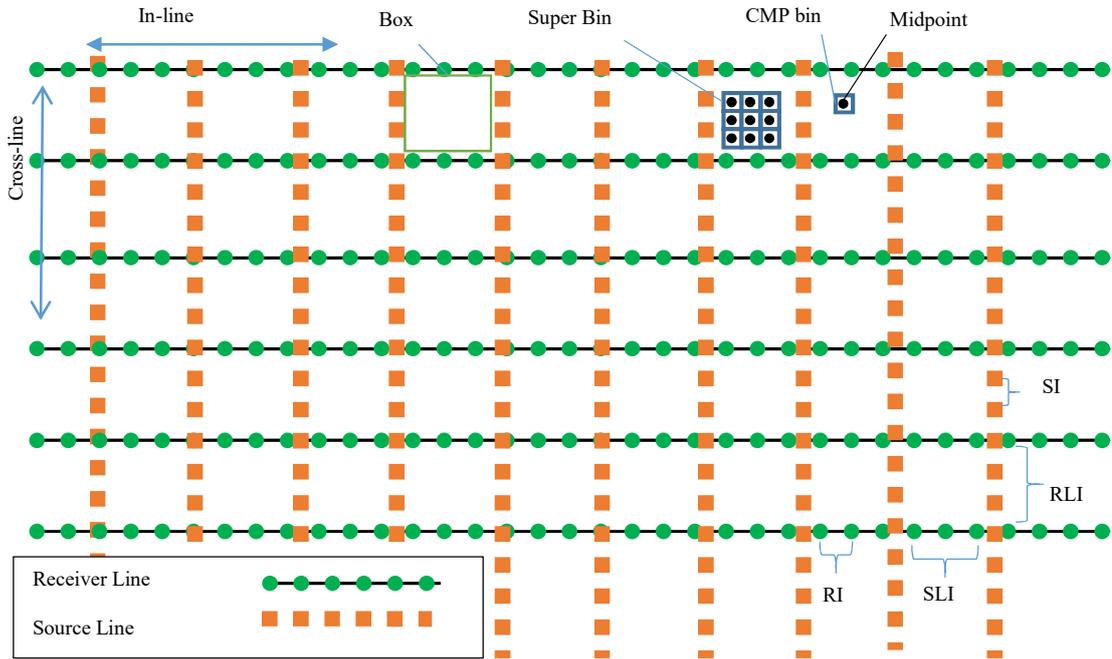


Figure 2.3: Typical Receiver Layout

A typical survey sensor layout is depicted by Figure 2.3. Receiver arrays are arranged sequentially along In-line axis. Receiver Line Interval (RLI) denotes the distance between two parallel receiver lines. Receiver Interval (RI) is the distance between two sensors. Cross-line is the axis perpendicular to the receiver lines. Shots are generated along the source line, which are depicted as red squares in Figure 2.3. Distance between two consecutive shots is denoted as the Source Interval (SI) and Source Line Interval (SLI) is the distance of two parallel shots line. A Common Mid-Point (CMP) bin is a small rectangular area that contains all traces with the same midpoint. A typical size of the CMP bin is $\frac{1}{2}RI \times \frac{1}{2}SI$. A group of CMP bins is called a super bin. Box or unit cell is an area bounded by two neighboring receiver lines and two shot lines.

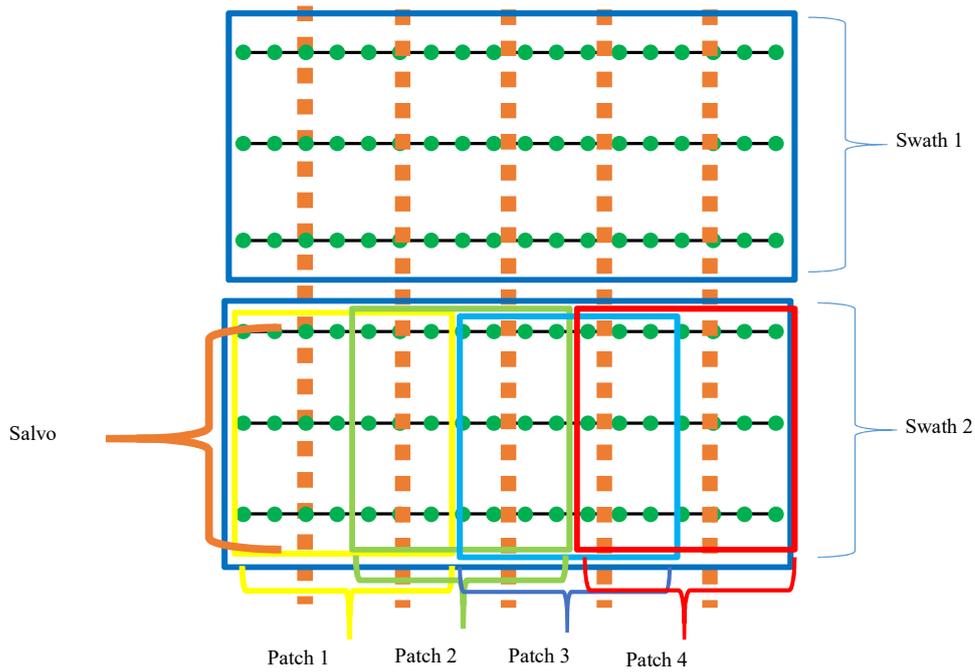


Figure 2.4: Swath Shooting

To obtain adequate data from a survey, Swath Shooting method is applied as illustrated in Figure 2.4. Survey area is divided into several non-overlapping areas called swaths. Each swath contains several overlapping rectangular areas named patches. A patch corresponds to a set of receivers in a rectangular area that record from a set of source points. The number of source points within a patch is called salvo.

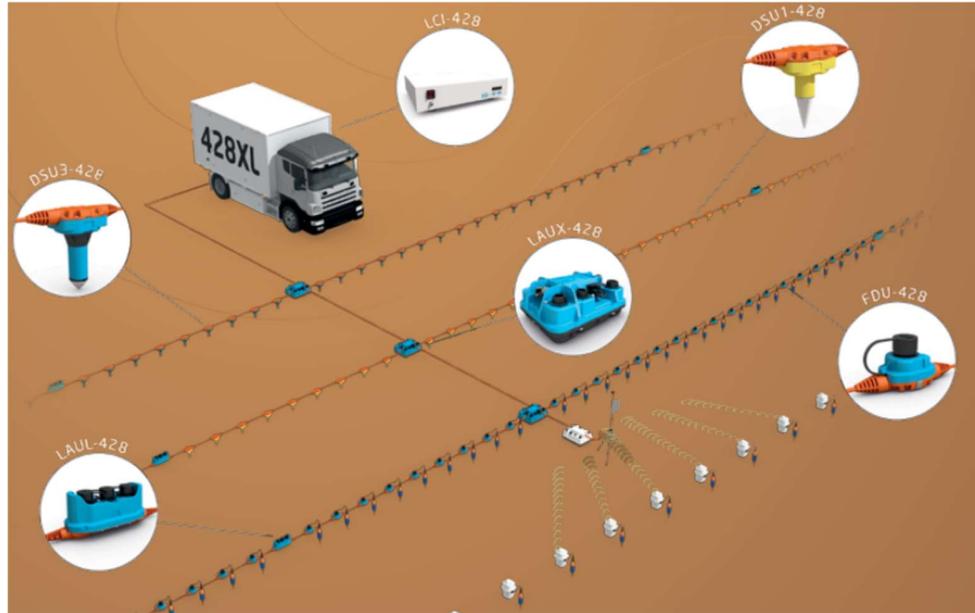


Figure 2.5 428XL Land seismic acquisition system

One example of sensor and network devices connection from Sercel 428XL seismic acquisition system can be seen in Figure 2.5 [17]. Digital Sensor Unit (DSU) is the receiver arranged along the receiver line. Each DSU is provided by a channel with 8Mbps data connection. Since the receiver line extends for a very long distance, signal must be amplified by Land Acquisition Unit Line (LAUL) that connects two receiver lines. At the middle of each receiver line, a Crossing Line Acquisition Unit (LAUX) is used to connect to other LAUX and finally lead to the fusion center. The most recent technology enables 10 sequential LAUXs. Some variant of LAUXs are designed for wireless sensor connections.

2.2 Seismic Data Format

A seismic exploration commonly uses SEG-Y as data format. This format consists of volume header data, trace header data, and trace data as depicted in Figure 2.6. Among these segments, the total length of the volume header data is 3600 bytes, and the first 3200 bytes are encoded using extended binary coded decimal interchange code (EBCDIC) character, which is used to store some description about the seismic data volume [18]. The sampling points, data format, the sampling interval, measurement units, and other key SEG-Y file information are stored in the second 400 bytes. Most of byte positions of these fields are stored in fixed location. For examples, the number of data traces is located at 3213-3214, the sample interval of original field recording is stored in 3219-3220, and number of samples or sampling points per data trace are recorded in in 3221-3222.

A trace sequence number (beginning from 1), sampling interval, number of sampling points in the trace, shot ground elevation, and some other information related to the trace data is typically stored in the trace header data [19]. The discrete amplitude value obtained from the sampled seismic signal after a certain time interval, is stored in trace data, and each sample point typically consumes 4 bytes for single precision [20]. The last revision of this format allows up to 65535 additional 240 byte trace headers with bytes 233-240 of each trace header reserved for trace header names.

In the earlier SEG-Y standards, all binary values were defined as using "big-endian" byte ordering. The latest revision allows "pairwise byte-swapped" and "little-endian" byte ordering primarily for I/O performance. Trace data sample values are either floating point

or integers numbers. There are four major number formats allowed namely unsigned integer, two's complement integer, IBM floating point, and IEEE floating point.

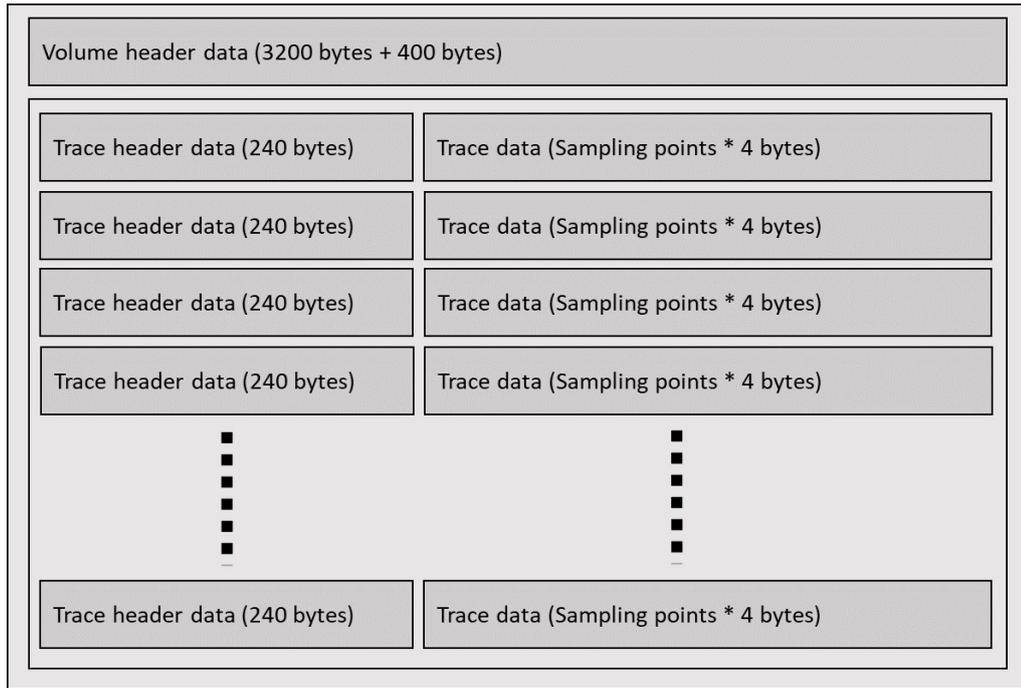


Figure 2.6: SEG-Y file format

2.2.1 Unsigned Integer

Unsigned integer is the simplest number format. SEG-Y format allows 1,2,3,4, and 8 bytes unsigned integer. The B bytes unsigned format has range from 0 to $2^B - 1$. The bit stream $b_{4B-1} \dots b_0$ can be converted to unsigned integer simply using the following expression.

$$\text{Value} = b_{4B-1} * 2^{4B-1} + \dots + b_0 * 2^0 \quad (2.1)$$

2.2.2 Two's Complement Integer

Two's Complement integer is the only number format for integer. SEG-Y format also allows 1,2,3,4, and 8 bytes two's complement integer. The B bytes unsigned format has range from -2^{B-1} to $2^{B-1} - 1$. The bit stream $b_{4B-1} \dots b_0$ can be converted to two's complement integer simply using the following expression.

$$\text{Value} = (b_{4B-1} * 2^{4B-1} + \dots + b_0 * 2^0 + 2^{4B-1}) \bmod 2^{4B} - 2^{4B-1} \quad (2.2)$$

2.2.3 IBM Floating Point

SEG-Y format allows 4 bytes IBM Floating Point. The normalized range of representable numbers is from approx. 5.39761×10^{-60} to 7.237005×10^{75} . The bit stream of this format follows 4 byte hexadecimal exponent data as illustrated in Figure 2.7.

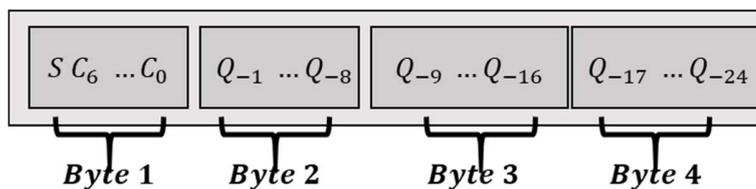


Figure 2.7: 4 byte hexadecimal exponent data

Bit S denotes the sign bit and bitstream $C_6 \dots C_0$ denotes the excess 64 hexadecimal exponent. The magnitude fraction is a 24-bit positive binary fraction denoted by $Q_{-1} \dots Q_{-24}$. The sample value can be calculated as follows.

$$\text{Value} = (S). (QQQQ, QQQQ, QQQQ, QQQQ, QQQQ, QQQQ) \times 16^{CCCC-64} \quad (2.3)$$

2.2.4 IEEE Floating Point

SEG-Y format allows 4 and 8 bytes IEEE Floating Point. The bit stream of this format follows 4 or 8 byte IEEE floating point standard as illustrated in Figure 2.8. The converted value can be converted depending on the value of e and f , where e and f denote the binary value of all C's (exponent) and Q's (fraction).

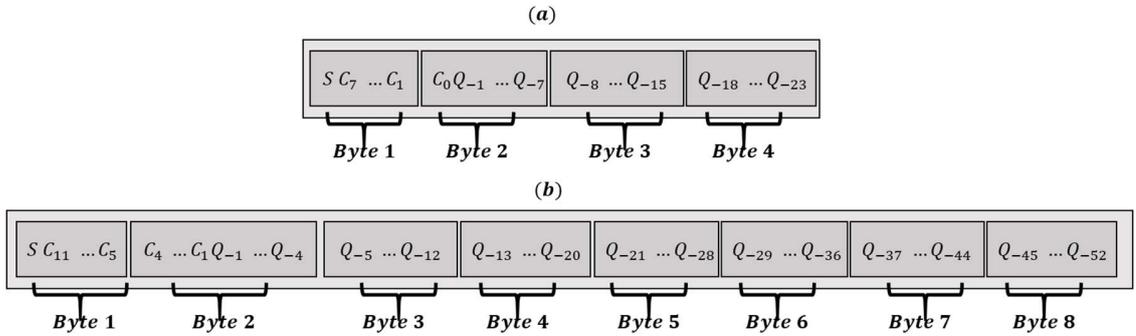


Figure 2.8: IEEE floating point (a) 4 byte (b) 8 byte

On 4 byte floating point, the value of the sample can be calculated as follows.

$$\text{Value} = \begin{cases} \text{NaN(Not - a - Number)} & e = 255, f \neq 0, \\ (-1)^S \times \infty & e = 255, f = 0, \\ (-1)^S \times 2^{e-127} \times 1.f & 0 < e < 255, \\ (-1)^S \times 2^{e-126} \times 0.f & e = 0, f \neq 0, \\ (-1)^S \times 0 & e = 0, f = 0. \end{cases} \quad (2.4)$$

Similar rule can be applied for 8 byte floating point, but with different range.

$$\text{Value} = \begin{cases} \text{NaN(Not - a - Number)} & e = 2047, f \neq 0, \\ (-1)^s \times \infty & e = 2047, f = 0, \\ (-1)^s \times 2^{e-102} \times 1.f & 0 < e < 2047, \\ (-1)^s \times 2^{e-102} \times 0.f & e = 0, f \neq 0, \\ (-1)^s \times 0 & e = 0, f = 0. \end{cases} \quad (2.5)$$

2.3 Seismic Data Compression Categories

In the introduction, we briefly discussed lossless and lossy compression. A large number of seismic data compression techniques have been proposed in the literature. In this section, we list and describe methods used for seismic data compression and its classification. Figure 2.9 shows the main categories of compression techniques currently used. Herein, existing works are classified based on the compression technique utilized for seismic data. We use similar classification of data compression classification presented in [21] for seismic data case. The major categories of seismic data compression includes variable length coding, run length coding, dictionary based, prediction, quantization, and transformation based techniques. In Figure 2.9, the gray shading categories indicate that the methods are typically used as a lossless data compression, whereas, the remaining categories introduced distortions to the data that lead to lossy compressions.

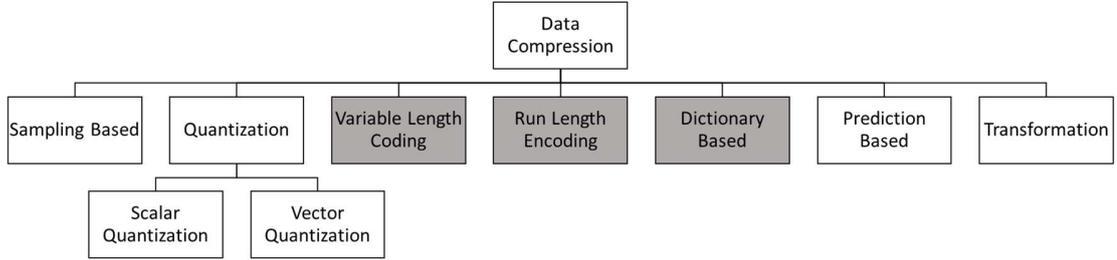


Figure 2.9: Compression Scheme Categories

2.3.1 Sampling Based Compression

A seismic trace is originally a continuous signal sampled with constant sampling period. Intuitively, it is clear that a high sampling rate f_s would result in a better signal representation, but also in many more samples and therefore bigger files. Therefore, the main problem is how often to sample a given wave. The solution to the sampling problem is to sample seismic wave with slightly above the Nyquist frequency, which is twice the maximum frequency f_{max} contained in the wave [22] [21].

$$f_s \geq 2f_{max} \quad (2.6)$$

Sampling based compression is the process of reducing the number of samples to convey information while keeping distortion loss within an acceptable margin. The implementations of this scheme can be found in [23, 24, 25, 16]. The simplest way to reduce the number of samples is by using down-sampling as reported in [16]. The signal is then reconstructed using the Cardinal function interpolation.

The compressed sampling (CS) has been proposed as a potential alternative, since the number of samples required depends on the sparseness of the signal. The CS field has existed for at least four decades, but after 2004, researchers' interest in the field has exploded due to several important results obtained by Donoho [26]. The CS is a novel sampling paradigm that goes against the traditional understanding of data acquisition for instance the Nyquist sampling theorem. Starting from its emergence, CS has attracted researchers to implement it for seismic data compression [23]. CS replaces the traditional paradigm of sample-then-compress with compress-while-sampling. This technique fundamentally requires numerical optimization techniques to recover full-length signals from a small number of randomly collected samples. The compression is achieved via multiplication of a randomized measurement matrix with the original signal vector to obtain a compressed vector. Rubin et.al [24] proposed a CS algorithm for seismic acquisition called randomized timing vector (RTV). The proposed algorithm generates a random vector and uses it to determine sampling timings and generate measurement matrix. Therefore, the algorithm reduces the number of sampling and eventually, the number of samples itself.

2.3.2 Quantization Based Compression

Quantization is a simple approach to achieve lossy compressions. The idea is to create a finite list of symbols, called quantizer, and to modify each of the original data to the nearest symbol. For instance, if the original data consists of real numbers in a certain interval, then each can be rounded off to the nearest integer. It takes a fewer bits to express the integer, so compression is achieved, but it is lossy because it is impossible to

retrieve the original real data from the integers [21, 27, 28, 29]. The simplest quantization is the uniform quantization with fixed step-size as shown in [30]. This approach starts with creating intervals with fixed length and each interval is assigned with a representative value called quantizer. Each sample that falls in an interval is then replaced with its corresponding quantizer. For example, a typical mid-tread uniform quantizer $Q(x)$ with step size Δ can be expressed as

$$Q(x) = \Delta \cdot \text{floor}\left(\frac{x}{\Delta} + \frac{1}{2}\right) \quad (2.7)$$

where $\text{floor}()$ denotes the floor function.

The non-uniform rule uses adaptive step-size by employing the μ law companding. Quantizer can also be created using a clustering algorithm like K-means algorithm. In vector quantization, the finite quantizer is constructed by applying a clustering algorithm to the initial samples.

In seismic data compression, quantization stage is commonly used before the encoding the sampled data to codewords [31, 32, 33, 16, 34]. Uniform and non-uniform rules were applied to compress magnitudes and phases in [35, 36, 37]. Several works also use this approach to quantize the residuals from other techniques by taking the advantage of the reduced variance [38, 23, 24].

The alphabet size of the quantized residual signal from transformation or modelling schemes may grow too large due to high dynamic range of the residual signal. The oversampling recursive least square (ov-RLS) is proposed in [25] to alleviate the problems encountered in large alphabet signal compression using single bit

oversampling. This approach is based on oversampling but a representation with fewer number of bits. For a fixed reconstruction quality, the sampling frequency should be increased exponentially to reduce the number of quantization levels, which results in a tremendous growth in the raw bit rate from ADC, making it feasible in most applications. The basic idea of data compression by tackling large alphabet signal is to create 1 bit samples from signal. It can be achieved by using a much higher sampling frequency on a modified 1-bit signal quantization scheme with a dither function which results in a non-uniform signal-dependent sampling. The dither function is a periodic saw-tooth function that is known at both the encoder and decoder for compression-decompression purpose [39]. For seismic image, Hamood et.al [40] proposes a functional quantization (FQ). The FQ quantizes the entire sample path of the seismic waveform in a target function space, instead of individual sample quantization.

2.3.3 Run Length Encoding

The basic idea of run length encoding (RLE) is that in many types of data, adjacent samples are often correlated, so there may be sequences of identical symbols which may be exploited to compress the data. Each sequence of samples with the same value is encoded as a pair consisting of run length and sample value. The run length can be represented as one byte, allowing for sequence length of up to 255 samples. The sample value can be encoded using original encoding style like integer or characters. This technique and its variants have been implemented in many works on seismic data compression to encode quantized samples [38, 41]. In marine data, a long run of zeros

can be found in the beginning part of the trace produced by delay in the water layer. This property is well exploited for seismic data compression using RLE in [30].

K-RLE is a variant of the variant that allows some variability on the input data stream during data encoding [24]. This method encodes the input signal by using an acceptable threshold values specified by a parameter K. The regular RLE is often referred as zero-RLE since the threshold value K equals to zero. The lightweight temporal compression (LTC) has similar principal with KLRE where the data is compressed by encoding streams of redundant sequences. Unlike KRLE, a sample is considered redundant is it falls within some range of lines interpolated from previous sample point [23].

2.3.4 Variable-Length Codes

Given a data file where the symbols are drawn from an alphabet, it can be compressed by replacing each symbol with a variable-length code-word instead of fixed-length one. The basic idea of variable-length codes (VLC) is to assign short code-words to symbols with high occurrences and long code-words to symbols with low occurrences. One of the most important and oldest technique in this category is entropy coding. This type of encoding attempts to represent the codeword length to be close to the entropy of the data. The entropy S from a source is given by

$$S = - \sum_{k \in K} P_k \log_2 P_k \quad (2.8)$$

where P_k denotes the probability of symbol k .

The Huffman coding [42] is a type of entropy coding. It is based on the statistics of the data, and it represents the symbols of the alphabet by VLC, depending on their probability of occurrence. The more likely a symbol is, the shorter the code it is assigned.

As mentioned, a seismic data file typically consists of header and seismic trace data. The header of a seismic data file is simply a text file where the alphabet is the set of 128 ASCII characters. Therefore, it can be easily compressed in a lossless fashion as reported in [14]. However, a seismic trace consists of individual samples from a signed integers or a floating points. Both formats are encoded with fixed length codes of 32 bits. Direct implementation of Huffman coding to seismic trace usually leads to low compression ratios due to high entropy. Therefore, implementations of this approach for seismic data are commonly found in the final stage of compression to encode transform coefficients [31, 43, 16, 44, 45]. To improve the reconstruction quality or achieve lossless, it is desired to encode the residual signal from a lossy compression as reported in [32, 33] for subband coding, in [46, 47] for adaptive prediction residual, and in [48, 41] for local trigonometric and wavelet coding.

Implementation of VLC for seismic trace data typically results in a low compression ratio due to its high dynamic ranges. For that reason, Li et.al [19] proposed an improved lossless group compression for decreasing the size of SEG-Y files. First, the file is split into several subgroups and the Gini coefficient is used to analyze the statistics of each subgroup. A combination of Huffman coding and dictionary based compression, called the Deflate algorithm, is then used to compress subgroups with more balanced distributions as indicated by low Gini coefficient.

The arithmetic coding [49] is another type of entropy coding that is commonly used in seismic data compression as presented in [38, 30]. Arithmetic coding solves the problem of assigning integer codes to the individual symbols by using a single long code to the entire data. This algorithm starts with a certain interval, reads the data symbol by symbol, and uses the probability of each symbol to narrow the interval. The interval is represented by its lower and upper limit or by one limit and its length. The narrow intervals constructed by the algorithm require more numbers to specify their boundaries. Data compression is achieved since a high-probability symbol narrows the interval less than a low-probability symbol. Eventually, symbols with high-probability contribute fewer bits to the output [21].

2.3.5 Dictionary based compression

Dictionary-based compression approach exploits the tendency of data-part to appear several times in a given data file. Therefore, a text file may contain several recurrences of a word, a phrase, or a syllable. In an image file, the same string of pixels may appear many times, and in a seismic trace, the same sequence of samples may appear several times. A dictionary based method creates a dictionary that contains indices and pieces of the data. As a string of data symbols is read from the input, the algorithm searches the dictionary for the longest match to the string. Once a match is found, the string is compressed by replacing it with a pointer to an entry in the dictionary.

Quite a few dictionary-based methods are known and the differences between them are in the way they organize and maintain the dictionary, in how they handle the strings not found in the dictionary, and in how they write their results (pointers, lengths, raw items, and perhaps flag bits) on the output.

In 1977, Jacob Ziv and Abraham Lempel proposed the LZ77 [50] that became the earliest notable methods in dictionary based compression. The LZ77 algorithms compress a data by replacing repeating occurrences sequence of samples with references to a single copy of that sequence existing earlier in the uncompressed data stream. Due to high variance, it is difficult to find repeating sequence of samples in seismic trace data. However, by grouping the samples based on their Gini coefficient, the dictionary based compression can be performed more efficient as reported in [19]. Data groups with low Gini coefficients are compressed using the Deflate algorithm which is a combination of the LZ77 and Huffman coding. The data groups with higher Gini coefficients are compressed with the Lempel-Ziv-Markov chain algorithm (LZMA). The LZMA uses a variance of LZ77 with large dictionary volumes and special feature for recurring used match distances. The algorithm find matches using efficient dictionary data structures, and generates a sequence of literal symbols and phrase pointers. The output is then encoded using a range encoder to model a probability prediction of each bit.

2.3.6 Prediction based compression

Statistical-model-based data predictions or prediction based compression (PBC) are promising ways of time series compression like seismic data. In PBC, the inherent temporal correlation samples in a trace is used to predict missing or future observations based on statistical model and existing observations. The statistical model and its prediction accuracy are the cores of the PBC [51] where the models are mainly based on auto-regression. Auto-regressive (AR) models are computationally simple and predict future observations as a weighted sum of previous samples . This technique is used in many lossless compression schemes [47, 34, 46, 44, 52] by storing the prediction errors

or residuals that come from the difference between predicted value and the actual value. In lossy schemes, the residual is stored whenever the error value exceeds a threshold [44]. Therefore, it reduces the residual data size and results in a higher compression ratio.

Despite its association with temporal correlation, PBC can be enhanced by exploiting both temporal and spatial correlation as reported by [39]. In a seismic sensor networks, spatial correlation is given by interdependency of data across sensors and shots. Theoretically, it is possible to exploit spatial correlation without direct communication across sensors. However, such schemes, namely distributed source coding (DSC), require a sophisticated usage of side information. Spatial correlation can also be retrieved from a single sensor given data from different shot locations in the past to enhance the prediction of the current trace. The enhancement starts with obtaining time shifts by examining the cross correlation between traces to find the delays. Second, the delays are used to align the traces. Finally, the prediction coefficients are constructed using all traces using RLS, called multi-trace-RLS. The residual is quantized, encoded, and then transmitted to prevent error accumulation during reconstruction.

2.3.7 Transformation based compression

To achieve higher compression ratios, data compression algorithms commonly have two major steps. The first step is decorrelation stage to exploit the redundancy in the data and the second is encoding which takes the advantage of the reduced entropy in the data as indicated by a lower entropy. Transformation-based compression approaches are very common for data decorrelation stage. The transformation techniques are designed to reduce data redundancy by identifying and removing the less important parts of the data and. Generally, transform-based approaches support lossy compressions by removing

irrelevant information in the transform domain where the raw data is transformed into a set of coefficients of basis functions. Typical transformation of signal $x[n]$ is given by

$$Y(k) = \sum_{n=1}^T x(n)w_T^{n,k}, k = 1 \dots T \quad (2.9)$$

$$x(n) = \sum_{k=1}^T y(k)\bar{w}_T^{n,k}, n = 1 \dots T, \quad (2.10)$$

where $y[k]$ denotes transform domain of $x[n]$, w is the transformation coefficient, and T denotes the signal length.

The Karhunen-Loeve transform (KLT) as known as the principal component analysis (PCA) known to be optimal transform with respect to variance, mean square error, and rate distortion function [53]. The basis functions for the PCA are the orthonormal eigenvector of the data autocorrelation matrix. The principal components can also obtained using neural networks learning as reported in [54]. Despite of its optimality, the PCA suffers from large overhead due to its data-dependent eigenvectors. Therefore, the PCA used to be employed as the benchmark for other seismic data compression techniques as reported in [35, 37, 55]. Liu et.al in [7] proposed an efficient way to reduce the communication burden on seismic sensor array for data compression. The statistics of the seismic traces acquired at all sensors are represented by a mixture model of a number of probability density functions. For sequential sensor arrays, the autocorrelation matrix is updated using weighted sum to reduce the communication burden [56]. Based on this mixture model, the distributed PCA constructs the global eigenvectors at the fusion center and distributes the eigenvectors to the sensors for compression.

Orthogonal transform is the most common transform used for data compression due to its efficiency for decorrelation. Its basis vector are orthogonal to each other such that the inner product is preserved. The Walsh-Hadamard transform (WHT) is one of the most well-known non-sinusoidal orthogonal transform where the basis consist of orthogonal pulse waveform, with amplitudes +1 and -1 [37]. This feature allows the transformation to be performed with addition and subtraction only. The WHT was the earliest transform technique used for seismic data compression [57]. The transform basis H_m for WHT is recursively given by

$$H_m = \frac{1}{\sqrt{2}} \begin{bmatrix} H_{m-1} & H_{m-1} \\ H_{m-1} & -H_{m-1} \end{bmatrix} \quad (2.11)$$

where m denotes the order and $H_0 = 1$. In transform domain, the WHT is able to compact 85% of the signal energy to a small portion frequency bandwidth. Therefore, the data compression can be achieved.

The N-point discrete Fourier transform (DFT) offers suboptimal transform where its compression capacity approaches the optimal transform for large N [37]. Its transform basis consist of complex number that allows spectral interpretation of seismic signal. The signal transformation to frequency domain can be performed efficiently using the fast Fourier transform (FFT). Spanias et.al [36, 35] employed magnitude-phase representation instead of real-imaginary DFT due to its suitability for transform coding implementation. The seismic signal is processed frame-by-frame using a sliding window. The magnitudes and phases are then encoded efficiently by assigning more bits for high energy frequency components than low energy ones. Since the magnitudes have high dynamic range, the quantization was performed to the log-magnitudes instead of the original values.

The discrete cosine transform (DCT) uses cosine functions oscillating at different discrete frequencies as its transform basis. Unlike DFT where the transform component is represented using complex number, the DCT transform component is real-valued hence can be represented using sign-magnitude format [37]. The performance of the DCT for seismic data compression is shown to be close with optimal transform [35, 58, 59].

Generalized linear-phase lapped orthogonal transform (GenLOT) [60, 61] was proposed based on the DCT. The GenLOT is implemented by segmenting the input signal into blocks with fixed size samples and transforming each block independently. The GenLOT uses lattice structure and weighted optimization that offers a better frequency partition and a higher coding gain that lead to more energy compaction for seismic data [62, 63]. The progressive compression scheme using GenLOT in [64] allows progressive quality control by transmitting the compressed file from the coarsest level down to the finest level. The GenLOT can be further optimized with respect to other measure like the DC leakage. The DC leakage quantifies the part of the DC energy that overlaps out of low-pass sub-band. It can be achieved by modelling the seismic signal using the symmetric noiseless auto-regressive model (SNAR) to design GenLOT filter optimized for seismic signal [65, 66]. In contrast with the GenLOT, where all the transform basis are of equal length, the generalized linear phase lapped orthogonal transforms with unequal length basis functions (GULLOT) [67] is proposed with different length of each transform basis. The GULLOT uses shorter basis for high frequency components that are suitable for reducing artifacts in seismic data. This feature is able to improve the quality of less smooth signal reconstruction in seismic shot gathers [68].

If a signal is oscillatory and has the same properties for the whole interval, then DFT and DCT become the most efficient methods. Despite of being oscillatory, seismic signal is also being transient or decaying such that the DFT and DCT require many components to damp the oscillation [69]. The discrete wavelet transform (DWT) is proposed to deal with the decaying oscillatory signal in seismic data. The basis functions for the DWT are all square integrable functions that support and are generated by scaling and translating. The basis construct an automatic windowing system where the window size is large for long signal trend vice versa. In term of computational cost, the DWT is more efficient with $O(n)$ operations compared with $O(n \log n)$ for the FFT. Due to these advantages, the DWT has been involved for seismic data compression techniques [41, 70, 71, 72]. The method is commonly used for lossy compression with a very low distortion as the degradation occurs as a slight smoothing effect [69, 38, 73].

Seismic data can be seen as three or four dimensions data that contain multidimensional redundancy. Therefore, two dimensions DCT and DWT cannot fully exploit the redundancy in the higher dimensions of data [29]. The multi-dimensional wavelet transform can be performed by repeated application of 1D wavelet filter-bank across all of the dimensions of the data [28, 70, 74]. A filter bank is band-pass filters that decompose the input signal into narrow spectral sub-bands. In practice, the filter-bank is applied recursively on the low frequency outputs by arranging the filter-banks according to sensor, time, and shot-dimension [13, 75]. This scheme allows real time transmission of data with high compression ratio [27, 76]. The emerging lifting scheme for multi-dimensional wavelet allows factorization of every wavelet filter into a finite sequence.

Seismic data compression can be achieved by shrinking the wavelet coefficients using a scale dependent non-linear soft thresh-holding [77, 78].

The implementation of multiple band filter bank for seismic signal is presented in [79, 32]. The multiple band filter bank is used to separate the seismic signal into multiple channel and is able to decompose seismic signal into localized time frequency coefficients [80, 33]. Two optimization models are proposed for multiple channel filter bank for common offset gathers (COGs) of seismic data in [30]. The first model uses separable and real-valued Gaussian-Markov processes fitted to the COGs correlation. This model is able to handle the statistics of COG to optimize the filter-bank with respect to coding gain. The second model use a memory-less infinite Gaussian mixture distribution fitted to the COG histograms after sub-band decomposition. This model allows optimization of the variable rate coding with respect to rate-distortion performance.

The wave packet transform is another transformation technique used for seismic data compression. [81, 82]. The wave packet transform expresses a signal as a weighted sum of wave packets. The wave packets are exponential decaying basis function that are suitable for seismic signal. The wavelet packet basis are waveforms defined by three independent parameters, namely position, scale, and frequency [83]. For seismic data with sampling frequency close to the Nyquist frequency, the wave packet transform outperforms the DFT and DWT without showing any artifact [84]. The extension of wave packet transform for two dimension can be found in [43] where the compression is performed by selecting appropriate frequency components. The most appropriate components are selected based on entropy criterion.

The embedded zero-tree wavelet (EZW) is an arithmetic coding scheme for wavelet coefficients that consists of three major steps. First step is by transmitting the larger coefficients, second step is refining the coefficients, and the last step is exploiting the spatial correlation between coefficients from different sub-band using the tree structures. The wavelet trees are created with DC components as the roots [62, 85]. These trees are then encoded using adaptive arithmetic coding. Another coding variant for wavelet is the set partitioning in hierarchical trees (SPIHT) that exploits the inherent similarities across the sub-bands [86, 34]. A combination of GenLOT with EZW framework for seismic data compression is shown to be superior in SNR comparing to the SPIHT [62]. For raw seismic data, the best combination is by using filter bank with long overlapping filters for time direction processing and short filter banks for distance direction [63, 87].



Figure 2.10 Bell Basis for local sine/cosine

Seismic signal is composed of oscillatory patterns with rapid variations of intensity that can only be synthesized with small scale wavelet coefficients. Therefore, the local cosine/sine transform (LCT) is proposed to achieve more efficient seismic data compressions [48]. The basis functions of LCT are orthogonal functions with good localization in time-frequency plane that consist of sine/cosine enveloped with bell functions [88] as illustrated in Figure 2.10. The bell function is bounded by 1 and 1.2, respectively, for one and two dimension to ensure the numerical stability for transformation and its inverse [89]. The semi-adaptive LCT is proposed in [90, 88] that allows flexibility in either time or space direction and uniform segmentation in the

counterpart. After migration, seismic data compressed with the multidimensional LCT is shown to produce a high quality image of subsurface structure [91, 38].

Transformation using signal adaptive transform or dictionaries offer more compact, sparse, and informative representation of the signal that is suitable for data compression application. The dictionary learning explores the similarities among local seismic traces [92, 93]. The dictionary and data representation can be constructed by solving optimization problem [94]. The dictionary learning for seismic data compression outperforms K singular value decomposition (KSVD) in term of reconstruction quality and learning times [39].

The full-3D seismic waveform tomography (F3DT) in seismic tomography combines broadband, multi-component seismic waveform observations into high-resolution 3D subsurface seismic structure. The disadvantage of the F3DT implementation is the high disk storage cost. Therefore, a new data format, called *zfp* [95], is proposed that enable high performance computing for data compression using transformation like DFT, WHT, or DCT.

2.3.8 Machine learning and other techniques

The other techniques that are not classified in the above categories may be dominated by machine learning approaches. Huang et.al., in [54, 96, 97], proved that neural networks can be used to obtain principal components for dimensionality reduction of seismic data. The neural networks with multiple hidden layers for dimensionality reduction of seismic data is presented in [98]. More comprehensive discussion is presented in chapter 5.

2.4 Metrics Used in Evaluating Data Compression Techniques

There are two major metrics used in seismic data compression to measure the performance of compression methods. The first metric is for measuring how much the data volume is reduced after compression. This metric is usually referred to as the compression ratio. The second metric is usually applied for lossy data compression to measure the level of distortion in the data after applying the compression. There are other metrics such as throughput and memory usage, to measure the compression/decompression speed and memory consumption during compression/decompression process, but these metrics are not commonly used in traditional applications.

2.4.1 Compression ratio

There are several definitions of compression ratios. The compression ratio is commonly defined as the ratio of the original data volume with respect to the compressed data volume. The definition can be mathematically expressed as follows

$$Cr = \frac{\text{Number of bits in the Original Data}}{\text{Number of bits in the Compressed Data}} \quad (2.12)$$

For example, a compression ratio of 10:1 means that the original volume is 10 times larger than the data volume after compression. This is the simplest representative of compression capability of data compression techniques. This metric is also a general purpose metric that can be used for any type of data compression approaches.

In [23, 24], the compression ratio is used to represent the percentage of data volume reduction. The compression ratio is defined as

$$Cr_{\%} = 100 \times \left(1 - \frac{\text{Compressed data volume}}{\text{Original data volume}}\right). \quad (2.13)$$

It can be noticed that this metric also represents the redundancy portion removed from the data. This metric also represents the complement of the inverse of compression ratio in eq. (2.12). A value of 60% means that the compression reduces 60% of its original volume.

Compression ratio is also represented as the bitrate or bit per sample (bps) as used in [32, 33, 13]. Bit rate is defined as

$$\text{BitRate} = \frac{\text{Number of Bits in the compressed data}}{\text{Number of Samples in the original data}}. \quad (2.14)$$

Bit rate is actually more suitable for data compressions that reduce the code-word size. This is typically found in compression based on variable length coding.

2.4.2 Reconstruction Quality for lossy compression

For lossy compression, the reconstruction quality represents the level of similarity of the reconstructed data with respect to the original data. However, most of the measures to represent the reconstruction quality are represented with the level of error or distortion introduced to the data by the compression scheme.

There are several measures for this purpose. The most common is based on the Mean Squared Error (MSE). The work in [32] used this metric to evaluate the reconstruction.

This metric can be expressed as

$$\text{MSE} = \frac{1}{MN} \sum_{i=1}^M \|\hat{\mathbf{x}}_i - \mathbf{x}_i\|^2, \quad (2.15)$$

where \mathbf{x} and $\hat{\mathbf{x}}$, respectively, denote the original and reconstructed trace. Number of traces in the dataset and the number of sample in a trace are denoted by M and N , respectively. The l2 norm operator $\|\mathbf{y}\|$ is given as

$$\|\mathbf{y}\| = \sqrt{y_1^2 + \dots + y_n^2}. \quad (2.16)$$

There are several direct derivations for MSE such as root mean squared error (RMSE), normalized root mean squared error (NRMSE), normalized error, signal to noise ratio (SNR), peak signal to noise ratio (PSNR), and absolute signal to noise ratio (ASNR).

As the name implies, the RMSE is the root square of the MSE. This metric can also be found in [32] and given as

$$\text{RMSE} = \frac{1}{MN} \sqrt{\sum_{i=1}^M \|\hat{\mathbf{x}}_i - \mathbf{x}_i\|^2}. \quad (2.17)$$

The major flaw of the direct MSE-based metric is that the metrics do not represent the actual distortion ratio with respect to the signal power. Therefore, several MSE based metrics are proposed by normalizing the distortion with respect to the magnitude of the original signal. Rubin et.al [24] represents the signal recovery errors by computing the NRMSE that can be calculated as

$$\text{NRMSE} = \frac{\sqrt{\text{mean}((\mathbf{x} - \hat{\mathbf{x}})^2)}}{\max(\mathbf{x}) - \min(\mathbf{x})}. \quad (2.18)$$

The other metrics are described in base 10 logarithmic scale. Another less popular metric is the normalized error e_{dBm} . This metric can be found in [35, 36, 37]. The metric is given as

$$e_{dBm} = 10 \log_{10} \left\{ \frac{\sum_{i=1}^M N \max(|\hat{\mathbf{x}}_i - \mathbf{x}_i|^2)}{\sum_{i=1}^M \|\mathbf{x}_i\|^2} \right\}, \quad (2.19)$$

where $|\bullet|$ denotes the absolute value operator. Another variant is the ASNR that uses the ratio of the absolute value of the data and the error as given as

$$\text{ASNR} = 10 \log_{10} \frac{\sum_{i=1}^M |\mathbf{x}_i|}{\sum_{i=1}^M |\mathbf{x}_i - \hat{\mathbf{x}}_i|}. \quad (2.20)$$

This metric can be found in [41]. PSNR as found in [48] is defined as

$$\text{PSNR} = 10 \log_{10} \frac{x_{max}^2}{\frac{1}{MN} \sum_{i=1}^M \|\mathbf{x}_i - \hat{\mathbf{x}}_i\|^2}, \quad (2.21)$$

where x_{max} denotes the maximum value of all data.

Finally, the most common metric, the SNR is a popular metric in signal processing communities to measure the signal distortion. The SNR can be computed as

$$\text{SNR} = 10 \log_{10} \frac{\sum_{i=1}^M \|\mathbf{x}_i\|^2}{\sum_{i=1}^M \|\mathbf{x}_i - \hat{\mathbf{x}}_i\|^2}. \quad (2.22)$$

The SNR is more representative than the PSNR since the metric uses the variance of the original signal instead of a single value like PSNR. This metric is sometimes also called as the signal to quantization noise ratio (SQNR) when the data reconstruction involves the quantization procedure as found in [30]. One can notice that MSE-based loss metrics

have been used in many works. It has been brought to attention for some researchers to assess the metrics based on the effects to the seismic images [27, 70, 99].

2.5 Conclusion and remarks

In this chapter, we presented a brief discussion of different types of seismic data compression methods. We classified large number of compression techniques proposed for seismic data to several classes. It can be noticed that the transformation approaches, especially the wave-based, dominate the compression methods proposed for seismic data. This is intuitive since a seismic signal is a wave that can be exploited for compression if we can model the signal. For lossy compression, the MSE-based metric is the most used metric. The final product of seismic data processing is a subsurface image. Therefore, a method that is oriented to the subsurface image quality may lead to a more efficient compression. In the next chapter, we present transform based compression using DPCA.

CHAPTER 3

DISTRIBUTED PRINCIPAL COMPONENT

ANALYSIS

This chapter discusses a framework based on the Principal Component Analysis (PCA) for data compression over seismic sensor networks, which is called the distributed PCA (DPCA). The algorithm of the DPCA is summarized as follows. First, the statistics of the seismic traces collected by all sensors in the network are expressed as a mixture probabilistic model, which is represented by a weighted sum of local probability density functions (pdfs). Each of the local pdfs matches the distribution of the traces from a single sensor in terms of its first two order statistics. Second, a distributed implementation of the PCA compression scheme for all sensors in the network is achieved using the model above. The implementation of the DPCA requires each sensor to send the mean vector and the covariance matrix of its traces to the fusion center through the network. The fusion center then determines a set of global PCs and broadcasts them back to the sensors for projection.

This chapter is organized as follows: A brief discussion of PCA as linear transform is presented in section 3.1. Derivation of the DPCA as mixture model and its implementation for practical seismic sensor networks is discussed in section 3.2. DPCA implementation on seismic sensor array and conclusions are presented in section 3.3 and section 3.4, respectively.

3.1 Principal Component Analysis

The PCA or well-known as Karhunen-Loeve Transform (KLT) has a linear optimal energy compaction for a set of given data. Instead of single signal, PCA is applied to a set of signals. Transformation and inverse relationship for single signal in (2.9) and (2.10) are still applicable but the signal must be set to zero average signal \mathbf{x}_0 by removing the mean \bar{x} from all sample

$$x_0(n) = x(n) - \bar{x}(n), \quad (3.1)$$

Transformation and its inverse as in (2.9) and (2.10) can be applied to $x_0(n)$ using transform basis \mathbf{W}_{PCA} . The transform basis \mathbf{W}_{PCA} can be obtained from eigenvector \mathbf{V}_X of the covariance matrix of data $\mathbf{X} \in \mathbb{R}^{T \times M}$ where M denotes number of data vector, which is easily calculated from the data by using Singular Value Decomposition (SVD) as follows.

$$\mathbf{\Sigma}_X = cov(\mathbf{X}) \quad (3.2)$$

$$\mathbf{\Sigma}_X = \mathbf{V}_X \mathbf{\Lambda}_X \mathbf{V}_X^T \quad (3.3)$$

$$\mathbf{W}_{PCA} = \mathbf{V}_X \quad (3.4)$$

$$\mathbf{W}_{PCA}^{-1} = \mathbf{V}_X^{-1} = \mathbf{V}_X^T \quad (3.5)$$

Each eigenvector \mathbf{p}_X in \mathbf{P}_X corresponds to an eigenvalue λ_X in $\mathbf{\Lambda}_X$. This compression methodology is similar with the DCT by selecting basis with high energy. In this case the energy is represented by the eigenvalues. Assuming that $\mathbf{\Lambda}_X$ is sorted in descending order,

compression can be done by selecting first L eigenvectors according to its corresponding eigenvalues. The reconstruction quality is evaluated by its Normalized Cumulative Energy (NCE) that is defined as follows:

$$NCE = \frac{\sum_{k=1}^L \lambda_k}{\sum_{k=1}^N \lambda_k} \quad (3.6)$$

The reconstructed signal can be obtained from the following relation

$$\hat{x}_0(n) = \sum_{k=1}^L y(k) \bar{v}_T^{n,k}, n = 1 \dots T, \quad (3.7)$$

where \bar{v} denotes the transpose of the selected transformation basis and $\hat{x}_0[n]$ is the reconstructed signal without zero mean. The mean vector \bar{x} is added to obtain the final reconstructed signal \hat{x} as expressed below.

$$\hat{X}_{PCA} = \hat{X}_0 + \bar{X} \quad (3.8)$$

Since the transformation basis $V \in \mathbb{R}^{T \times M}$ are different for each data set, the selected eigenvectors $\hat{V} \in \mathbb{R}^{N \times L}$ must be transmitted along with the transformed data $Y \in \mathbb{R}^{M \times L}$.

The compression ratio is given by:

$$C_{r_{PCA}} = \frac{TM}{ML + TL + T}. \quad (3.9)$$

3.2 Mixture Model based DPCA

In the proposed framework, the basic assumption is that sensors in the network have storage, computation and communication capabilities. The main target of our design is

the use a set of global PCs among all the sensors for compression instead of using local PCs. Furthermore, during the procedure of obtaining the global PCs, only the statistics of the data collected by the sensors are transmitted to the fusion center, which leads to a reduced communication load. In this section, the concept of the mixture model is introduced first and the seismic traces collected by multiple sensors are modeled as the realizations of the mixture model.

Assume that the distribution of a T -dimensional stochastic vector given by:

$$\mathbf{x} = [x_1, x_2, \dots, x_T]^T. \quad (3.10)$$

is a convex sum of a number of sub-population distributions, then the probability of \mathbf{x} can be described as

$$f(\mathbf{x}) = \sum_{i=1}^M \omega_i f_i(\mathbf{x}). \quad (3.11)$$

where M denotes the number of mixtures and $f_i(\mathbf{x})$ are the local pdfs of \mathbf{x} . The weights ω_i , for $i = 1, 2, \dots, M$ are non-negative and their sum is one. By defining the parameter set for each component as θ_i , $i = 1, 2, \dots, M$, we have $P(\theta_i | \mathbf{x} = X) = \omega_i$ for any realization vector \mathbf{X} . It is assumed that the first two statistical moments, i.e., the mathematical expectation $\boldsymbol{\mu}_i$ and the covariance $\boldsymbol{\Sigma}_i$, of the local pdfs exist, and the higher order moments are ignored, then the parameter sets of the local pdfs are represented as

$$\theta_i = \{\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\}, \quad (3.12)$$

thus the first two moments of mixture model (3.11) are given as:

$$\boldsymbol{\mu} = \sum_{i=1}^M \omega_i \boldsymbol{\mu}_i \quad (3.13)$$

$$\boldsymbol{\Sigma} = \sum_{i=1}^M \omega_i \boldsymbol{\Sigma}_i + \sum_{i=1}^M \omega_i (\boldsymbol{\mu}_i - \boldsymbol{\mu})(\boldsymbol{\mu}_i - \boldsymbol{\mu})^T. \quad (3.14)$$

A seismic trace is a time series with a length, say T , recorded by a sensor from one shot.

Let us define it as:

$$\mathbf{x}_j^{(i)} = [x_j^{(i)}(1), x_j^{(i)}(2), \dots, x_j^{(i)}(T)]^T, \quad (3.15)$$

where i and j denote the indices of sensors and shots, respectively. Scalar $x_j^{(i)}(k)$ denotes the k -th sample of the trace from the sensor i and the shot j . Trace $\mathbf{x}_j^{(i)}$ is regarded as a realization of a T -dimensional stochastic variable \mathbf{x} , which is assumed to follow an unknown local pdf $f_i(\mathbf{x})$. Once the i -th sensor collects a number of traces, say N_i traces $\{\mathbf{x}_j^{(i)}\}_{j=1}^{N_i}$, the first two moments of pdf $f_i(\mathbf{x})$ are estimated as follows:

$$\boldsymbol{\mu}_i = \frac{1}{N_i} \sum_{j=1}^{N_i} \mathbf{x}_j^{(i)} \quad (3.16)$$

$$\boldsymbol{\Sigma}_i = \frac{1}{N_i - 1} \sum_{j=1}^{N_i} (\mathbf{x}_j^{(i)} - \boldsymbol{\mu}_i)(\mathbf{x}_j^{(i)} - \boldsymbol{\mu}_i)^T. \quad (3.17)$$

Suppose that there are M sensors in the network, applying the same modeling formulation mentioned above to other sensors yields M local pdfs, i.e., $f_i(\mathbf{x})$, for $i = 1, 2, \dots, M$. To capture the global statistics of the seismic traces collected by the sensor network, we

construct the global pdf of the traces, \mathbf{x} , as a convex sum of the local pdfs $f_i(\mathbf{x})$ as in (3.11) with weights

$$\omega_i = \frac{N_i}{N}, \quad (3.18)$$

where $N = \sum_{i=1}^M N_i$ and its first two moments, $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, are obtained as in (3.13)-(3.14).

Based on this modeling, the DPCA scheme is proposed as follows. First, decomposing $\boldsymbol{\Sigma}$ as

$$\boldsymbol{\Sigma} = \mathbf{P}\boldsymbol{\Lambda}\mathbf{P}^T, \quad (3.19)$$

where $\boldsymbol{\Lambda} \in \mathcal{R}^{T \times T} = \text{diag}[\lambda_1, \lambda_2, \dots, \lambda_T]$ and $\mathbf{P} \in \mathcal{R}^{T \times T} = [\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_T]$. The scalars $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_T$ and the vectors $\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_T$ are the eigenvalues and their corresponding eigenvectors of $\boldsymbol{\Sigma}$, respectively. Let

$$\tilde{\mathbf{P}}_k = [\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_k], k \leq T, \quad (3.20)$$

be the first k columns of \mathbf{P} . Once the sensors have the global PCs $\tilde{\mathbf{P}}_k$, they project the original traces on the PCs to obtain the projection coefficients $\{\mathbf{y}_j^{(i)}\}_{j=1}^{N_i}$ as follows,

$$\mathbf{y}_j^{(i)} = \tilde{\mathbf{P}}_k^T (\mathbf{x}_j^{(i)} - \boldsymbol{\mu}), j = 1, 2, \dots, N_i. \quad (3.21)$$

This scheme indicates that, for calculating a set of global PCs from the traces collected by multiple sensors, it only requires the first two moments of the local pdfs associated with their weights instead of accessing all traces. This can be done in the following steps:

1. Each sensor calculates its own first two moments as in (3.16)-(3.17) and sends them along with N_i , the number of traces, to the fusion center through the network.
2. The fusion center calculates the global PCs and broadcasts them along with the global mean $\boldsymbol{\mu}$ to all sensors;
3. Each sensor projects its traces on the global PCs and sends the project coefficients to the fusion center.

3.3 Application of the DPCA on the seismic sensor network

In this section, we decompose the practical seismic sensor network into two fundamental topologies: star and cascade connections. Then we apply the developed mixture model-based DPCA scheme to these topologies. Based on the results, we propose the application solution of the DPCA for the practical seismic sensor network.

3.3.1 Applying the DPCA on two fundamental topologies

One of the fundamental topologies considered in this section is shown in Figure 3.1 (a), which is a typical star connection. A number of sensors, S_1, S_2, \dots, S_M denoted by circles in the schematic figure, are connected to a fusion center G denoted by a square. Note that there are no connections between the sensors. After all sensors collect the traces, they calculate the first two moments of their own traces and send them along with the number of traces to the fusion center. The fusion center calculates the global PCs from the moments it receives from all sensors, and sends them back to each sensor. Then, the sensors project their traces on the global PCs and send the coordinates to the fusion

center. Last, the fusion center can reconstruct the original trace data from the received coordinates and the global PCs. This indicates that the proposed scheme can be directly applied on star connection topologies.

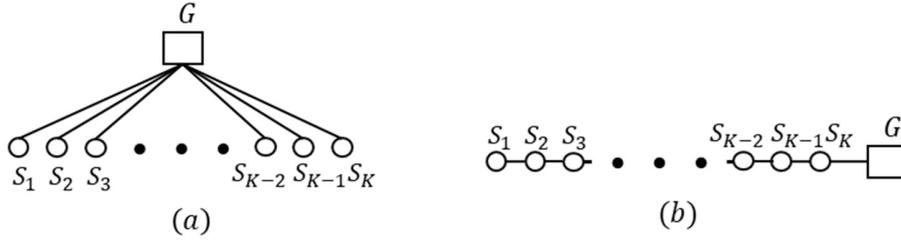


Figure 3.1: Sensor layout model (a) star (b) sequential

The second fundamental topology of the seismic sensor network is shown in Figure 3.1 (b). In this topology, the sensors are connected in a cascade way. Each of those sensors can only communicate with its nearest neighbors, and the sensor located at the end of this linkage is connected to the fusion center.

The most direct way is to require each sensor to send its statistics and whatever it receives from its left-hand-side neighbor to the sensor at its right-hand-side. For example, sensor S_i receives $\{\theta_j, N_j\}_{j=1}^{i-1}$ from sensor S_{i-1} and sends them along with θ_i and N_i to sensor S_{i+1} . However, since there are no direct connection between the sensors and the fusion center, except for the last sensor, all information needed at the fusion center must be transferred through multiple hops. The closer a sensor is to the fusion center, the more preceding sensors it has, which implies that it has to undertake heavier communication burden. In other words, the data package in the communication among the sensors increases as the index of the sensors increases during this procedure. To alleviate this

drawback, we develop a peculiar scheme for this *sequential connection* topology. First, substitute (3.18) in (3.13)-(3.14) and rewrite them as follows,

$$\boldsymbol{\mu} = \frac{1}{N} \sum_{i=1}^M N_i \boldsymbol{\mu}_i \quad (3.22)$$

$$\boldsymbol{\Sigma} = \frac{1}{N} \sum_{i=1}^M N_i \boldsymbol{\Sigma}_i + \frac{1}{N} \sum_{i=1}^M N_i \boldsymbol{\mu}_i \boldsymbol{\mu}_i^T + \boldsymbol{\mu} \boldsymbol{\mu}^T - \boldsymbol{\mu} \left(\frac{1}{N} \sum_{i=1}^M N_i \boldsymbol{\mu}_i \right)^T - \left(\frac{1}{N} \sum_{i=1}^M N_i \boldsymbol{\mu}_i \right) \boldsymbol{\mu}^T \quad (3.23)$$

In order to obtain $\boldsymbol{\mu}$ at the fusion center, one can easily see from (3.22) that it is not necessary to separately transmit all $\boldsymbol{\mu}_i$'s and N_i 's from the sensors to the fusion center. Take sensor S_i as an example, what we need is only to transmit the cumulative terms $\sum_{j=1}^i N_j \boldsymbol{\mu}_j$ and $\sum_{j=1}^i N_j$ to the next sensor. The next sensor S_{i+1} , updates the received cumulative terms by adding $N_{i+1} \boldsymbol{\mu}_{i+1}$ and N_{i+1} to them, respectively. Then the new cumulative terms become $\sum_{j=1}^{i+1} N_j \boldsymbol{\mu}_j$ and $\sum_{j=1}^{i+1} N_j$. It then sends these updates to sensor S_{i+2} . This procedure is repeated until the fusion center receives two cumulative terms $\sum_{j=1}^K N_j \boldsymbol{\mu}_j$ and $\sum_{j=1}^K N_j$ from the last sensor. The global mean $\boldsymbol{\mu}$ is easily obtained from those two cumulative terms as in (3.22).

Similarly, in order to obtain $\boldsymbol{\Sigma}$ at the fusion center, taking sensor S_i as an example, what the sensor needs is receiving cumulative terms $\sum_{j=1}^i N_j \boldsymbol{\Sigma}_j$ and $\sum_{j=1}^i N_j \boldsymbol{\mu}_j \boldsymbol{\mu}_j^T$ from the previous sensor, updating them, then sending the updates to the next sensor. Note that based on this procedure, the overall communication burden does not increase throughout the sequential line of sensors. This scheme is summarized in Algorithm 3.1.

```

Begin
Starting from sensor  $S_1$ ;
Sending  $\{N_1, N_1\boldsymbol{\mu}_1, N_1\boldsymbol{\Sigma}_1, N_1\boldsymbol{\mu}_1\boldsymbol{\mu}_1^T\}$  to  $S_2$ 
For Sensor  $S_i, 2 \leq i \leq M$  do
Receiving from sensor  $S_{i-1}$   $\{\sum_{j=1}^{i-1} N_j, \sum_{j=1}^{i-1} N_j \boldsymbol{\mu}_j, \sum_{j=1}^{i-1} N_j \boldsymbol{\Sigma}_j, \sum_{j=1}^{i-1} N_j \boldsymbol{\mu}_j \boldsymbol{\mu}_j^T\}$ 
Updating cumulative terms as:

$$\sum_{j=1}^i N_j = \sum_{j=1}^{i-1} N_j + N_i$$


$$\sum_{j=1}^i N_j \boldsymbol{\mu}_j = \sum_{j=1}^{i-1} N_j \boldsymbol{\mu}_j + N_i \boldsymbol{\mu}_i$$


$$\sum_{j=1}^i N_j \boldsymbol{\Sigma}_j = \sum_{j=1}^{i-1} N_j \boldsymbol{\Sigma}_j + N_i \boldsymbol{\Sigma}_i$$


$$\sum_{j=1}^i N_j \boldsymbol{\mu}_j \boldsymbol{\mu}_j^T = \sum_{j=1}^{i-1} N_j \boldsymbol{\mu}_j \boldsymbol{\mu}_j^T + N_i \boldsymbol{\mu}_i \boldsymbol{\mu}_i^T$$


```

Algorithm 3.1: Sequential connection case

3.3.2 Practical seismic sensor network

In a practical geophysical exploration, the area to be surveyed is covered by a number of lines of sensors, see Figure 3.2 for example. It is assumed that there are M lines of sensors, in each, say the r th line, there are M_r sensors connected in a *sequential connection* fashion. The last sensor of each line is connected to a local fusion center. Then all local fusion centers $\{G_r\}_{r=1}^M$ are connected to the global fusion center G in a *star connection* fashion.

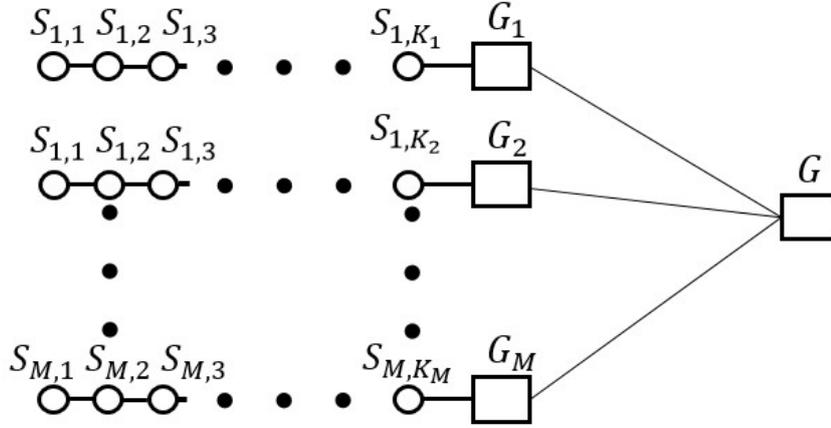


Figure 3.2: Practical seismic sensor networks.

Suppose that sensor $S_{r,c}$ collects $N_{r,c}$ traces, which follow a pdf $f_{r,c}(\mathbf{x})$. As analyzed above, the global pdf of all traces is modeled as:

$$f(\mathbf{x}) = \sum_{r=1}^M \sum_{c=1}^{M_r} \omega_{r,c} f_{r,c}(\mathbf{x}), \quad (3.24)$$

where

$$\omega_{r,c} = \frac{N_{r,c}}{\sum_{r=1}^M \sum_{c=1}^{M_r} N_{r,c}}. \quad (3.25)$$

Let $\mu_{r,c}$ and $\Sigma_{r,c}$ be the mean vector and covariance matrix of pdf $f_{r,c}(\mathbf{x})$, respectively. To easily obtain the global covariance matrix of $f(\mathbf{x})$ at the fusion center G by applying the scheme we introduced in the previous sections, the sensor network is decomposed as a sum of M sub-networks, each of which contains a line of sensors in sequence. Define the weights of the sub-networks as $\omega_r = \sum_{c=1}^{M_r} \omega_{r,c}$, $r=1, \dots, M$. Therefore, the traces

collected by each of the sub-networks are assumed to follow $f_r(\mathbf{x}) = \sum_{c=1}^{M_r} \frac{\omega_{r,c}}{\omega_r} f_{r,c}(\mathbf{x})$

with parameters as

$$\boldsymbol{\mu}_r = \sum_{c=1}^{M_r} \frac{\omega_{r,c}}{\omega_r} \boldsymbol{\mu}_{r,c} \quad (3.26)$$

$$\boldsymbol{\Sigma}_r = \sum_{c=1}^{M_r} \frac{\omega_{r,c}}{\omega_r} \boldsymbol{\Sigma}_{r,c} + \sum_{c=1}^{M_r} \frac{\omega_{r,c}}{\omega_r} (\boldsymbol{\mu}_{r,c} - \boldsymbol{\mu}_r)(\boldsymbol{\mu}_{r,c} - \boldsymbol{\mu}_r)^T. \quad (3.27)$$

For each line of sensors, say the r th line, Algorithm 3.1 can be applied to the sensors $\{\mathcal{S}_{r,c}\}_{c=1}^{M_r}$ to pass their statistics to the local fusion center G_r . At the end, G_r has the statistics of this sub-network, i.e., $\boldsymbol{\mu}_r$, $\boldsymbol{\Sigma}_r$ and $N_r = \sum_{c=1}^{M_r} N_{r,c}$. Global pdf $f(\mathbf{x})$ can be rewritten as

$$f(\mathbf{x}) = \sum_{r=1}^M \omega_r f_r(\mathbf{x}), \quad (3.28)$$

The mean vector and covariance matrix of the global pdf $f(\mathbf{x})$ can be rewritten as

$$\boldsymbol{\mu} = \sum_{r=1}^M \omega_r \boldsymbol{\mu}_r \quad (3.29)$$

$$\boldsymbol{\Sigma} = \sum_{r=1}^M \omega_r \boldsymbol{\Sigma}_r + \sum_{r=1}^M \omega_r (\boldsymbol{\mu}_r - \boldsymbol{\mu})(\boldsymbol{\mu}_r - \boldsymbol{\mu})^T. \quad (3.30)$$

To obtain the global covariance matrix $\boldsymbol{\Sigma}$ at G , one may just treat the connection of the local fusion centers $\{G_r\}_{r=1}^M$ and global fusion center G as in the *star connection*, then apply the communication scheme for the sequential topology to the local fusion centers.

After calculating the global PCs $\tilde{\mathbf{P}}_k$, the fusion center G disseminates them to all sensors for compression.

3.3.3 Compression ratio analysis

The compression ratio is defined as

$$r = \frac{\text{volume of original data}}{\text{volume of compressed data}}. \quad (3.31)$$

The volume of the original seismic traces is TN . As for the LPCA compression using k PCs, the compressed data stored in the fusion center consists of three parts: coefficients $\{\mathbf{y}_j^{(i)}\}_{i,j=1}^{M,N_i}$ with volume kN , k PCs for each sensor with total volume kMT and the local means with volume MT . Thus, the compression ratio of the LPCA using k PCs on average is calculated as

$$r_L(k) = \frac{TN}{kN + kMT + MT}. \quad (3.32)$$

Similarly, the compression ratio of the DPCA using k PCs is calculated as

$$r_D(k) = \frac{TN}{kN + kT + T}. \quad (3.33)$$

Note that the only difference between $r_L(k)$ and $r_D(k)$ comes from the fact that the fusion center only stores k global PCs and the global mean in DPCA, instead of k PCs and one local mean for each of the sensors in LPCA.

3.3.4 Experimental results

The *East Texas USA* Database [100] consists of sequentially arranged 33 sensors with an interval of 220 feet. There are 18 traces in each sensor originated by 18 shots produced by dynamites at a depth of 80 – 100 feet underground. The length of each trace is 1501 time samples, which is approximately 3 seconds. Since the magnitude of the acoustic wave is attenuated as it propagates to the sensors over distance, different sensors receive traces from the same shot with different magnitude. To mitigate this nuisance, we normalize the magnitude of the trace to be between -1 to 1 , as shown in Figure 3.3.

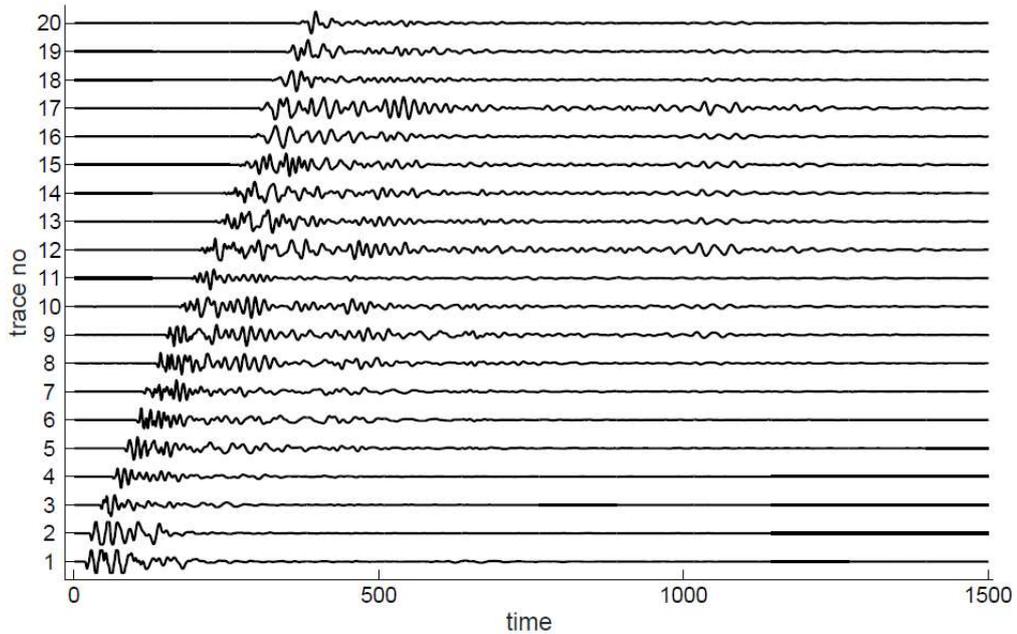


Figure 3.3: The normalized traces of (East Texas USA Database).

We take the NCE as a metric to evaluate the preserved energy after compression. First, we use only data from first 10 sensors to compute the PCs. Figure 3.4 shows the NCE of 10 sets of local PCs, each of them is calculated from a local covariance matrix, in solid

lines, and the NCE of the global PCs in dash line. One can easily notice some phenomena. First, for both DPCA and LPCA schemes, the NCE increases as the number of PCs increases, and later it levels off when the number of PCs is large enough. This can be due to the fact that the more PCs are used, the more energy is preserved. Since the PCs, or the eigenvalues, are arranged in a decreasing order, adding more small PCs does not improve the NCE significantly. Second, the NCEs of the LPCA increase drastically at the very beginning, which means that the traces collected by the same sensor share high similarities, therefore they could be highly compressed using a small number of local PCs. Last, the NECs of the LPCA are obviously higher than that of the DPCA by using the same number of PCs, or in other words, to preserve the same amount of energy during the compression, we need more global PCs than local PCs. This can be explained by the fact that the sensors are located away from each other, therefore traces collected by all sensors share relatively lower similarities than that from the same sensor. However, the DPCA compression still has more overall benefits than the LPCA. First, it requires one matrix decomposition in the fusion center, instead of 10 matrix decompositions at the sensors as in the LPCA compression. Moreover, although the DPCA needs more PCs than the LPCA, it leads to a higher compression ratio than the LPCA. For example, to preserve approximately 99% energy, we need around 12 PCs for the LPCA while 34 PCs are needed for the DPCA, resulting in compression ratios of $r_D = 4.60$ for the DPCA and of $r_L = 1.37$ for the LPCA. The compression ratio of the DPCA is around 3.36 times as that of the LPCA.

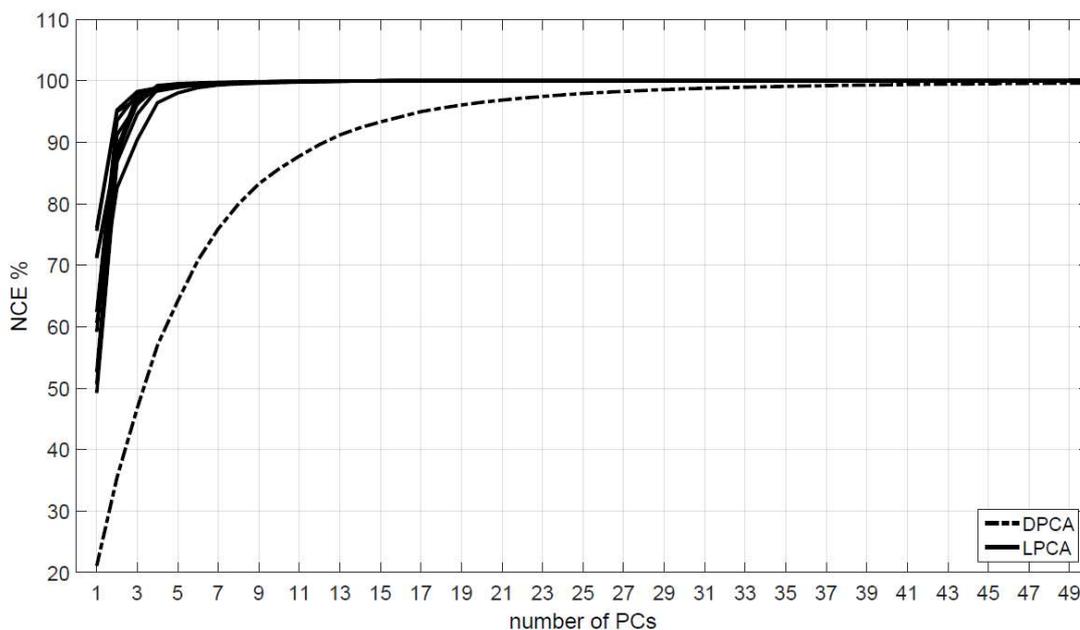


Figure 3.4: The NCEs of the LPCA and the DPCA by using 10 sensors and 18 shots

Figure 3.5 shows the NCEs of the LPCA and the DPCA by using data from all sensors and shots. It can be noticed that the NCE of the DPCA is lower than that of that of using only 10 sensors. Figure 3.6 plots the NCE of all transform based on the number of component required. As comparison for the DPCA and DCT, Walsh-Hadamard Transform (WHT) NCE is added to the plot. Since the PCA represents optimal transform for local sensor, DPCA provides global optimal transform components for all sensors. This optimality is proved by its superiority in term of number of components required to satisfy particular level of NCE. In the figure, to obtain more than 99% of NCE, 78, 392, and 1054 components are required for DPCA, DCT, and WHT, respectively. These numbers correspond to 5:1 for DPCA and 2:1 for DCT of compression ratio. WHT has the same expression as DCT for compression ratio, however, the number of components required is much larger than that of the DCT. Large number of components makes the WHT inefficient with 0.7:1 of compression ratio.

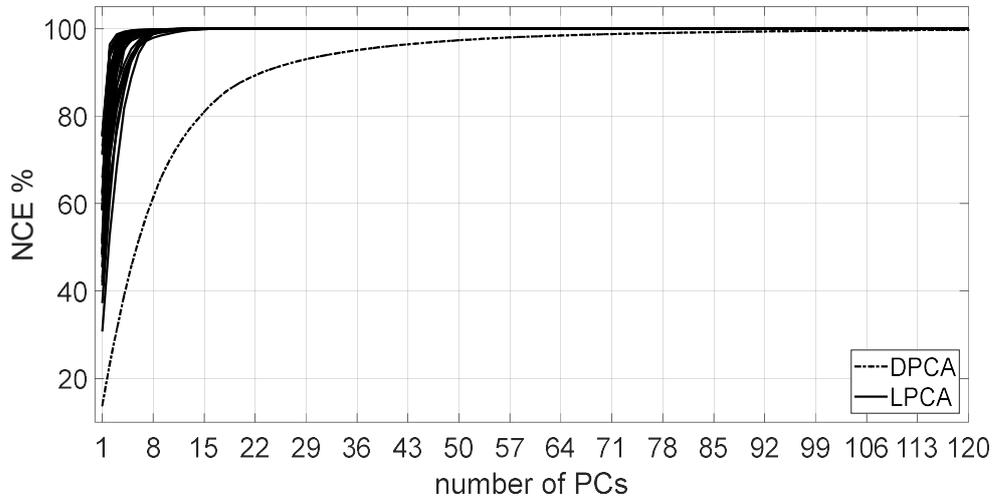


Figure 3.5: The NCEs of the LPCA and the DPCA by using 33 sensors and 18 shots

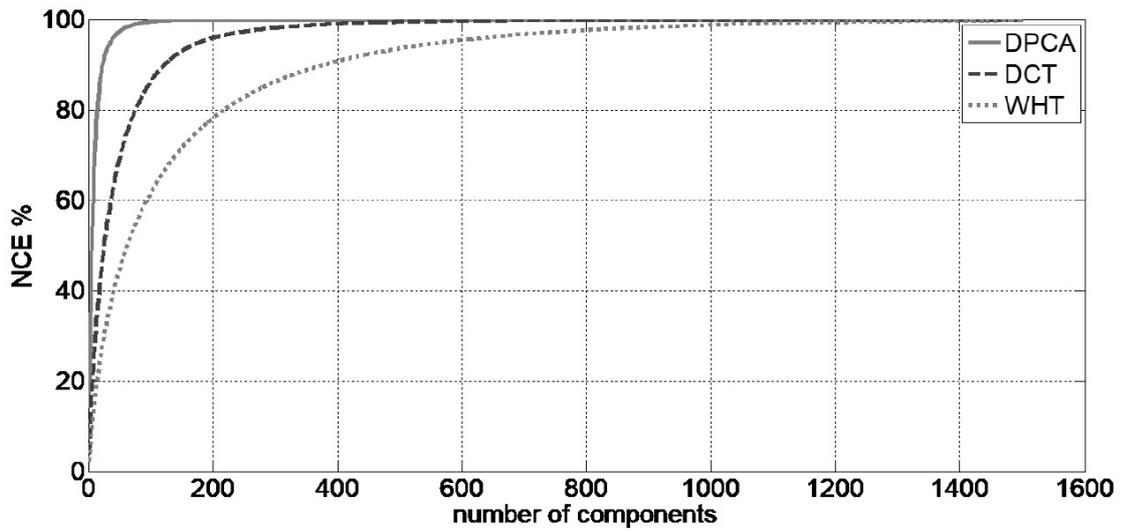


Figure 3.6: Cumulative Energy Comparison of (East Texas USA Database).

The DPCA actually requires only 78 out of 1501 components for each trace to reach 99% of NCE. However, the DPCA uses data dependent components, which results in different

components for different dataset thus the overhead in the form of the components must be transmitted as well. If the number of traces is much larger than the number time samples, and each trace contains high correlation with other traces, compression becomes more efficient.

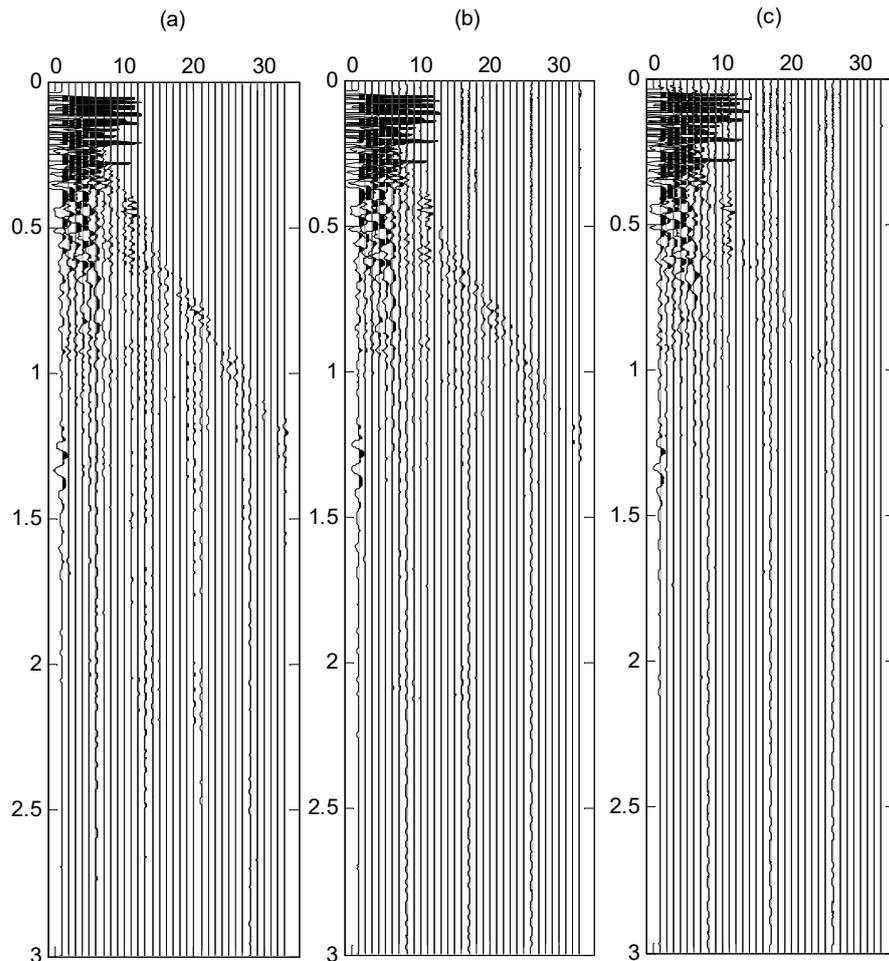


Figure 3.7: Trace reconstruction (a) original (b) 99.9% of NCE (c) 99% of NCE.

The common shot gathers (CSG) reconstruction sample using 99% and 99.9% of NCE can be seen in Figure 3.7. The early stages of traces are with high amplitudes, the

reconstruction of this stage keeps a high quality, however, in the later stages with low amplitudes, the traces are diminished in the 99% NCE case. Reconstruction with 99.9% NCE maintains lower amplitude reflections with no significant loss from original traces.

3.4 Concluding remarks

In this chapter, we presented a distributed Principal Component Analysis (DPCA) compression scheme for seismic sensor networks. The seismic traces collected by multiple sensors are modeled as realizations of a mixture of probability density functions (pdfs), whose mean vector and covariance matrix can be estimated at a fusion center in a distributed fashion without accessing the individual traces. A set of global Principal Components (PCs) is obtained by decomposing the global covariance matrix. These PCs are then broadcasted to all sensors for data projection. There are two advantages of the proposed scheme compared with the local Principal Component Analysis (LPCA) scheme. First, the proposed algorithm exhibits a lower computation cost in terms of matrix decomposition. Secondly, it achieves a higher compression ratio. Furthermore, an efficient method is proposed to alleviate the communication burden of the sequential sensor array, in which the data package transmitted among the sensors does not increase in size. Finally, a number of experiments are implemented to show the efficiency of the proposed scheme. Specifically, to preserve 99% of signal energy during the compression, the DPCA achieves at least three times compression ratio as the LPCA in both real and synthetic data. Moreover, the proposed scheme has a lower reconstruction error than the DCT and WHT compression techniques.

CHAPTER 4

MODEL BASED SEISMIC DATA COMPRESSION

In this chapter, we propose to use exponentially decaying sinusoidal waves to model seismic traces. A trace is described as a superposition of exponentially decaying sinusoidal waves. Each wave is regarded as a model component, and has a distinct starting time, decaying factor, frequency, amplitude, and phase shift. A parameter estimation algorithm for this model is developed using particle swarm optimization (PSO) technique and is extended for multiple model estimation. A suitable number of model components is selected according to the residual energy. The residuals are then quantized and encoded using fixed-length encoding to improve the reconstruction quality. The proposed model-based compression scheme is experimentally compared with the DCT using a real data set.

This chapter is organized as follows: Seismic trace modelling and its parameter estimation is discussed in section 4.1. Experimental results is presented in section 4.2. Section 4.3 presents the concluding remarks.

4.1 Seismic Trace Modelling

A seismic trace, $x[k]$, with N time samples is modeled as a sum of a number of exponentially decaying sinusoidal waves (EDSWs) as

$$x(k) = \sum_{(i=1)}^M s_{\theta_i}(k) + v(k) \quad k = 1, 2, \dots, N, \quad (4.1)$$

where $v[k]$ denotes unmodeled signal and the EDSWs, $s_{\theta_i}(k)$ are defined as

$$s_{\theta_i}(k) = \begin{cases} e^{-h_i(kT-T_i)} \alpha_i \sin(\omega(kT - T_i) + \phi_i) & kT \geq T_i, \\ 0 & kT < T_i, \end{cases} \quad (4.2)$$

where T is the sampling period of the seismic trace and h_i , T_i , α_i , ω_i , and ϕ_i are decaying factor, starting time, initial amplitude, frequency, and phase shift, respectively. Define parameter set for each sinusoidal wave as $\theta_i = \{T_i, h_i, \alpha_i, \omega_i, \phi_i\}$, then the parameters of all sinusoidal waves are defined as $\Theta = \{\theta_1, \theta_2, \dots, \theta_M\}$.

The goal of this chapter is to design a compression scheme by representing a seismic trace with a finite number of parameters from a suitable number of model components that leads to a high compression ratio. To achieve that goal we develop an efficient scheme to estimate the parameters Θ of model (4.1) from a seismic trace. The model is then enhanced by encoding the final residual to achieve higher reconstruction quality.

4.1.1 Single EDSW estimation

In this section, we consider parameter estimation for a single EDSW model case, i.e. model (4.2) with $M = 1$, $\theta = \{k_s, h, \alpha, \omega, \phi\}$:

$$x(k) = \begin{cases} e^{-h_s(kT-T_s)} \alpha_s \sin(\omega(kT - T_s) + \phi_s) & kT \geq T_s, \\ 0 & kT < T_s \end{cases} \quad (4.3)$$

We can estimate the parameter set using non-linear optimization techniques by minimizing the discrepancy between the model and the actual seismic trace. The problem of parameter estimation can be formulated as the following optimization problem.

$$\hat{\theta} = \arg_{\theta} \min e(\mathbf{x}, \mathbf{x}(\theta)), \quad (4.4)$$

where $e(\mathbf{x}, \mathbf{x}(\theta))$ the cost function corresponding to a reconstruction error is obtained as

$$e(\mathbf{x}, \mathbf{x}(\theta)) = \sum_{k=1}^N (x(k) - \hat{x}_{\theta}(k))^2, \quad (4.5)$$

and signal $\hat{x}_{\theta}[k]$ denotes the reconstructed seismic trace as in (4.5) with the corresponding parameter estimate θ .

Obviously, this is a multi-dimensional optimization problem, where the parameter to be optimized is θ and the optimization objective is to minimize e . Due to the large search space, the parameters are not easy to obtain. In addition, there are often multiple local optima in the landscape of e , so traditional optimization techniques are easy to get trapped in local optima [101, 102, 103].

The difficulties above can be solved when the particle swarm optimization (PSO) method is used to minimize the cost function. Since the PSO algorithm evaluates a high number of cost function points, the optimal parameter is selected from the points that satisfy equation (4.4). Moreover, PSO implementations are robust to initial population of model parameters, which is suitable when large number of parameters to be optimized are present in the model. This technique was further improved by introducing inertial weight

[104] to assure the particles to converge to the best point in the search trajectory [105, 103].

The PSO algorithm was initially developed by Kennedy and Eberhart [106], based on the social behavior of swarm of animals. Each particle that represents a single individual in the swarm, records the best solution found by itself and by the whole swarm along the search space. Each individual updates its positions in the search trajectory and exchanges information with other individuals, according to the following equations:

$$v_{p,d}^{t+1} = wv_{p,d}^k + c_1r_1(\theta_{p,d}^{ind} - \theta_{p,d}^t) + c_2r_2(\theta_{p,d}^{glo} - \theta_{p,d}^t), \quad (4.6)$$

$$\theta_{p,d}^{t+1} = \theta_{p,d}^t + v_{p,d}^{t+1}, \quad (4.7)$$

where

$v_{p,d}^k$ velocity of particle p on dimension d at iteration t ;

$\theta_{p,d}^t$ position of particle p on dimension d at iteration t ;

θ^{ind} individual best position found by the particle itself;

θ^{glo} global best position found by whole swarm;

r_1, r_2 random numbers with uniform distribution in the range $[0,1]$;

c_1 the individual fraction;

c_2 the social fraction;

w inertial weight.

Each search dimension corresponds to a parameter in (4.3). The process is terminated when either the maximum number of iterations is reached or the last change of the best solution is smaller than a pre-defined value for a number of iterations. A particle with the best cost function in (4.5) is selected as the best parameter set. The overall algorithm is summarized in Figure 4.1.

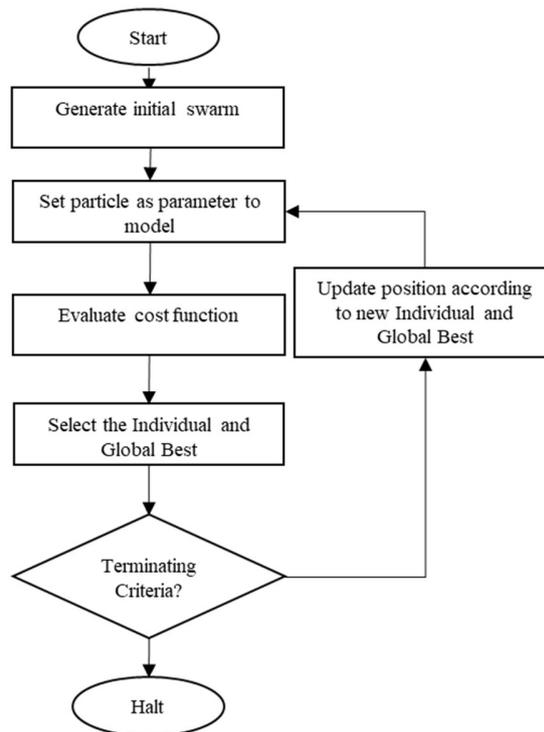


Figure 4.1: Single EDSW estimation with PSO

4.1.2 Multiple EDSWs estimation

In this section, the extension of single EDSW estimation to multiple EDSW is presented. The suitable number of EDSWs, M , should be adjusted as a compromise between number of parameter sets and the reconstruction quality. However, the value of M can not be pre-

determined. To deal with that problem, we propose a sequential scheme to estimate the parameters of the multiple EDSWs model (4.1) without approximating the value of M as follows

$$\theta_i^* = \arg_{\theta_i} \min \sum_{k=1}^N [x(k) - \sum_{j=1}^{i-1} s_{\theta_j^*}(k) - s_{\theta_i^*}(k)], \quad (4.8)$$

where θ_i^* denotes the optimal estimate of θ_i , which depends on the optimal estimates of the previous $i - 1$ parameter sets $\theta_1^*, \theta_2^*, \theta_3^*, \dots, \theta_{i-1}^*$. This scheme can be implemented on a given seismic trace $x[k]$ as in the following steps. First, assume that the seismic trace is modeled by only one EDSW ($M = 1$) as in (4.1). Then, apply the single EDSW estimation method in the previous section to $x[k]$ and obtain the first parameter estimate set θ_1^* . The signal residual from the first step is given as

$$r_{\theta_1^*}(k) = x(k) - s_{\theta_1^*}(k). \quad (4.9)$$

The signal residual energy is denoted by $r_{\theta_1^*}^2 = \|r_{\theta_1^*}\|^2$. Next, apply the single EDSW estimation method on $r_{\theta_1^*}[k]$ to obtain the second parameter estimate θ_2^* and second residual signal $r_{\theta_2^*}[k]$. This step is repeated and the parameter sets are estimated after discarding each component. At the i th iteration, the parameter of model $s_{\theta_i}[k]$ are estimated from previous residual signal $s_{\theta_{i-1}}[k]$, leading to the next residual as follows:

$$r_{\theta_i^*}(k) = x(k) - s_{\theta_i^*}(k). \quad (4.10)$$

$$r_{\theta_i^*}(k) = x(k) - \sum_{j=1}^{i-1} s_{\theta_j^*}(k). \quad (4.11)$$

The procedure is terminated once the residual energy is lower than a predefined threshold

r_E

$$r_{\theta_i^*}^2 < r_E, \quad (4.12)$$

where the residual energy is given as

$$r_{\theta_i^*}^2 = \sum_{k=1}^N (r_{\theta_i^*}(k))^2, \quad (4.13)$$

and its normalized residual energy represents the ratio of the residual energy with respect to the original signal energy

$$\gamma_{\theta_i^*} = \frac{\sum_{k=1}^N (r_{\theta_i^*}(k))^2}{\sum_{k=1}^N (x(k))^2}. \quad (4.14)$$

The procedure can also be terminated once the energy dropping rate is lower than a predefined threshold r_D , i.e.,

$$\frac{r_{\theta_{j-1}^*}^2 - r_{\theta_j^*}^2}{r_{\theta_{j-1}^*}^2} 100\% < r_D. \quad (4.15)$$

This scheme is illustrated in Figure 4.2 .

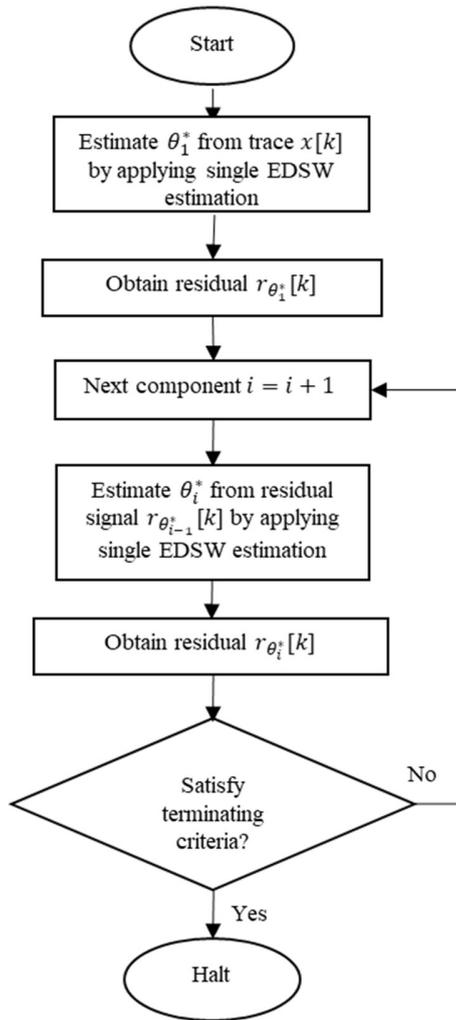


Figure 4.2: Multiple EDSW estimation

4.1.3 Residual encoding

Suppose that K traces are considered and each trace contains N samples, there are NK seismic time samples totally. The data volume of K original seismic data traces in L - bit format is $V = NK L$ where seismic data file format for seismic data typically uses 32-bit ($L = 32$) fixed length encoding. If the traces are reconstructed with M model

components where each consists of five parameters, then the reconstructed volume is

$V_M = NKM$. The compression ratio is given as:

$$Cr_M = \frac{V}{V_M} \quad (4.16)$$

$$C_r = \frac{L}{5M}. \quad (4.17)$$

The reconstructed traces \check{x} are obtained from M model components estimated from each trace on average and the residuals $r = x - \check{x}$ are encoded with fixed-length codewords quantization where each code-word uses the same number of bits. This scheme improves the reconstruction quality but still results in a lossy compression. This quantization maps each residual sample to one of a small number of symbols. The symbols are designed by applying k-means algorithm [107] on the residual samples. Each symbol is then encoded as a $l = \log_2 L_q$ bit code-word, where L_q is the number of codewords, or the quantization level. Therefore, the volume of those residuals and the $5M$ parameters coded in L bits is

$$V_{rec} = KNl + 5MKL. \quad (4.18)$$

The compression ratio of the proposed method is

$$Cr_{rec} = \frac{V}{V_{rec}} \quad (4.19)$$

$$Cr_{rec} = \frac{NL}{Nl + 5ML}. \quad (4.20)$$

After applying quantization to the residuals $Q(r)$, we can calculate the normalized reconstruction error energy that represents the ratio of the reconstruction error energy with respect to the original signal energy.

$$\gamma_e = \frac{\sum_{k=1}^N (x(k) - (\check{x}(k) + Q(r(k))))^2}{\sum_{k=1}^N (x(k))^2}. \quad (4.21)$$

4.2 Experimental results

We investigate the performance of our scheme using the East Texas dataset. The proposed scheme is applied on the dataset and its performance is evaluated and compared with the DCT.

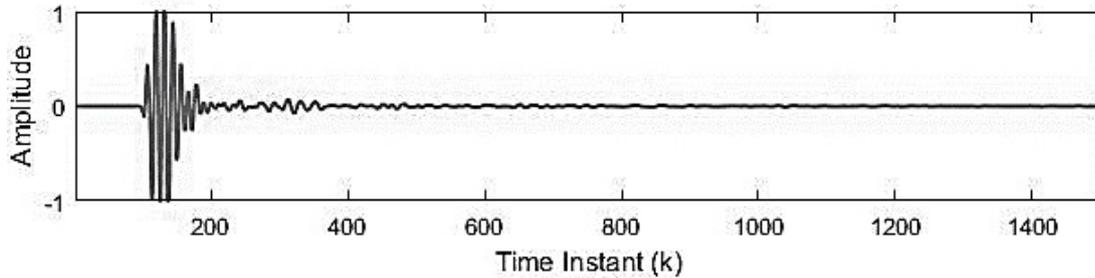


Figure 4.3: An Original Normalized Seismic Trace Sample

4.2.1 Parameter Estimation

In our experiment, we used 100 particles to search the optimal parameter set with a maximum of 100 iterations, and the individual and social fraction are equally set to 1.49 $c_1 = c_2 = 1.49$. We take a sample trace plotted in Figure 4.3, as an example to show the behavior of the sequential parameter estimation scheme. The first five EDSWs, s_1, \dots, s_5 , estimated from the sample trace are plotted in Figure 4.4. The constructed traces with one

EDSW ($M = 1$), two EDSWs ($M = 2$), up to five EDSWs ($M = 5$) are plotted in Figure 4.5(a)-(e), respectively. The corresponding residual signals are plotted in Figure 4.6(a)-(e), respectively. It can be easily noticed that each of them reduces significant component in the residuals with large energy.

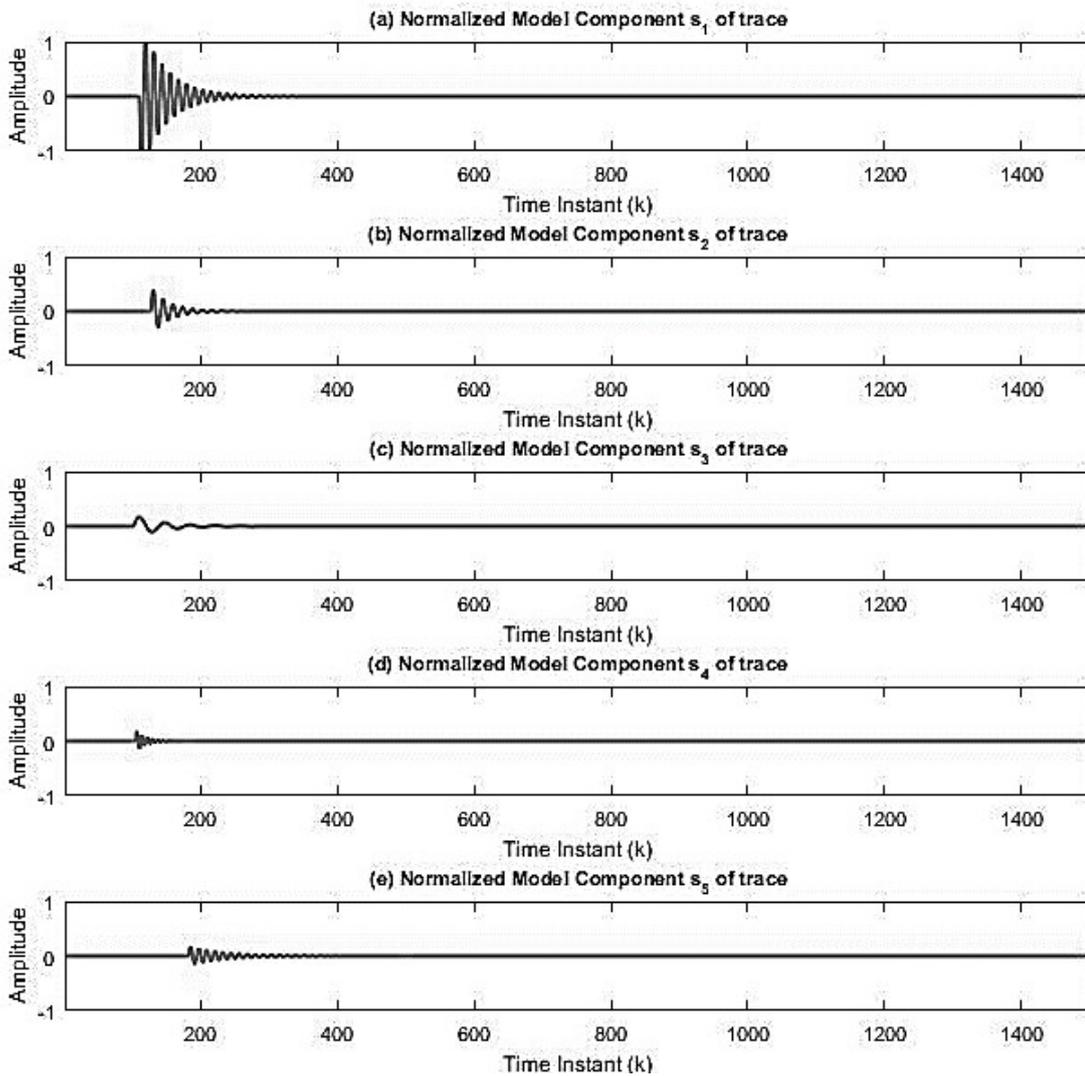


Figure 4.4: The model components s_1, s_2, s_3, s_4, s_5 of a trace sample

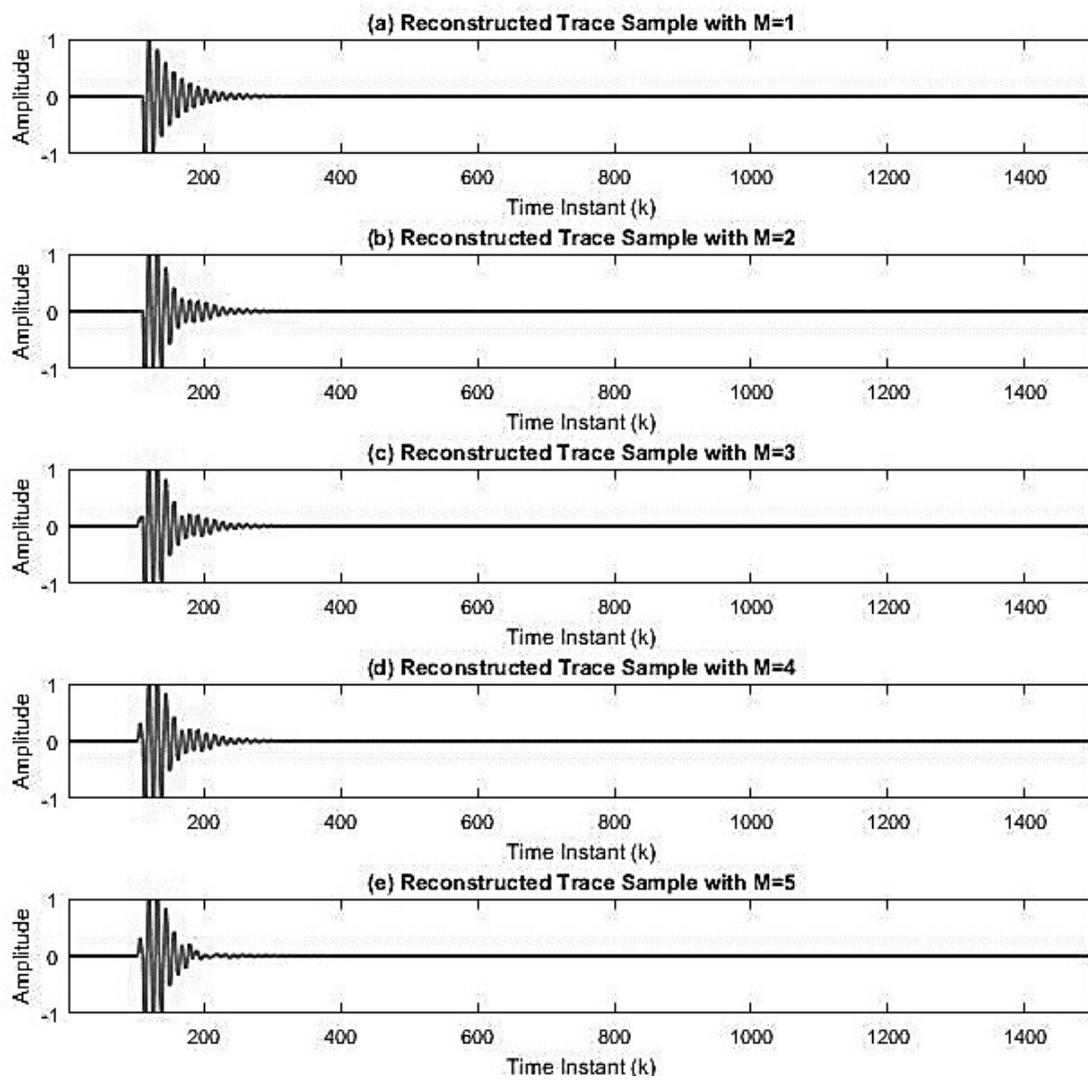


Figure 4.5: The reconstructed sample trace with $M = 1, 2, 3, 4, 5$.

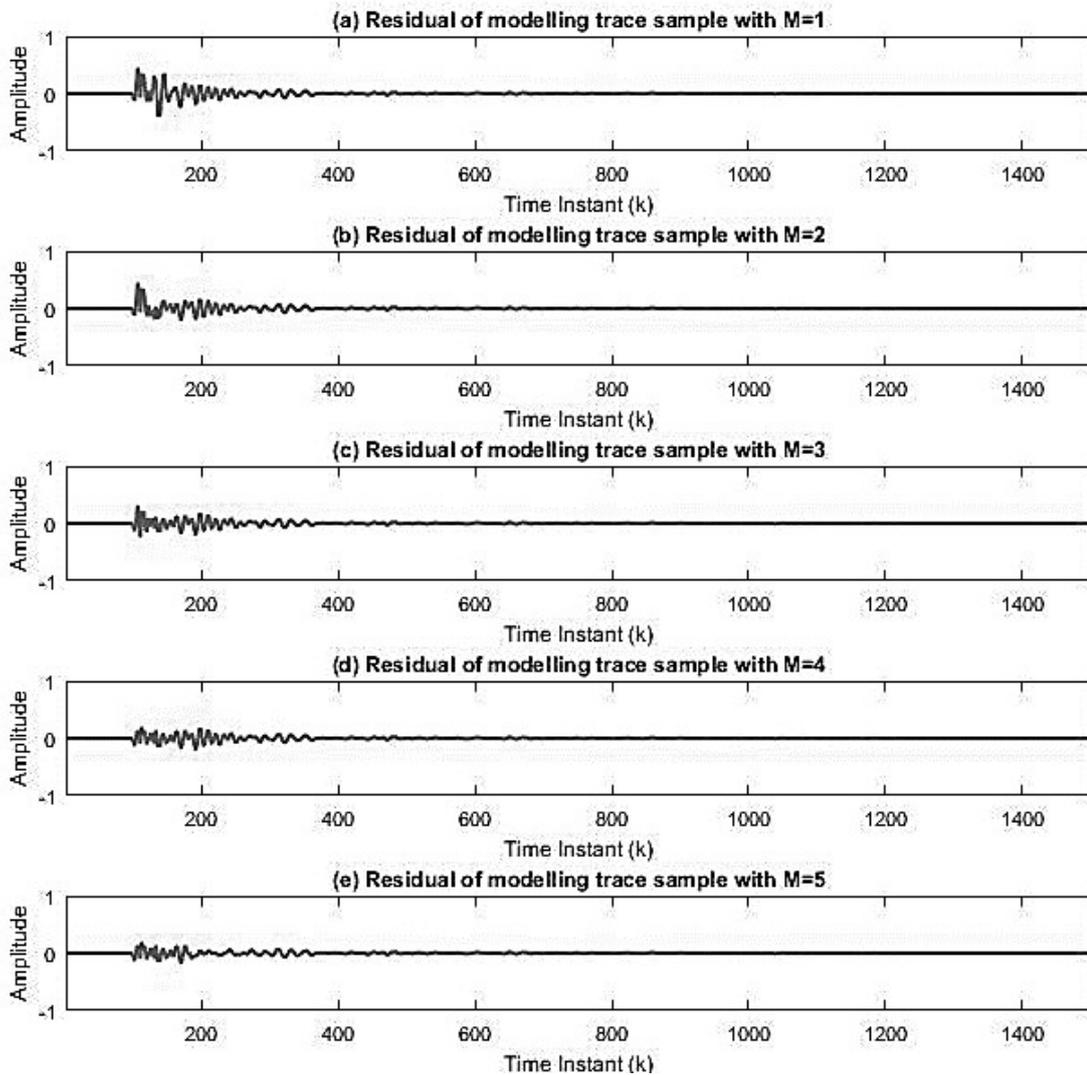


Figure 4.6: The residual of modelling sample trace with $M = 1, 2, 3, 4, 5$.

We take another sample trace, plotted in Figure 4.7, as an example to show the behavior of the sequential parameter estimation scheme. The reconstructed traces by using different numbers of EDSWs ($M=1,4,7,10$) are respectively shown in Figure 4.7 (b), (c), (d), and (e). The normalized residual energy is plotted as a function of M in Figure 4.8. It can be easily noticed that each component reduces the residual energy significantly.

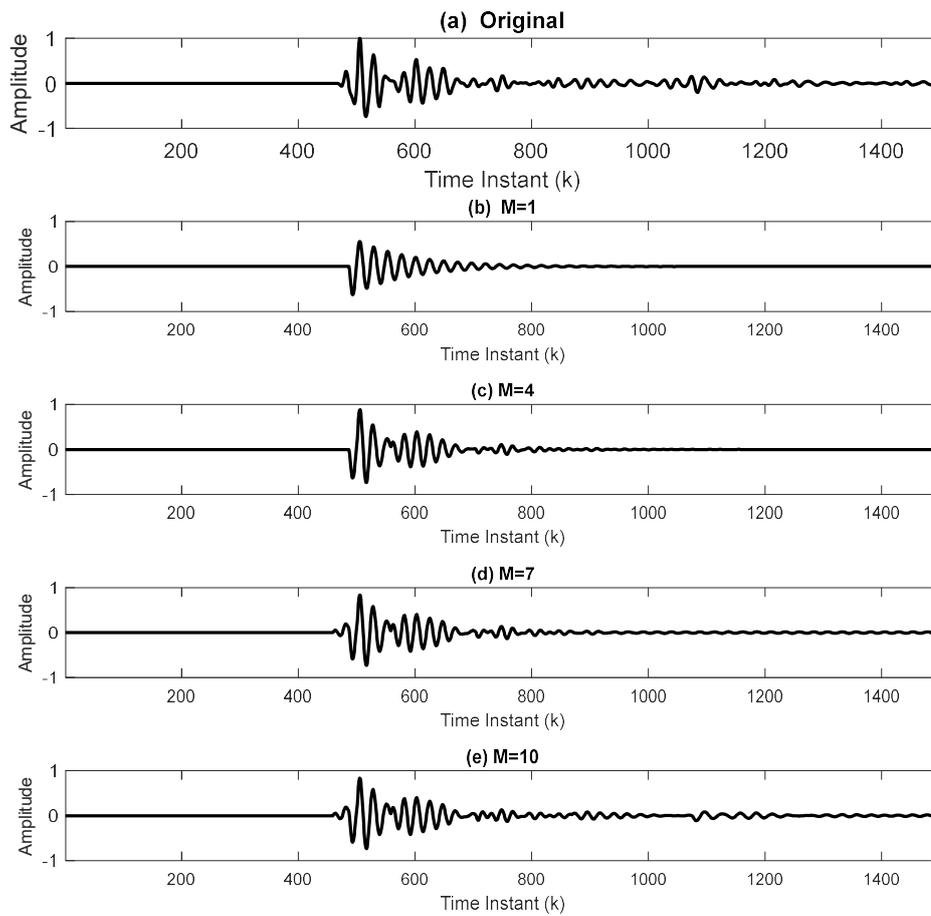


Figure 4.7: Another Sample Trace Reconstruction

(a) original, (b) The reconstructed sample trace with $M=1$, (c) $M=4$, (d) $M=7$, (e) $M=10$.

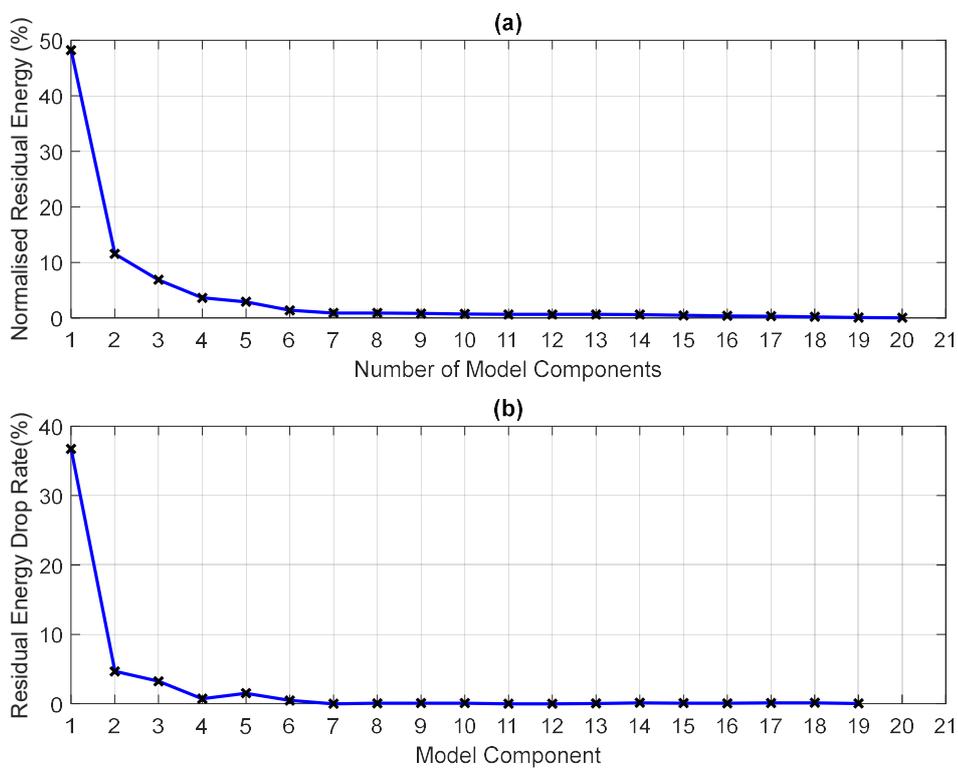


Figure 4.8: The energy of the normalized residuals for sample trace

(a) Normalized residual energy of the trace sample, (b) Energy drop rate of the trace sample.

The proposed method is then applied on all traces. The average normalized residual energy, in percentage, of all traces is shown as a function of M in Figure 4.9 (a). Its corresponding compression ratio is shown in Figure 4.9 (b). If the residuals are discarded, the compression ratio is calculated based on the estimated model parameters. It can be noticed that this scheme achieves a very high compression ratio. The compression ratio ranges from 15:1 to 300:1 as the number of model components changes from $M = 20$ to $M=1$. We compare our method with Expectation Maximization (EM) based parameter estimation with six parameter model where the extra parameter is a constant. Direct

mapping of normalized residual energy to compression ratio is plotted in Figure 4.9. It can be seen that PSO-based offers comparable results at high compression ratio.

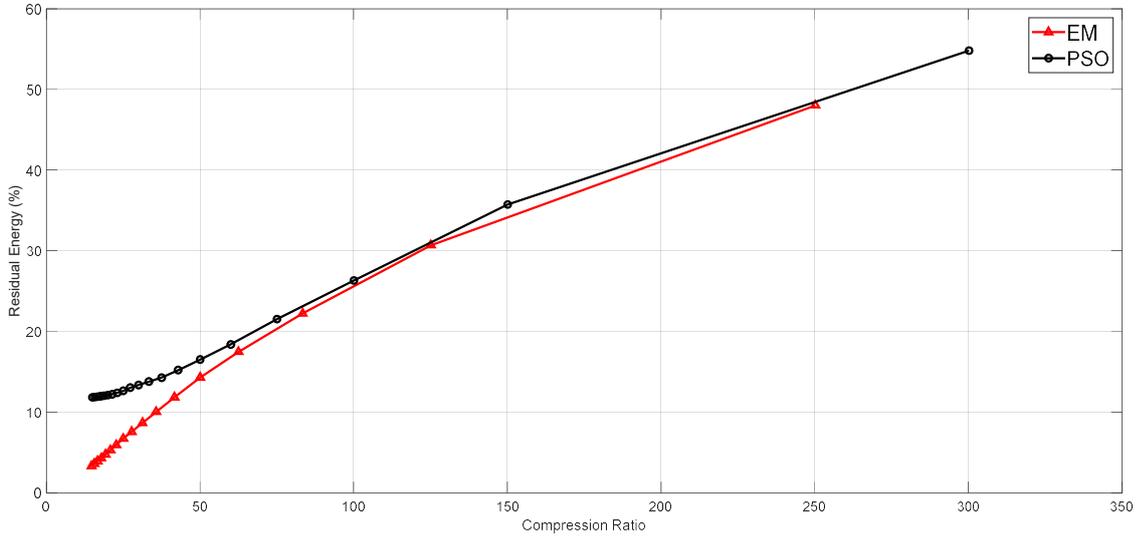


Figure 4.9: Normalized Residual Energy comparison as function of Compression Ratio.

4.2.2 Encoding the residuals

To improve the reconstruction quality, scalar quantization is applied to the residual signals. In this section, we implement the k-means algorithm to design three codebooks, the first codebook is with 16 quantization levels (4 bits per sample), the second is with 32 quantization levels (5 bits per sample), and the last is with 64 quantization levels (6 bits per sample). The residuals of the first 300 traces are used to train the codebooks then the remaining residuals are encoded using the codebooks for testing. The average normalized reconstruction error energy is plotted in Figure 4.10 (a). It is obvious that the scalar quantization of the residuals reduces the reconstruction error energy as we can see in Figure 4.10 (a) compared with Figure 4.9 (a). As a trade-off, the compression ratio decreases as exhibited in Figure 4.10 (a) compared with that of model only reconstruction in Figure 4.9 (a).

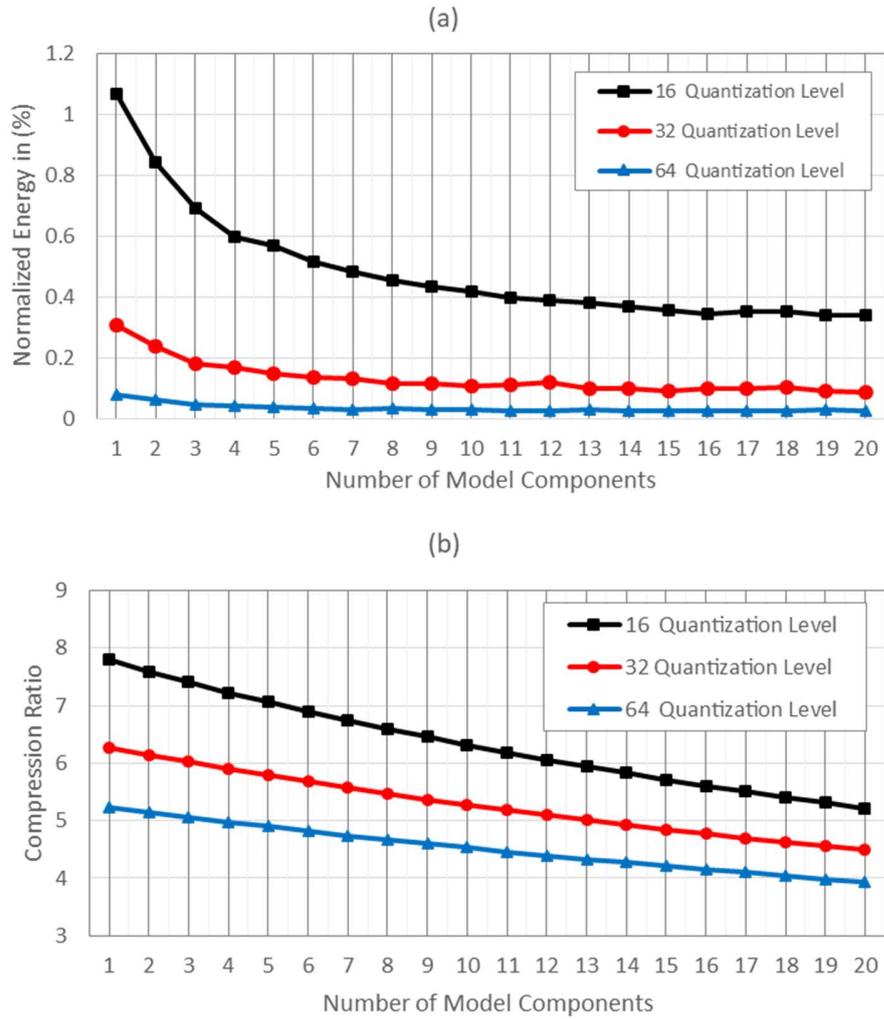


Figure 4.10: The reconstruction quality with residual quantization

- (a) The average normalized reconstruction error energy, in percentage, of all traces as a function of M , the number of model components; (b) Compression Ratio with residuals discarded as a function of M .

4.2.3 Comparison with DCT

The DCT is shown to have the closest performance to the optimal transform like PCA or KLT [35]. The transformation basis for the DCT is data-independent since it consists of a set of cosines with discrete frequencies. The transformation is defined as follows [37],

$$y_c(n) = 2 \sum_{k=0}^{N-1} x(k) \cos\left(\frac{\pi(2k+1)n}{2N}\right), \quad (4.22)$$

for $n = 1, 2, \dots, N - 1$. The inverse is defined as follows.

$$x(k) = \frac{1}{N} \left(\frac{y_c(0)}{2} + \sum_{n=0}^{N-1} y_c(n) \cos\left(\frac{\pi(2k+1)n}{2N}\right) \right), \quad (4.23)$$

for $k = 1, 2, \dots, N - 1$.

On many types of data, a signal can often be reconstructed using only a few frequency components. This property is useful for compression where the data is represented with a few parameters. The compression ratio is defined as follows.

$$C_c = \frac{N}{J}, \quad (4.24)$$

where N and J denote the number of time samples and selected frequency components, respectively. We apply the same procedure to enhance the reconstruction quality by quantizing the residuals and encode the quantized residual using fixed-length codewords. Therefore, the volume of the l bits encoded residuals and J selected frequency components coded in L bits is

$$C_{cq} = \frac{NL}{Nl + JL}. \quad (4.25)$$

Notice that the compression ratio for each EDSW model components equals to the compression ratio of five DCT components. This is due to the fact that each DCT component is only defined with a single value which is the frequency coefficient. For

DCT, the components are selected from the J lowest frequency components. This is due to the fact that most energy of a seismic trace concentrates at the lower frequency spectrum [16]. Table 4.1 summarizes the compression ratio of the DCT and EDSW corresponding to the residual energy. It can be noticed that our proposed method outperforms the DCT in term of the residual energy for the same compression ratios.

Table 4.1: DCT and EDSW residual energy

Compression Ratio	DCT (%)	EDSW (%)
300:1	99.98	54.8
150:1	99.95	35.72
100:1	99.92	26.31
75:1	99.90	21.52
60:1	99.86	18.37
50:1	99.80	16.5
30:1	99.05	13.35
25:1	99.1	10.94
20:1	95.28	8.82
15:1	84.8	6.82

We used the DCT with number of frequency components $J = 5, 10, 15, \dots, 100$ that correspond to the same compression ratio of using EDSW with $M = 1, 2, 3, \dots, 20$. Recall that single EDSW component consists of five parameters while single DCT component is only represented by a single parameter. The normalized reconstruction error energy are plotted in Figure 4.11 (a),(b), and (c) for 16-levels, 32-levels, and 64-levels of quantization, respectively. From the figure, it is noticed that the reconstruction error of

the DCT is insensitive to the number of frequency components J . From all cases, the first five components of DCT show good performance but adding more frequency components does not progressively improve the performance.

The DCT is further investigated using the same residual encoding with number of frequency components $J = 50, 100, 150, \dots, 200$. The normalized reconstruction error energy of the DCT are plotted in Figure 4.12(a), (b), and (c) for 16-levels, 32-levels, and 64-levels of quantization, respectively. The normalized reconstruction error energy for the proposed method for the same compression ratio is shown in the same figure. It can be noticed that the proposed method achieves a lower reconstruction error energy than that of the DCT.

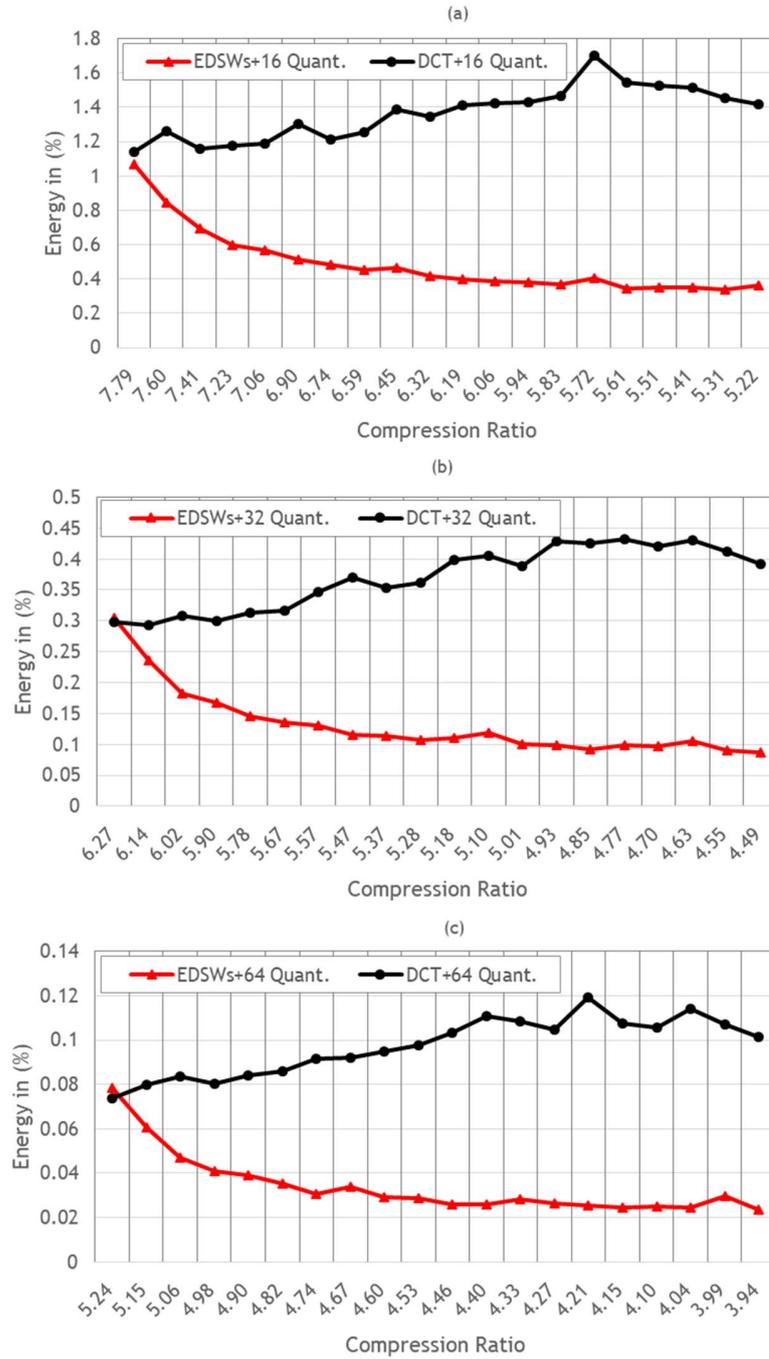


Figure 4.11: The reconstruction quality with residual quantization of DCT and EDSW

(a) Normalised Reconstruction Error with 16 Quantization level (b) Normalised Reconstruction Error with 32 Quantization level (c) Normalised Reconstruction Error with 64 Quantization level.

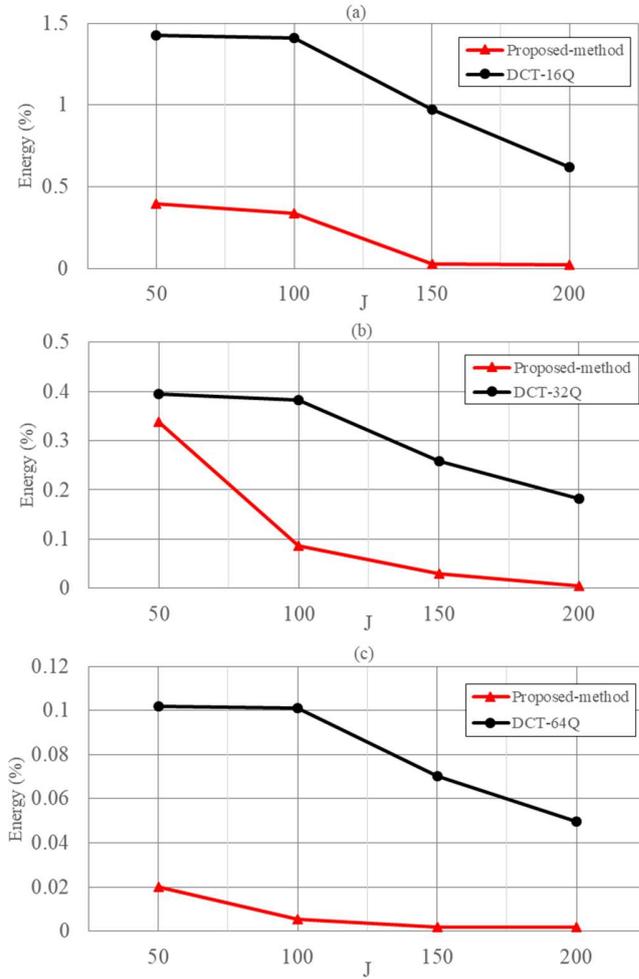


Figure 4.12: The reconstruction quality with residual quantization of the proposed method $J = 50, 100, 150, 200$
 (a) Normalized Reconstruction Error with 16 Quantization level (b) 32 Quantization level, and (c) 64 Quantization level.

4.3 Concluding remarks

In this chapter, we developed a model-based compression scheme for seismic data. First, each seismic trace was modeled as a superposition of exponentially decaying sinusoidal waves (EDSWs). Each EDSW was regarded as a model component and defined by a set of parameters. Secondly, a parameter estimation procedure was devised to minimize the

residual energy of this model. In the procedure, the parameter set was estimated component by component. A suitable number of model components was selected by the level of the residual energy during parameter estimation process. This step led to a very high compression ratio from 300:1 with a much lower residual energy than that of the DCT. It was shown that the residual energy is decreased by adding more model components. To improve the reconstruction quality, the residuals were encoded using fixed length code-word scalar quantization. For the same compression ratios, our proposed method achieved error energy ranging from 0.02% to 0.001% while the DCT achieved normalized energy ranging from 0.1% to 0.04%. The proposed method, in average, showed much lower distortion than that of the DCT for the same compression ratios.

CHAPTER 5

MACHINE LEARNING APPROACHES

5.1 Introduction

In this chapter, we present machine learning and deep learning (DL) approaches for seismic data compression. The DL is a class of algorithms to train the deep neural networks (DNN) where the neural networks (NN) have many hidden layers [108]. There are several popular architectures for DNN such as deep auto-encoder (DAE), convolutional neural networks (CNN), and recurrent neural networks (RNN). The DAE is the most suitable architecture for data compression using dimensionality reduction, and restricted Boltzmann machine (RBM) is the most suitable DL algorithm for this architecture. The DAE have been used with outstanding results for dimension reduction [109], and MNIST dataset digits classification [110, 111]. The success of DAE to perform dimensionality reduction shows its potential for data compression. Other architectures were also investigated for regression in [112, 113].

Machine learning approach has become one of the most prominent compression methods since it is able to model the structure of the data from multiple samples [114, 115]. Therefore, data compression techniques using machine learning can be found in many recent studies. A near lossless biomedical data compression is presented using NN in [116]. Tan et.al. [117] applied a multi-layer auto-encoder for mammogram image

compression. The training is performed using image patches instead of the whole images. In [118], Srivastava proposed a deep belief networks (DBN) with multimodal inputs to learn representation from data and image for information retrieval tasks. In [119], Yann Olivier proved that minimizing the code length of the data and minimizing reconstruction error of an auto-encoder are highly correlated. In [120], authors used the auto-encoder for electro-cardiogram (ECG) compression. For seismic data, Huang [54] showed that NNs can be used to find the principal components for compression.

In this chapter, we present three machine learning and deep learning approaches for seismic data compression. First, a compression scheme using auto-encoder and RBM for infield seismic data compression is presented. We show that the RBM is able to enhance the performance of NN for data compression during seismic data acquisition. Second, we develop the DNN for prediction based data compression. This technique is able to achieve a very high reconstruction quality with medium compression ratios. Lastly, we introduce an alternative architecture and learning algorithm for DNN using Extreme Learning Machine (ELM). In contrast to the Back-Propagation (BP) and RBM, the ELM offers non-iterative training procedure for NN. We develop an asymmetric DAE architecture for seismic data compression. This asymmetric architecture offers a fast training procedure for the DAE with a similar performance with DNN using the RBM and BP.

This chapter is organized as follows: The implementation of RBM for auto-encoder for seismic data compression is presented in section 5.2. Prediction based compression using neural networks is discussed in section 5.3. Finally, the DAE using ELM for seismic data compression is presented in section 5.4.

5.2 Restricted Boltzmann Machine for Auto Encoder

In this section, we develop a practical scheme of NN with a single hidden layer for infield seismic data compression. We utilize the RBM pre-training to the NN to obtain a set of parameters that are close to the optimal ones. Then we use the scaled conjugate gradient (SCG) algorithm [121] to fine-tune the NN. Once the NN is trained with a portion of data, the weights are broadcasted to all geophones. Each geophone is then able to compress its own seismic data. Since seismic data contains high redundancy both temporal and spatial, using a portion of early acquired data for training is expected to be able to compress the upcoming data with a good quality.

5.2.1 Methodology

We use a feed-forward neural networks with auto-encoder architecture as shown in Figure 5.1 for data compression. The architecture consists of three parts: encoder, code, and decoder layers. The encoder and decoder parts do not necessarily have a single layer but in this subsection, to achieve fast encoding and decoding operations, we use only a single layer for both parts. Therefore, the architecture is equivalent to an NN with an input layer with N_I units, an output layer with N_O units and a hidden layer with N_C units, where the number of units in the input layer is equal to the number of units in the output layer, i.e., $N_I = N_O$. Each hidden unit, say unit j , linearly maps all inputs from the input layer, x_i , to a scalar value, y_j as:

$$y_j = b_j^E + \sum_{i=1}^{N_I} x_i w_{ij}^E, \quad (5.1)$$

where b_j^E is the bias of unit j in the hidden layer, x_i is the i th input, and w_{ij}^E is the weight that connects unit i of the input layer to unit j in the hidden layer. Similarly, each unit in the output layer, say unit k , is a linear mapping of the outputs of the hidden units:

$$\hat{x}_k = b_k^D + \sum_{j=1}^{N_C} y_j w_{jk}^D, \quad (5.2)$$

where b_k^D is the bias of unit k in the output layer, and w_{jk}^D is the weight that connects unit j of the hidden layer to unit k in the output layer. For the sake of simplicity, we stack weights w_{ij}^E into matrix $\mathbf{W}^E \in \mathbb{R}^{N_I \times N_C}$, where the element in the i th row and j th column is w_{ij}^E . Then we stack bias b_j^E into vector $\mathbf{b}^E \in \mathbb{R}^{N_C}$, where the j th element of \mathbf{b}^E is b_j^E . Similarly, matrix $\mathbf{W}^D \in \mathbb{R}^{N_C \times N_O}$ and vector $\mathbf{b}^D \in \mathbb{R}^{N_O}$ are defined for parameters w_{jk}^D and b_k^D , respectively.

To perform seismic data compression, each seismic trace, say $\mathbf{x} \in \mathbb{R}^{N_I}$, is fed into the NN as an input vector, then the output of the hidden layer, $\mathbf{y} \in \mathbb{R}^{N_C}$ is regarded as the compressed representation of \mathbf{x} . The output of the NN, $\hat{\mathbf{x}} \in \mathbb{R}^{N_O}$ becomes the reconstructed trace of \mathbf{x} . The compression ratio is defined as the ratio between the length of the original seismic traces and the length of the compressed traces, i.e.,

$$c_r = \frac{N_I}{N_C}. \quad (5.3)$$

The main target of this NN is to obtain weights and biases $\Theta = \{\mathbf{W}^E, \mathbf{W}^D, \mathbf{b}^E, \mathbf{b}^D\}$ that minimize the discrepancy between the input seismic traces and the reconstructed output seismic traces:

$$E(\Theta) = \frac{1}{MN} \sum_{i=1}^M \|\hat{x}_i(\Theta, x_i) - x_i\|^2, \quad (5.4)$$

where M is the number of traces for training, x_i is the i th trace in the training set and its corresponding output is $\hat{x}_i(\Theta, x_i)$. In the implementation, the fusion center collects traces of first several shots from all geophones. The NN is trained using the collected traces. To achieve near optimal initial NN parameters with a good generalization performance, the RBM is used to pre-train the encoder part of the NN using the collected traces. The roughly trained NN is further fine-trained using SCG method. Next, the encoder weights of the trained NN are then broadcasted to the geophones.

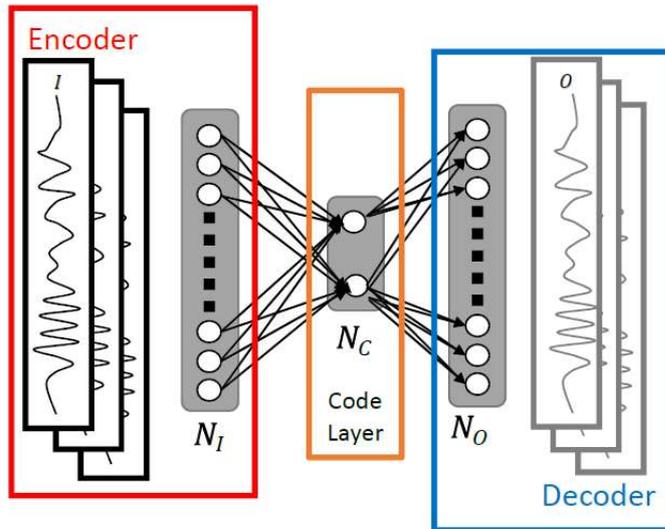


Figure 5.1: Auto-Encoder with a single hidden layer

A. Unsupervised Learning with RBM for linear hidden units

In this sub-section, we discuss the RBM pre-training for NN with linear hidden units. The pre-training is expected to provide near optimal initial weights and to enhance the generalization capability since most of the information in the weights is drawn from sampling the data [109, 108]. Since the NN architecture is symmetric and both parts are associated with the same data, we only need to pre-train the encoder part as illustrated in Figure 5.2, where the units of input layer are called visible units $\mathbf{v} \in [0,1]^{N_I}$ and the hidden units $\mathbf{h} \in \mathbb{R}^{N_c}$ model dependencies between the elements of observations. In the RBM training, we denote visible and hidden layers biases as $\mathbf{b} \in \mathbb{R}^{N_I}$ and $\mathbf{c} \in \mathbb{R}^{N_c}$, respectively. The weights between the two layers are denoted as $\mathbf{W} \in \mathbb{R}^{N_I \times N_c}$.

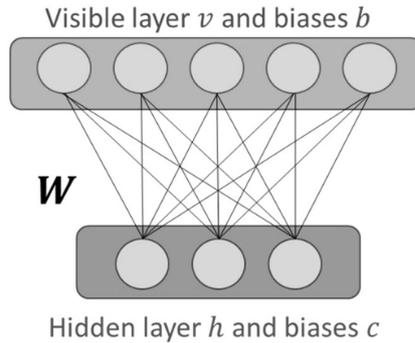


Figure 5.2: A Restricted Boltzmann Machine

The initial values of the weights and biases are set to be random with Gaussian distribution $N(0,0.1)$. The RBM pre-training is implemented by performing two phases, i.e. positive and negative phases. First, in the positive phase, we set the training vector x to be the values of the visible units (\mathbf{v}^+). The value of each hidden unit (h_j^+) is set to a real random value with the following distribution

$$p(h_j^+) = \mathcal{N}(\hat{h}_j^+, s) \quad (5.5)$$

$$\hat{h}_j^+ = c_j + \sum_i v_i^+ w_{ij}, \quad (5.6)$$

where s is the variance of the training data. The values from the positive phase h_j^+ are used to calculate the values in the negative phase as follows

$$\hat{v}_i^- = \sigma\left(b_i + \sum_j h_j^+ w_{ij}\right), \quad (5.7)$$

$$\hat{h}_j^- = c_j + \sum_i \hat{v}_i^- w_{ij}, \quad (5.8)$$

where $\sigma(x) = \frac{1}{1+\exp(x)}$ is the logistic sigmoid function. Once all values from the positive and negative phases are obtained for all input vectors, the weight and bias updates for each iteration are given by

$$\Delta w_{ij} = \varepsilon_w (\langle v_i \hat{h}_j \rangle^+ - \langle \hat{v}_i \hat{h}_j \rangle^-) \quad (5.9)$$

$$\Delta b_j = \varepsilon_b (\langle v_i \rangle^+ - \langle \hat{v}_i \rangle^-) \quad (5.10)$$

$$\Delta c_j = \varepsilon_c (\langle \hat{h}_j \rangle^+ - \langle \hat{h}_j \rangle^-), \quad (5.11)$$

where $\varepsilon_w, \varepsilon_b$, and ε_c are the step sizes for weights, visible and hidden layer biases, respectively. This procedure is repeated until the maximum number of iteration is reached.

The NN then uses the weights obtained from the unsupervised training as its initial weights, given by

$$\Theta_0 = \{\mathbf{W}_0^E, \mathbf{W}_0^D, \mathbf{b}_0^E, \mathbf{b}_0^D\} = \{\mathbf{W}_{RBM}, \mathbf{W}_{RBM}^T, \mathbf{c}_{RBM}, \mathbf{b}_{RBM}\}, \quad (5.12)$$

where the subscript RBM indicates the weights and biases estimated from RBM.

B. Scaled Conjugate Gradient for Supervised Training

In this sub-section, we briefly discuss the supervised training used in our scheme. The NN weights and biases Θ are further fine tuned using SCG to minimize the cost function given in (5.4). In this algorithm, the weights are updated using steepest gradient descent \mathbf{p} . A Lagrange multiplier λ is used to regulate the indefiniteness of the second partial derivative of the cost function $E''(\Theta) = \nabla^2 E(\Theta)$ to determine the global minimum of the cost function.

The main idea of the SCG is summarized as follows. For iteration k , the parameters are updated as,

$$\Theta^k = \Theta^{k-1} + \lambda^{k-1} \mathbf{p}^{k-1}, \quad (5.13)$$

the Hessian for this algorithm is approximated by

$$[\nabla^2 E(\Theta^k)] \mathbf{p}^k = \left(\nabla E(\Theta^k + \sigma^k \mathbf{p}^k) - \nabla f(\Theta^k) \right) / \sigma^k + \lambda^k \mathbf{p}^k = \mathbf{s}^k, \quad (5.14)$$

where $0 < \sigma^k \leq 1$, and the optimized step size is given as

$$\lambda^k = -\nabla^T E(\Theta^k) \mathbf{p}^k / ((\mathbf{p}^k)^T \mathbf{s}^k + \lambda^k |\mathbf{p}^k|^2), \quad (5.15)$$

and λ^k is a scalar, whose value is determined by the comparison parameter Δ^k ,

$$\Delta^k = (E(\Theta^k) - E(\Theta^k + \lambda^k \mathbf{p}^k)) / (E(\Theta^k) - \lambda^k \mathbf{p}^k), \quad (5.16)$$

if $\Delta^k > 0.75$ then $\lambda^k = \frac{1}{2} \lambda^{k-1}$ and if $\Delta^k < 0.25$ then $\lambda^k = 4 \lambda^{k-1}$.

We use the SCG for supervised training instead of standard BP algorithm due to its robustness. Regular back-propagation algorithm is highly dependent on user-defined parameters such as learning rate and momentum constant [121, 122]. Those values are usually determined using trial-and-error. Therefore, it is impractical to use regular back-propagation in the field. On the other hand, the SCG offers self-adjusted parameters inside the algorithm shown above.

C. Computational Complexity

Each trace is a real valued vector $\mathbf{x} \in \mathbb{R}^N$. Given the encoder weights $\mathbf{W}_E \in \mathbb{R}^{N \times N_c}$ and biases $\mathbf{b}_E \in \mathbb{R}^{N_c}$, the compression transformation in matrix form is given by

$$\mathbf{y} = \mathbf{W}_E^T \mathbf{x} + \mathbf{b}_E, \quad (5.17)$$

where $\mathbf{y} \in \mathbb{R}^{N_c}$ denotes the compressed data. This process is a very light-weight since it takes only $N \times N_c$ multiplication and $N \times N_c$ addition operations, hence suitable for geophone implementation.

The decompression procedure has the similar process where the compressed data $\mathbf{y} \in \mathbb{R}^{N_c}$ is transformed back to its original dimension using decoder weights $\mathbf{W}_D \in \mathbb{R}^{N_c \times N}$ and biases $\mathbf{b}_D \in \mathbb{R}^N$. The transformation in matrix form is given by

$$\mathbf{x} = \mathbf{W}_D^T \mathbf{y} + \mathbf{b}_D, \quad (5.18)$$

where $\mathbf{x} \in \mathbb{R}^N$ denotes the reconstructed data. It takes only $N \times N_C$ multiplication and $N \times N_C$ addition operations which are easy tasks for a data center computer.

D. Compression Strategy

The proposed compression scheme consists of three stages. First, the fusion center collects training data from traces generated in the early shots. Secondly, the center constructs a NN for compression. This stage starts with pre-training the NN with RBM using the training data and then the NN is fine-tuned with SCG. Finally, the center broadcasts the encoder part of the NN to all geophones. Each geophone is then able to compress newly acquired traces from the upcoming shots using the encoder part and transmits the compressed trace to the center. The center or receiver can reconstruct the compressed data using decoder part of the NN. This scheme reduces the communication cost between geophone and the center. Since the NN is with a single layer and uses linear units, compressing each trace can be done very fast. The training may take time, but it is performed at the fusion center that possesses a high computing capability.

5.2.2 Experimental results

We evaluate our proposed scheme using two different real seismic datasets namely, East Texas and UTAM Avenue datasets to evaluate our proposed scheme. Both datasets acquisition systems match our scheme where the shots are ignited sequentially. We assess the reconstruction quality using the peak signal to noise ratio (PSNR). The PSNR is defined as :

$$\text{PSNR} = 10 \log_{10} \frac{x_{max}^2}{\frac{1}{MN} \sum_{i=1}^M ||\mathbf{x}_i - \hat{\mathbf{x}}_i||^2}, \quad (5.19)$$

where x_{max} is the maximum value in the data. Before training the NN, each trace is normalized between 0 to 1 and aligned such that the maximum value is shifted to the center of the trace.

The simulations were performed using a 2.4GHz intel core i7 processor using MATLAB software. In following sub-section, we show the detail of training process of our proposed method, and we investigate how many traces are required to train the NN for real-time seismic data compression. Finally, we compare our method with the Linear Predictive Coding (LPC).

A. Experiments using East Texas Dataset

The East Texas data set [100] is used in this section to evaluate the proposed method. This set consists of 594 seismic traces and each has 1501 time samples. Those traces are acquired by recording 18 seismic shots using 33 geophones. The interval between geophones is fixed 220ft but shots interval is varying between 220ft and 1100ft. The shots are generated by dynamite explosion at 80-100ft of depth. The shots are ignited sequentially, and are recorded by all geophones. The acquisition scheme is illustrated in Figure 5.3.

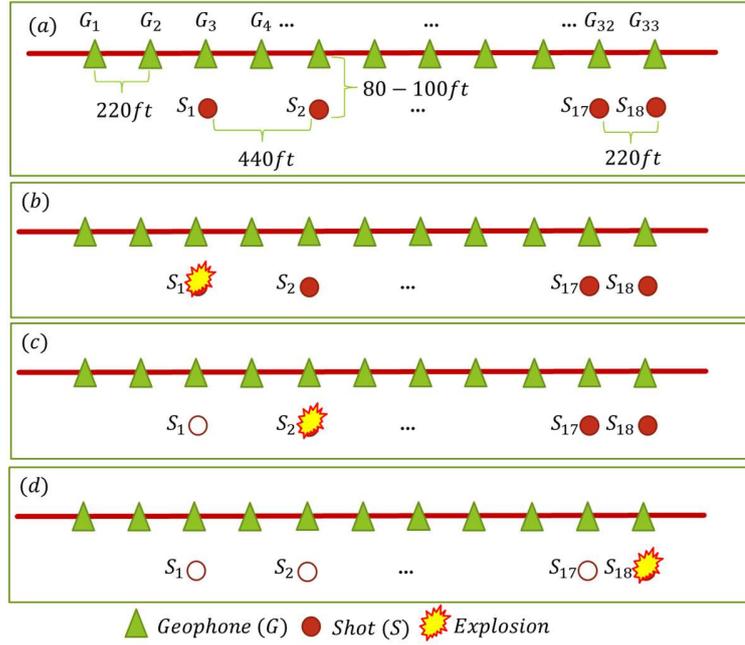


Figure 5.3: Seismic acquisition system (a) geophones and shots layout (b) first shot (c) second shot (d) final (18th) shot

In this experiments, the maximum number of iterations is set to 20 and 15000 for the RBM pre-training and the SCG supervised training, respectively. The supervised training is terminated if the cost function of the validation data does not decrease after ten iterations or training gradient is less than 10^{-5} . The code layer uses 150 units to achieve 10:1 compression ratio.

In the first experiment, we take 264 traces from first eight shots. We use 99 traces from first three shots as training data, 33 traces from the next shot as validation data, and other 132 traces from four shots after the first four shots as testing data. The RBM is pre-trained using the training data. The weights obtained from the RBM represent the features learned as shown in Figure 5.4 sorted by from the largest l_2 -norm. We plot the principal components (PCs) of the data in Figure 5.5 sorted by largest eigenvalues. We can see that the features learned by the RBM roughly resemble the PCs.

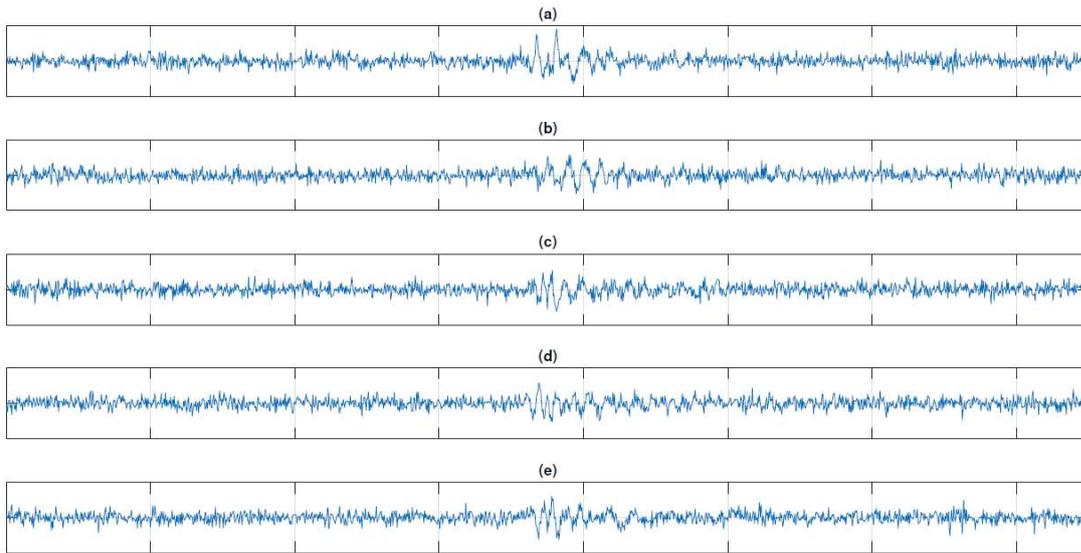


Figure 5.4: Sample of the features learned

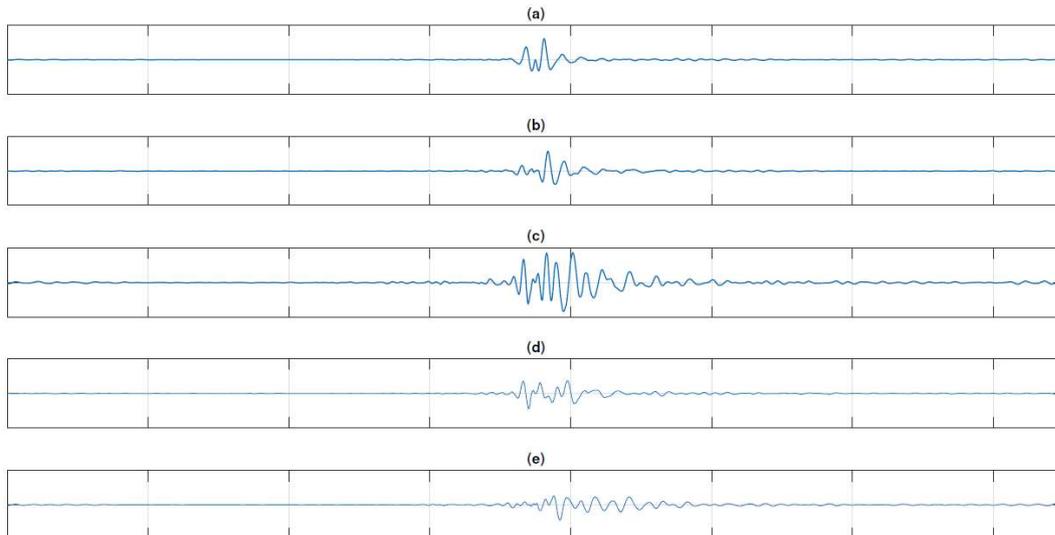


Figure 5.5: Sample of principal components of the data

Further, the NN is fine-tuned using supervised training with MSE as the cost function. In the training, the MSE of all training, validation, and testing data are evaluated but only gradient from the training data is used to update the NN weights. The learning

performance is plotted in Figure 5.6. It can be easily observed that the learning performance of the NN is significantly improved in two aspects. First, the unsupervised learning with RBM enhances the learning performance for both training and testing data. Second, the RBM improves the generalization capability. Without RBM pre-training, the cost function of the testing data starts to diverge from that of the training data at early training, only after 20 iterations. Meanwhile, with help of RBM, this phenomenon occurs after 200 iterations. This indicates that the unsupervised learning with RBM improves the generalization capability since the NN performance using the testing data is as good as the training data. Therefore, the newly acquired data in the future can be compressed with a very close quality as the past training data.

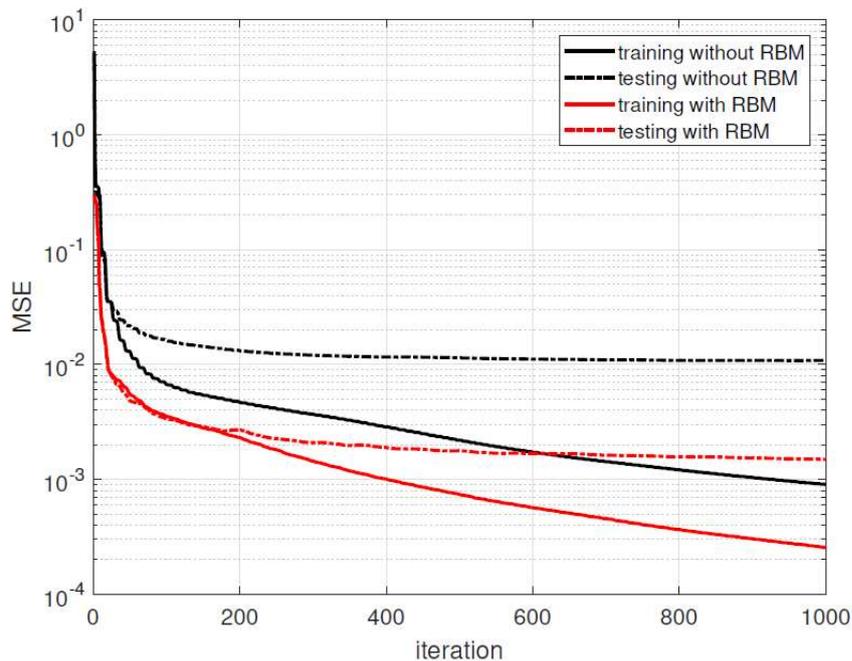


Figure 5.6: Learning Rate during Supervised Training

Table 5.1: Training Duration

No	Statistics	With RBM		Without RBM
		pretraining (s)	training (s)	training (s)
1	Mean	30.8	373.7	3334.4
2	Minimum	30.1	244.07	2992.5
3	Maximum	33.8	2804.4	3714.6

Although the implementation of unsupervised learning with RBM pre-training takes some time, the total time of the NN with RBM pre-training to reach the same level of cost function is much smaller than that of NN without RBM pre-training. This result is summarized in Table 5.1. Without RBM pre-training, the NN requires 3334 seconds on average to finish the training at maximum iterations, but RBM pre-training with average duration of 30 s contributes small portion of total training but is able to speeds up the training time to only 373.7 s after reaching minimum gradient. Although our scheme finishes the training earlier, our scheme achieves a lower MSE. The trained NN is then tested to measure the compression and decompression times for a single trace as shown in Table 5.2 with 100 code layer units. It is clear that the processes are very fast as both operations take only less than a millisecond.

Table 5.2: Compression and Decompression time of East Texas dataset

No	Statistics	Compression (ms)	Decompression (ms)
1	Mean	0.21	0.22
2	Minimum	0.1	0.13
3	Maximum	0.39	0.69

In the second experiment, we investigate the minimum number of traces for training for the NN to be able to compress the upcoming data with a good quality. We start by using traces generated from first three shots as training data and increase it gradually by another three shots. Validation data is the traces from one shot after the training data. The testing data is the traces from the last five shots. Since we separate the training, validation, and testing data from its shots, there is no overlapping data. We use the same compression ratio as in the previous section and the result is summarized in Table 5.3. If we set the minimum target 30dB of PSNR, then 198 traces from the 6 first shots as training data are sufficient to achieve the minimum goal. It can be noticed that unsupervised pre-training with RBM boosts the performance of the NN as indicated by high PSNR.

Table 5.3: Experiment Result with $c_r = 10:1$

Number of Shots	Number of Traces	with RBM		without RBM	
		Training	Testing	training	Testing
3	99	41dB	28dB	30dB	19dB
6	198	47dB	35dB	32dB	25dB
9	297	46dB	39dB	36dB	30dB
12	396	45dB	40dB	41dB	32dB

As we use more traces as training data, the testing data reconstruction error becomes closer to the training data. It indicates that the NN has a better generalization performance against the new upcoming data. After using traces from the first 12 shots, we plot the PSNR of the traces from last 5 shots in Figure 5.7. As proof that the NN is now robust against the newly acquired traces, it can be noticed that the reconstruction quality does not change significantly after each shot.

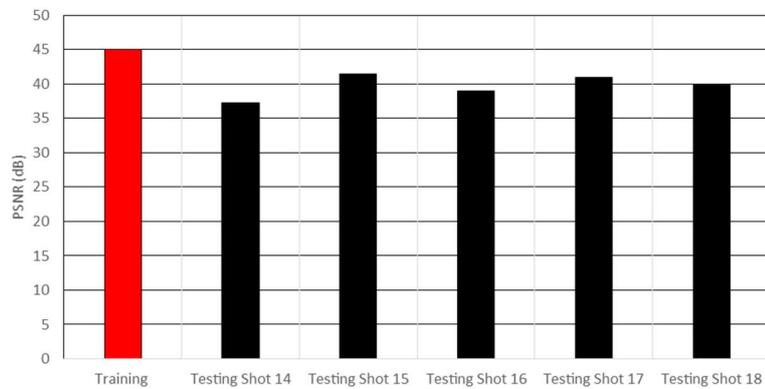


Figure 5.7: PSNR of traces from each shot of East Texas data

Figure 5.8 plots the PSNR of the reconstructed data using NN as a function of number of units in the code layer. The number of units ranging from 15 to 750 corresponds to compression ratios from 100:1 to 2:1. It can be noticed that increasing the number of units progressively increases the reconstruction quality. At high compression ratios of 100:1 and 75:1, the proposed scheme achieves 27dB and 28dB, respectively. This is a very good achievement since it is difficult to achieve more than 20dB at such high compression ratios using other techniques.

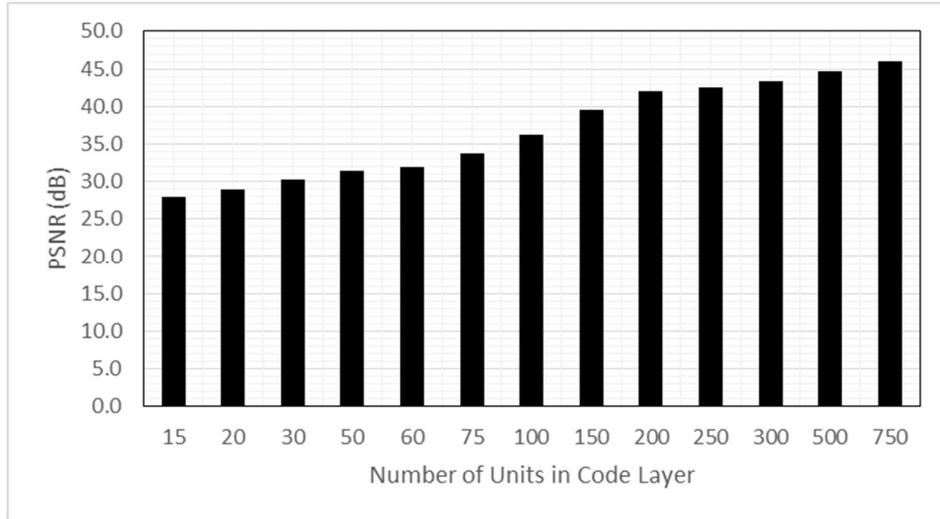


Figure 5.8: PSNR of the data as a function of Number of Units in Code Layer

B. Experiments using UTAM Avenue seismic dataset

We further evaluate our scheme on the Avenue data from *UTAM* seismic data library [123] with 100 active geophones and 100 shots. The interval between any two successive geophones is around 3 m. The shots are generated sequentially at the same locations of every other geophone, i.e., by approximately 6 m away from each other. Each of the recorded traces consists of 4000 time samples using a sampling period of 0.25 ms.

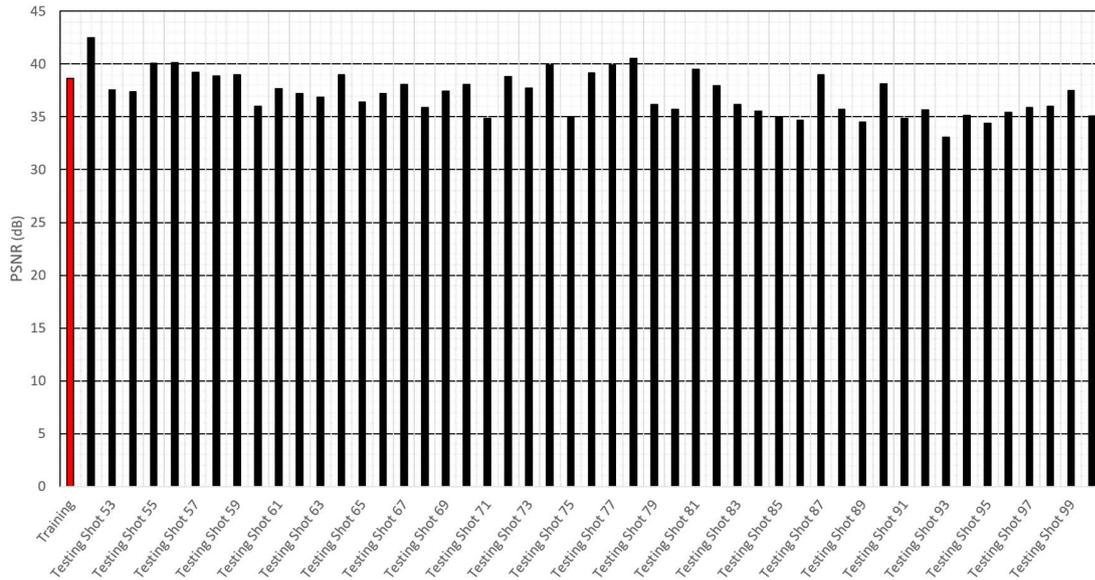


Figure 5.9: PSNR of each shot of UTAM Avenue dataset.

We use the 1st to 50th shots as training data, the 51st shot as cross-validation data, and the shots at 52, 53, ... 100 as testing data with 400 code layer units corresponding to 10:1 of compression ratio. The average PSNR of training and testing data are 38dB and 37dB, respectively. The PSNR of the training data as average and each testing shots are plotted in Figure 5.9. It can be noticed that PSNR of each shot does not change significantly by the position of the shots. Since the each trace in this dataset is almost three times longer than that of the East Texas, the compression and decompression times of this dataset obviously take longer times as summarized in Table 5.4. The results are taken from five trials.

Table 5.4: Compression and Decompression time of UTAM dataset

No	Statistics	Compression (ms)	Decompression (ms)
1	Mean	2.2	2.9
2	Minimum	1.8	1.9
3	Maximum	1.4	1.46

C. Comparison with LPC

The Linear Predictive Compression (LPC) uses an autoregressive (AR) model to represent discrete time series as

$$\hat{x}(m) = \sum_{i=1}^n a_i x(m-i) \quad (5.20)$$

where n is the horizontal length of this linear model and a_i denote the coefficients, which are obtained by minimizing the following cost function J

$$J = \sum_{m=n+1}^N \left[x[m] - \sum_{i=1}^n a_i x(m-i) \right] \quad (5.21)$$

It is expected that the corresponding modeling residual signal $e_L[k]$:

$$e_L(m) = x(m) - \hat{x}(m) \quad (5.22)$$

$$= x(m) - \sum_{i=1}^n a_i x(m-i) \quad (5.23)$$

is with a lower entropy than $x[m]$. The lossless LPC uses p coefficients a_i s, the first n time samples $x(1), x(2), \dots, x(n)$, along with losslessly decoded residual samples

$e_L(n + 1), e_L(n + 2), \dots, e(N)$, to reconstruct the original signal $x[m]$. To achieve a higher compression ratio, the residual samples have to be compressed in a lossy way. However, lossy compression of residual samples results in an accumulation of the reconstruction error. Therefore, to avoid error accumulation in practice, the compressor (or, sending node) emulates the reconstruction procedure of the decompressor (or, receiving node) to calculate the residual from the estimates of the reconstructed samples:

$$x'(m) = \sum_{i=1}^n a_i x_r(m - i) \quad (5.24)$$

where $x_r(m)$ denotes the reconstructed sample of $x(m)$. Then the residual samples are calculated as in (5.23). The quantization of residual is given as

$$Q[e'_L(m)] = e'_L(m) + q(m) \quad (5.25)$$

where $Q[\cdot]$ means quantization and $q(m)$ is quantization error. Thus the sample is reconstructed as

$$x_r(m) = \hat{x}(m) + Q[e'_L(m)] \quad (5.26)$$

$$= x(m) + q(m) \quad (5.27)$$

Although $x_r[m]$ seems not having any accumulated error, this procedure is still risky. First, coefficients x_i s are optimal for $\hat{x}[m]$ as in (5.20) instead of $\hat{x}'[m]$ as in (5.24). The predictive error is expected to be larger. Secondly, quantizer $Q[\cdot]$ is designed based on the distribution of $e_L[m]$, instead of $e'_L[m]$.

We compare our scheme with the LPC with the same experimental setups on East Texas dataset. We applied the LPC with order $p = 5, 10, 20$ and used three scalar quantizers

$L_q = 64, 128, 256$ for residual encoding. For the NN, we adjust the number of unit in the code layer N_c to match the compression ratio of the LPC with given p and L_q . The normalization is able to suppress the diverging of the LPC. The PSNR calculated from the reconstructed traces are plotted Figure 5.10(a)-(c) for 64,128, and 256 quantization levels, respectively. The corresponding results of the NN with the same compression ratio are plotted in the same figure. The PSNR of the proposed method is much higher than that of the LPC with the same compression ratio. As a sample, a reconstruction is shown in Figure 5.11.

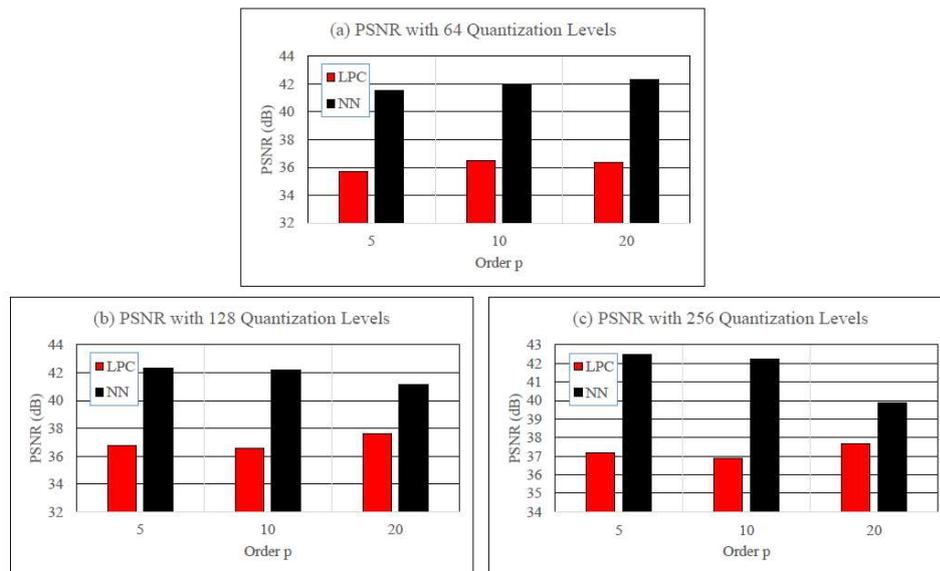


Figure 5.10: PSNR of LPC and the proposed method.

(a) 64 Quantization Levels (b) 128 Quantization Levels (c) 256 Quantization Levels

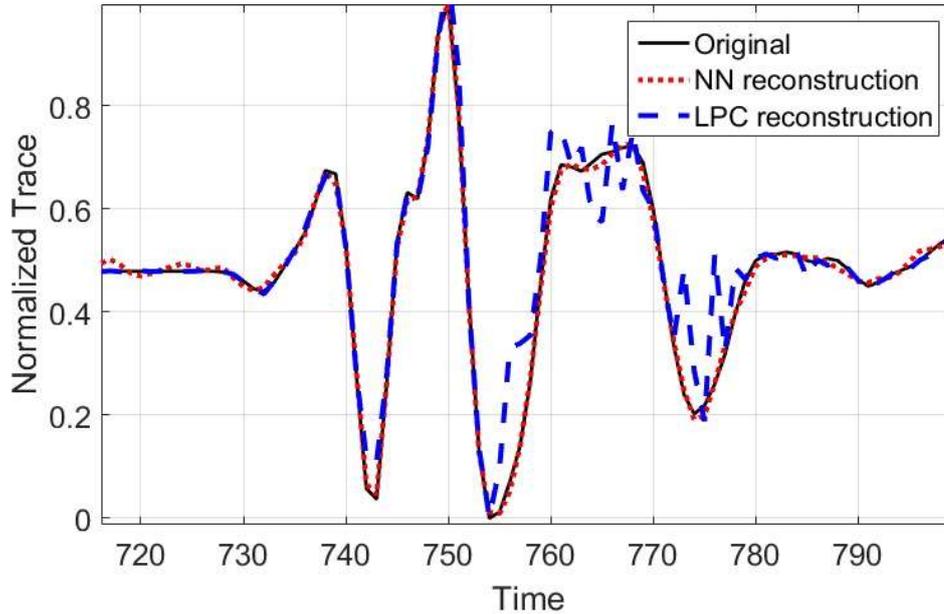


Figure 5.11: Reconstruction Sample

5.2.3 Concluding remarks for data compression using RBM

In this section, we presented a practical implementation of seismic data compression using NN. There are several advantages of using this strategy. First, we have simple NN architecture for geophone implementation. The single layer auto-encoder with linear hidden units works as a regular linear transformation. This architecture provides minimum size of weights and lightweight computation for geophone implementation. Second, RBM unsupervised learning enhanced learning rate of the supervised training. This enhancement is important for seismic acquisitions that require fast processing that can be delayed due to the NN training. The unsupervised learning speeds up the supervised training from almost one hour to only six minutes. Third, the RBM is shown to enhance the generalization performance of the NN. This enhancement allows the NN to achieve a higher reconstruction quality for both training and testing. In our

experiments using two real seismic datasets, it is also worth mentioning that we can train the NN using early acquired data to compress newly acquired data with more than 30dB of PSNR and 10:1 of compression ratio. Our method is also shown to be significantly superior to the LPC. Both compression and decompression times are very fast and take only 0.2ms in average.

5.3 Prediction based compression using deep learning

In this section, we develop a prediction based compression scheme using the DNN. In a typical prediction based method, the $(n + 1)^{th}$ sample of a signal is predicted by its n past samples. The discrepancies between the predicted and actual samples are sent or stored as residuals instead of the original sample itself. The compression is achieved because the variance of the residuals is lower than that of the original sample, and hence may be represented by fewer bits per sample. We utilize the DNN for regression to achieve more precise prediction as indicated by a lower residual variance. The residuals are then quantized and encoded using the Huffman coding. The developed scheme is empirically compared with the LPC using a real seismic dataset.

5.3.1 Methodologies

A typical prediction based compression scheme is illustrated in Figure 5.12. The scheme is implemented in two-stages. In the first stage, a portion of the seismic data samples is used for training the predictor until the goal is reached. The weights or coefficients of the trained predictor are sent to the receiver side for reconstruction. The first n samples of

each trace are also sent to the receiver as the initial data segments for prediction, where n is the order of prediction [124].

Prediction is a supervised learning problem. Given input vector $\mathbf{x} = [x_{m+1} \ x_{m+2} \ \dots \ x_{m+n}]$ and output $y = [x_{m+n+1}]$ pairs obtained from the training data set $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_k, y_k)\}$, The predictor learns the best parameters for predicting model $\hat{y} = h(\mathbf{x})$ that minimizes the loss function, i.e. $e(y, \hat{y}) = \frac{1}{2}(y - \hat{y})^2$.

In the second stage, the generated residuals are quantized using K-means algorithm. The quantized residuals are further encoded with entropy coding. The encoded residual sequences are then transmitted to the receiver side and decoded to reconstruct the data.

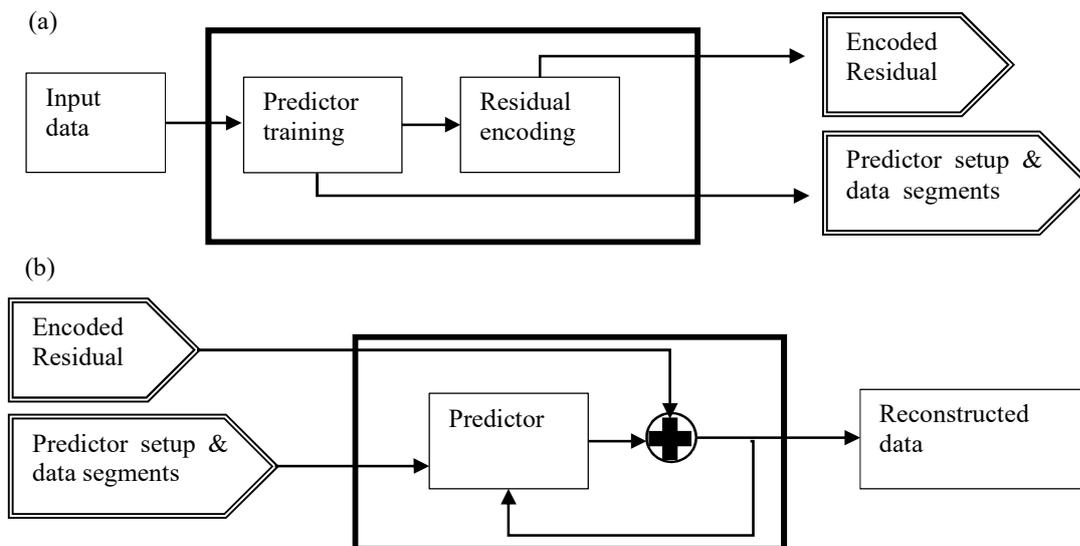


Figure 5.12 Prediction based method (a) sender side and (b) receiver side

A. Classical prediction based compression using the LPC

The LPC as discussed in page 105 uses the AR model to represent discrete time series as $\hat{x}(k) = \sum_{i=1}^p a_i x(k-i)$, where p is the order of the AR model and a_i s are the LPC coefficients, which are obtained by minimizing mean squared fitting errors. The corresponding modeling residual signal $e_L(k) = x(k) - \hat{x}(k)$ representing the predictive error has a lower entropy than $x[k]$. The lossless LPC uses p coefficients a_i s, the first p time samples $x(1), x(2), \dots, x(p)$, along with decoded residual samples $e_L(p+1), e_L(p+2), \dots, e_L(N)$, to reconstruct the original signal $x[k]$. To achieve a higher compression ratio, the residual samples have to be compressed in a lossy way. However, lossy compression of residual samples results in accumulation of the reconstruction errors. Therefore, to avoid error accumulation in practice, the compressor (or, sending node) emulates the reconstruction procedure of the decompressor (or, receiving node) to calculate the residual from the estimates of the reconstructed samples.

The quantization error during residual quantization is also expected to be larger. The predictive and quantization errors are accumulated in the reconstructed trace and therefore the decompressed signal may largely diverge. Therefore, we normalize each trace to reduce the quantization errors and to prevent the diverging reconstruction.

B. Prediction using DNN

In this section, we discuss the DNN as a time series predictor. Figure 5.13 shows the DNN for prediction which comprises an input layer with n units, where n is the order of prediction, an output layer with a single unit, and L hidden layers $H_1, H_2, \dots, \text{and } H_L$. We

used the same prediction based strategy as the LPC, where the residual is quantized. The predicted value enhanced with the quantized residual becomes the input to predict the next value. The DNN is expected to achieve more precise prediction such that the variance of the residual becomes smaller.

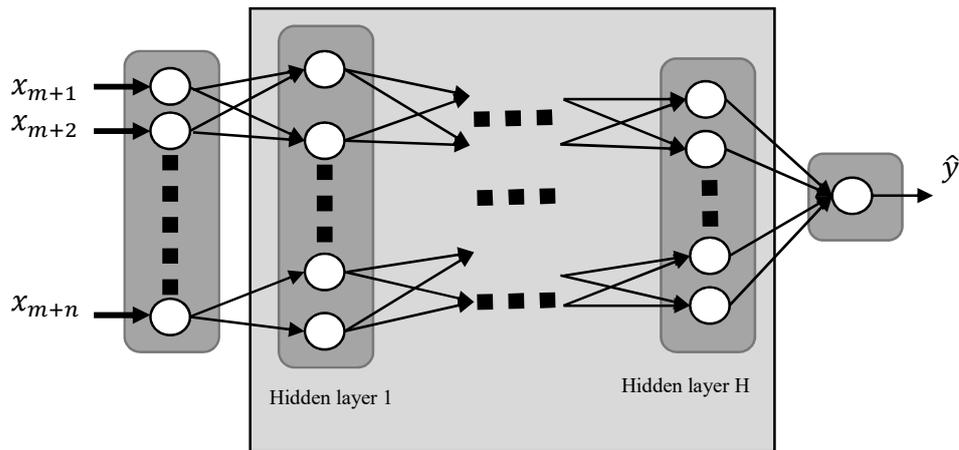


Figure 5.13 A DNN with multiple hidden layers

The DNN can model extremely complex and non-linear relationships between inputs and outputs from training data. However, a DNN is generally difficult to train since the error gradient would vanish as the number of hidden layers increases. Recent works on deep learning have shown that the DNN can be trained layer-wise to achieve good results for many machine learning tasks such as recognition, classification, and regression [109, 125]. The idea of deep learning is to learn the data structure by pre-training a neural networks with multiple hidden layers in an unsupervised way using large amounts of data. The neural networks is further fine-tuned using target data for supervised training to slightly adjust the learned features for more precise prediction [126].

Pre-training is performed by decomposing each layer as an input-output pair. Each pair is then treated as an RBM. We use RBMs with binary units that are suitable for the sigmoid activation function in the hidden layers. Similar with the RBM for linear activation function, the RBM with binary units consists of two layers namely visible \mathbf{v} and hidden \mathbf{h} (Figure 5.2). The biases \mathbf{b} and \mathbf{c} denote the visible and hidden layer biases. The visible and hidden units are connected with weights \mathbf{W} . The initial values of the weights and biases are assumed to be random with Gaussian distributions $N(0, 0.1)$.

In the positive phase, we set the training vector \mathbf{x} as the values of the visible units (\mathbf{v}^+).

The value of each hidden unit (h_j^+) is set to one with probability

$$\hat{h}_j^+ = p(h_j^+ = 1 | \mathbf{v}^+) = \sigma\left(c_j + \sum_i v_i^+ w_{ij}\right) \quad (5.28)$$

where $\sigma(x)$ is the logistic sigmoid function $f(x) = 1/(1 + \exp(-x))$. The values from the positive phase h_j^+ are used to calculate the values in the negative phase as follows

$$\hat{v}_i^- = \sigma\left(b_i + \sum_j h_j^+ w_{ij}\right) \quad (5.29)$$

$$\hat{h}_j^- = \sigma\left(c_j + \sum_i \hat{v}_i^- w_{ij}\right). \quad (5.30)$$

Finally, the updates for the weights and biases are given by

$$\Delta w_{ij} = \epsilon(\langle v_i \hat{h}_j \rangle^+ - \langle \hat{v}_i \hat{h}_j \rangle^-) \quad (5.31)$$

$$\Delta b_i = \epsilon(\langle v_i \rangle^+ - \langle \hat{v}_i \rangle^-) \quad (5.32)$$

$$\Delta c_j = \epsilon(\langle \hat{h}_j \rangle^+ - \langle \hat{h}_j \rangle^-) \quad (5.33)$$

where $\langle \cdot \rangle$ denotes expectation operator over all data and ϵ is the learning rate. Superscript $+$ and $-$ indicate the positive and negative phase, respectively. This process is repeated for a pre-determined maximum number of iterations. The same procedure is applied for the next RBM using the outputs of the previous RBM as inputs to the current one. This process continues until the last RBM. The weights and biases obtained from this pre-training process are used as initial values for a feed-forward neural networks obtained by stacking the RBM pairs in sequence. This resultant neural networks is trained using regular feed-forward back-propagation for fine tuning. The training process continues until the error between the predicted output and the actual value reaches a pre-determined value or a maximum number of iterations is reached.

C. Residual encoding and the Huffman encoding

The residuals of the prediction are then quantized and encoded into fixed-length code-words. Precise predictions yield low residual variance energy. Therefore, the residual can be quantized to a lower quantization level. This step can be improved further since the strings with different number of occurrences can be efficiently encoded using entropy based coding. The Huffman encoding [42] is used in our experiment as an entropy coding that exploits non-uniformity of symbols to achieve a shorter average code-word length [127]. This algorithm allows the symbols with higher occurrences to have shorter code-words and vice versa [21].

5.3.2 Experimental results

In this sub-section, we evaluate the performance of proposed scheme using the East Texas dataset and compare its performance with the LPC. In our experiments, we normalized each trace between 0 and 1 as follows

$$\mathbf{x}_0 = \frac{(x-x_{min})}{(x_{max}-x_{min})}, \quad (5.34)$$

where x and x_0 denote the original and normalized trace, respectively. Scalars x_{max} and x_{min} denote the maximum and minimum value in the trace. Both of these values must be sent to the receiver side to obtain the reconstructed trace in the original scale.

In our experiments, we divided our data equally into training, cross validation, and testing. Based on performance on cross validation data, we used 20 iterations for pre-training for each RBM with a unified learning rate of 0.1 for all weights and biases. Three RBMs are pre-trained to generate initial weights and biases for three hidden layers with 30-20-10 units. All hidden units used sigmoid activation units. The fine-tuning is done using the back-propagation technique with a learning rate of 0.001 and a maximum number of iterations of 100.

The compression ratio is defined as the ratio of the original data volume and the compressed data volume. In our compression scheme, a single DNN is a universal predictor for all traces since it learned the features from all the training data. On the other hand, the LPC coefficients are only optimal for each trace since the AR-model is derived for each trace individually. Therefore, the compression ratio of the LPC is defined as

$$CR_{LPC} = \frac{Nl}{(2n+2)l+(N-n)l_q} \quad (5.35)$$

where N , n , and l_q denote the trace length, initial data segment length, and code-word size, respectively. The quantization levels L_Q determine the code-word length as $l_q = \log_2 L_Q$. Typical seismic data format uses $l = 32$ bits for each sample. With the same notations, the compression ratio of the DNN based compression is simply

$$CR_{DNN} = \frac{Nl}{(n+2)l+(N-n)l_q} \quad (5.36)$$

The reconstruction quality is measured using the following signal to noise ratio (SNR)

$$SNR = 10 \log_{10} \left[\frac{\sum_{m=1}^N x^2(m)}{\sum_{m=1}^N (x(m) - \hat{x}(m))^2} \right] \quad (5.37)$$

where \mathbf{x} and $\hat{\mathbf{x}}$ denote the original and the reconstructed data.

In the first experiment, we investigate the capability of the RBM to learn the feature of the data. When the RBM of the first hidden layer are trained using full seismic traces, it learns the features of the traces. The model has learned the parts of the traces like sinusoid on specific parts of the trace as shown in Figure 5.14. We apply fast Fourier transform (FFT) to the signal and learned features. The signal energy distributions of the data and the learned features are shown in Figure 5.15. Seismic data is known to have energy concentrated at a certain frequency range [57]. It can be noticed that in Figure 5.15, the signal energy is concentrated in a major lobe and the learned features are very active in that range indicating that the RBM is able to model the data.

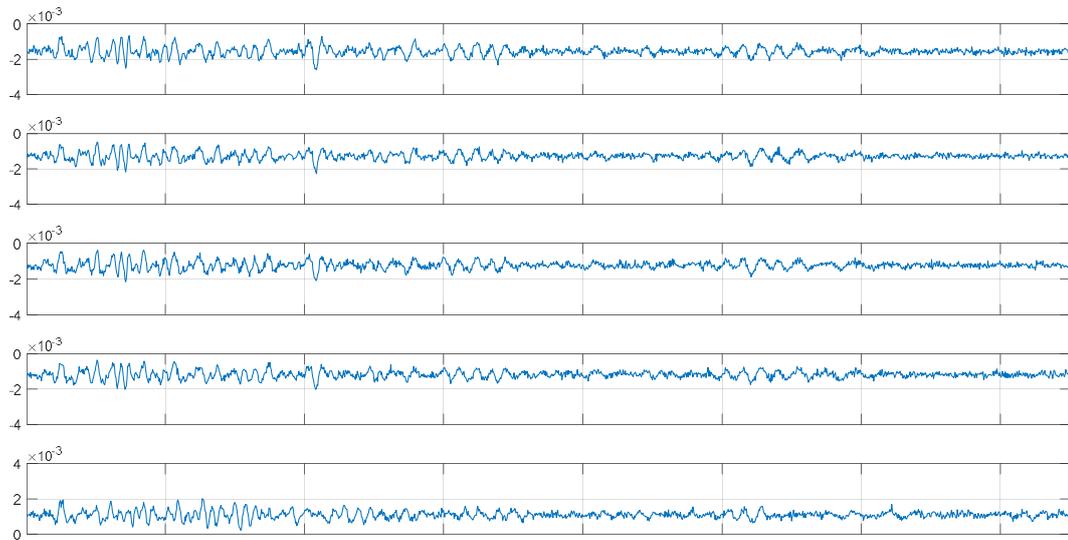


Figure 5.14 A subset of the features learned by an RBM on a full-traces. The samples are the five features with the highest L_2 norm

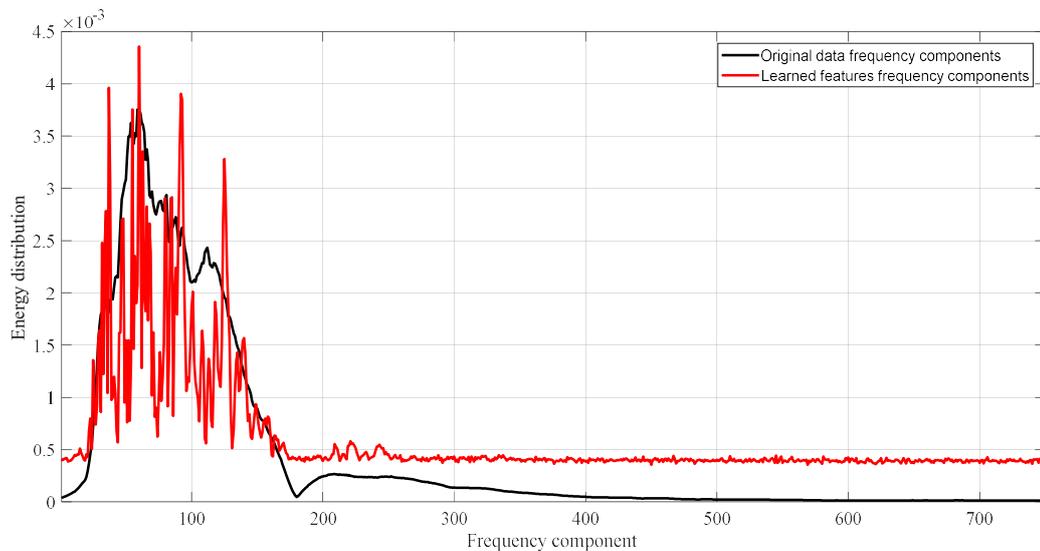


Figure 5.15 Signal energy distribution over the frequency components

We then pre-train the DNN with segments of seismic traces and fine-tune it, in a supervised manner, to predict next sample. The residuals of the prediction using DNN and LPC are shown in Figure 5.16. Note that the residuals are obtained using normalized original data, The normalized original data is more uniformly distributed than that of the

residual predictors. It can be noticed that the DNN exhibits the lowest residual variance as indicated by a compact distribution. A lower variance indicates that the residuals can be quantized with a higher quality than that of residuals with higher variance for the same quantization levels. From the information theory point of view, a lower variance yields a lower entropy that allows a higher compression ratio.

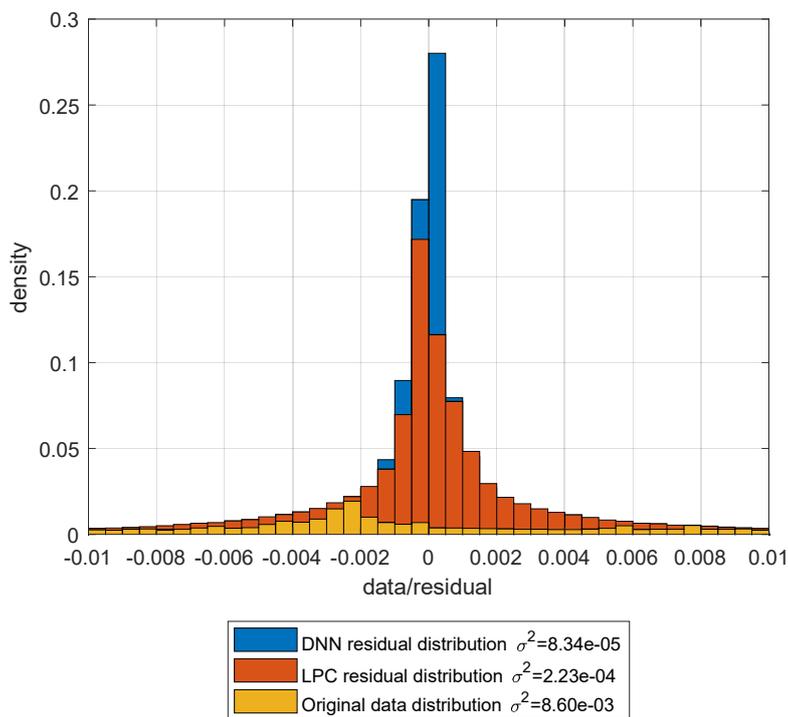


Figure 5.16 Original data and prediction residual distribution

Three codebooks of the residual encoding are constructed using the residuals of the training data where the first, second, and third codebooks implement quantization levels of $L_Q = 16, 32, 64$. Table 5.5 shows the SNR of the DNN and the LPC for data segment lengths $n = 5, 10, 15$, respectively. It can be noticed that the DNN outperforms the LPC significantly for all configurations. Increasing the quantization level obviously increases the reconstruction quality for both methods. The reconstruction quality of the DNN

averagely increases by 7dB for each quantization level increment. The LPC suffers from diverging when the quantization levels are not big enough. Both techniques do not show significant improvements when the initial segment lengths are increased.

Table 5.5 SNR of the LPC and the DNN

N	LPC (dB)			DNN (dB)		
	$L_Q = 16$	$L_Q = 32$	$L_Q = 64$	$L_Q = 16$	$L_Q = 32$	$L_Q = 64$
5	23.1*	28.96	30.27	43.02	52.15	58.31
10	26.56*	27.64	29.55	42.07	51.45	57.41
15	25.86	28.69	29.88	42.81	52.81	58.82

Prediction based compressions heavily depend on the residuals encoding. The simplest way is to apply uniform encoding scheme or fixed length code-words to all quantized residuals. Such residual encoding however contributes the major portion of the compressed data overhead. Therefore, we utilize the Huffman algorithm to the quantized residuals to achieve higher compression ratios. Table 5.6 shows the compression ratios of uniform and the Huffman encoding. It can be noticed that the Huffman encoding improves the compression ratios. A reconstruction sample is shown in Figure 5.17. The sample shows minor differences between original and reconstruction.

Table 5.6 Compression ratio of the DNN

CR	$L_Q = 16$			$L_Q = 32$			$L_Q = 64$		
	$n = 5$	$n = 10$	$n = 15$	$n = 5$	$n = 10$	$n = 15$	$n = 5$	$n = 10$	$n = 15$
Uniform	7.8	7.6	7.4	6.3	6.2	6.07	5.3	5.2	5.1
Huffman	14.4	13.3	12.2	11.04	10.7	9.6	8.75	8.7	7.9

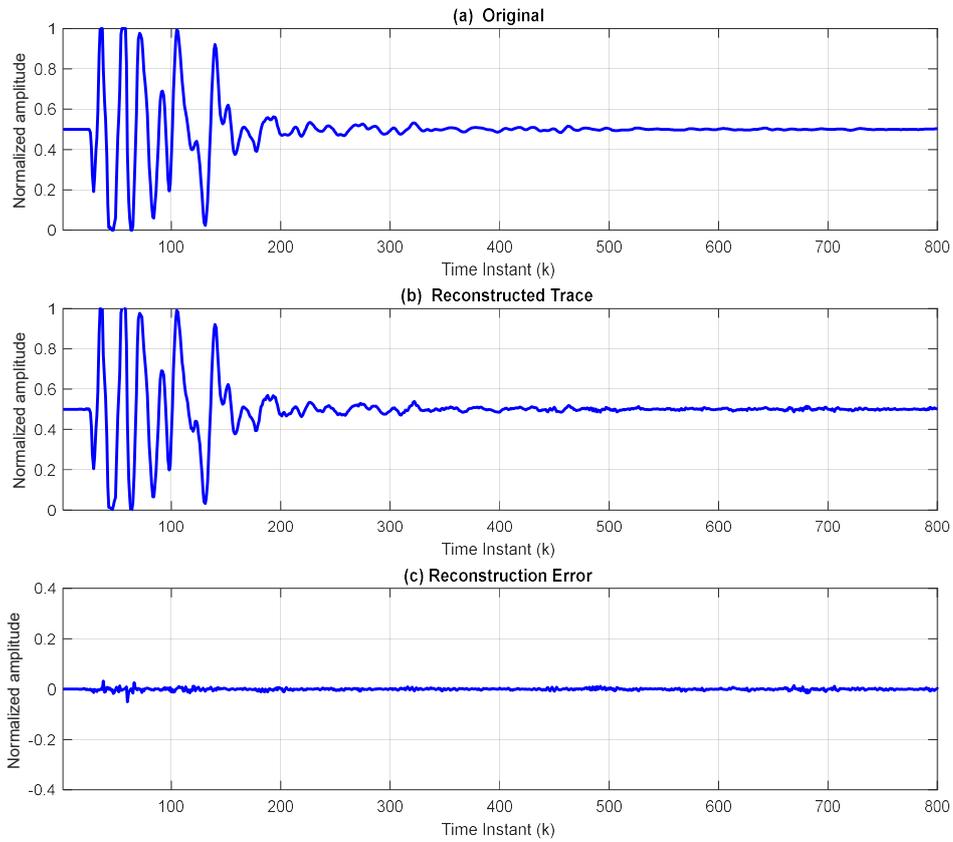


Figure 5.17 Reconstruction Sample (a) original (b) reconstructed using DNN (c) reconstruction error

5.3.3 Concluding remarks on the prediction based compression

In this section, we presented a prediction based seismic data compression scheme using deep neural networks (DNN). First, a DNN with multiple hidden layers was pre-trained

using restricted Boltzmann machines (RBMs) to obtain near optimal initial weights. The DNN was then fine-tuned in a supervised fashion to achieve a higher prediction precision. The residual between actual and predicted samples were quantized to achieve higher compression ratios. Our experiments with a real data set showed that the DNN significantly outperformed the LPC in term of reconstruction quality. The quantized residuals were further encoded using the Huffman coding. The DNN with the encoded residuals achieved more than 40dB of SNR with a compression ratio of more than 10:1 without any significant difference in the reconstructed data.

5.4 Data Compression using Stacked Auto-Encoder with Extreme Learning Machine

In this section, we introduce the Stacked Auto-Encoder ELM (SAE-ELM) with fast learning for seismic data compression. We propose a model with deep asymmetric auto-encoder architecture where the encoder and decoder hidden layers use different types of activation functions. The training data is fed to the encoder part to obtain output values of each hidden layer sequentially. The decoder weights are then calculated using ELM given the hidden layer values from the encoder. The performance of the proposed model is compared with the two existing SAE learning techniques namely SAE with Multiple Back-Propagation (SAE-MBP) [128] and Stacked RBM with Single Back-Propagation (SRBM-SBP) [109]. For all learning models, identical NN configurations are used for comparisons. The performances are evaluated based on Normalized Mean Squared Error (NMSE).

5.4.1 Stacked Auto Encoder and Extreme Learning Machine

In this sub-section, first, we discuss the principle of data compression using SAE. Second, the basic idea of the ELM is presented. Finally, the extension of the ELM for auto-encoder is discussed.

A. Auto encoder for data compression

The SAE is a feed-forward neural network that can be used for supervised learning. It consists of an input layer, hidden layers, and an output layer as depicted in Figure 5.18. The outputs of each layer become inputs of the next layer until reaching the output layer [115]. In the center of the hidden layers, a bottleneck or code layer is usually characterized by a layer with the smallest number of units. SAE is designed to capture the features from the input data. It learns data representations layer-by-layer sequentially. The first layer detects the first order features from the input data, the second layer detects the second order features from the first order features and so on. SAE can be used for data compression as it is able to reduce the input data dimensionality. The encoder side performs the data compression with specific compression ratios determined by the number of units in the code layer. The compressed data representation in the code layer is then reconstructed at the decoder side.

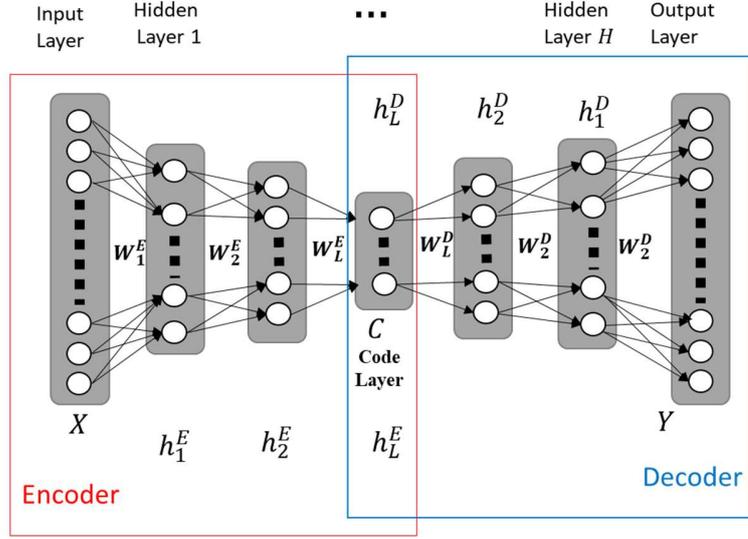


Figure 5.18 A Stacked Auto Encoder

Consider an SAE with L layers for encoding and L layers for decoding, and input signal \mathbf{x} with n samples such that $\mathbf{x} = [x(1), x(2), \dots, x(n)]^T$. SAE attempts to reconstruct \mathbf{x} at the output layer using two operations; compression and decompression, where the former can be performed in the encoder side, and the latter can be performed in the decoder side [115]. First, the encoder gradually transforms \mathbf{x} to a compressed representation \mathbf{z} , at the code layer, such that $\mathbf{z} = [z(1), z(2), \dots, z(m)]^T$ where $m \ll n$. This operation yields a compression ratio of $\frac{m}{n}$. Due to the encoding operation, an intermediate signal \mathbf{z}_l is produced at each layer l by the following transformation:

$$\mathbf{z}_l = f(\mathbf{W}_l \mathbf{z}_{l-1} + \mathbf{b}_l), \quad (5.38)$$

where f denotes the activation function, $l = [1, \dots, L]$, $\mathbf{z}_0 = \mathbf{x}$, \mathbf{z}_l has m_l samples such that: $m = m_L, m_0 = n$, $\mathbf{z}_L = \mathbf{z}$, \mathbf{W}_l is a $m_l \times m_{l-1}$ weight matrix and \mathbf{b}_l is a $m_l \times 1$ bias vector. After that, the decoder gradually transforms back \mathbf{z} to produce a reconstruction $\hat{\mathbf{x}}$.

Due to the decoding operation, an intermediate representation $\hat{\mathbf{x}}_l$ is produced at each layer l by the following transformation:

$$\hat{\mathbf{x}}_l = f(\mathbf{W}_{l+1}^D \hat{\mathbf{x}}_{l+1} + \mathbf{b}_{l+1}^D), \quad (5.39)$$

where $l = [L - 1, \dots, 0]$, $\hat{\mathbf{x}}_L = \mathbf{z}$, $\hat{\mathbf{x}}_l$ has n_l samples such that: $m = n_L, n_l = n, \hat{\mathbf{x}}_0 = \hat{\mathbf{x}}$, \mathbf{W}_l^D is a $n_l \times n_{l-1}$ weight matrix and \mathbf{b}_l^D is a $n_l \times 1$ bias vector.

A single hidden layer of auto encoder is not sufficient to model the structure in a set of data [109]. Therefore, an SAE with multiple hidden layers ($H \geq 2$) is typically used to perform seismic data compression. However, it is difficult to train SAEs with multiple hidden layers. The error gradients from the output layer gradually vanish while they are back-propagated layer by layer to the input layer, making the training extremely lengthy [117]. There are two major approaches to deal with this problem, namely SAE-MBP and SRBM-SBP. The SAE-MBP uses BP to train AE applied multiple times i.e. once for each stacking and final BP after stacking for fine-tuning [128]. The SRBM-SBP uses stacked RBM (SRBM) in which RBM is applied multiple times for each stacking and BP is applied only once at the end of last stacking for fine tuning [109].

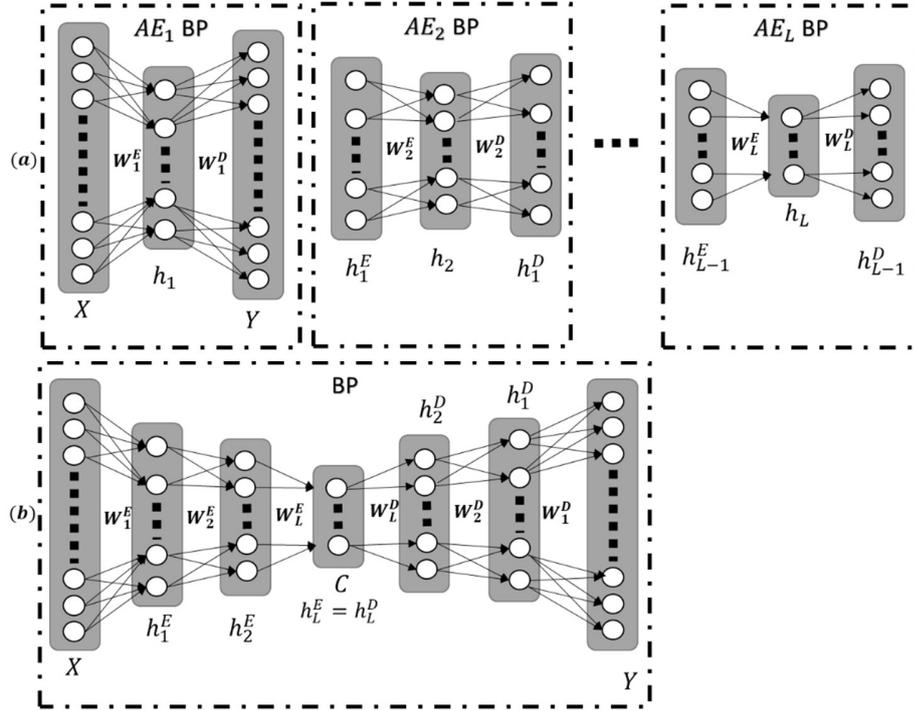


Figure 5.19 SAE-MBP with (a) multiple BPs for pre-training (b) final BP for fine-tuning

B. Stacked Auto Encoder with Multiple Back Propagation

The SAE-MBP approach is proposed in [128]. In this approach, the training is implemented in stages. In the first stage, the AE_1 is constructed with three layers, namely, input layer x , hidden layer h_1 , and output layer y as depicted in Figure 5.19a. Using training data as inputs, the weights W_1^E and W_1^D and biases are trained using back-propagation (BP) or gradient descent (GD) based optimization.

In the second stage as shown in Figure 5.19a, a separate one-hidden-layer network AE_2 consisting of input layer h_1^E , a hidden layer h_2 , and output layer h_1^D is created. Using the inputs $x_1(1), x_1(2) \dots x_1(m_1)$, where $x_1 = z_1$ is the hidden layer outputs of h_1 , the weights of W_2^E and W_2^D can be trained by using BP and GD. Hidden layer h_2 is then

inserted into the AE_1 . This procedure is repeated sequentially for all hidden layers h_1, h_2, \dots, h_L to create a single SAE as illustrated in Figure 5.19b.

After the training of all separate AE networks are completed, all the weights of the SAE are trained together using BP or GD to reach a better optimum. This kind of SAE is also referred to as the stacked auto-associators.

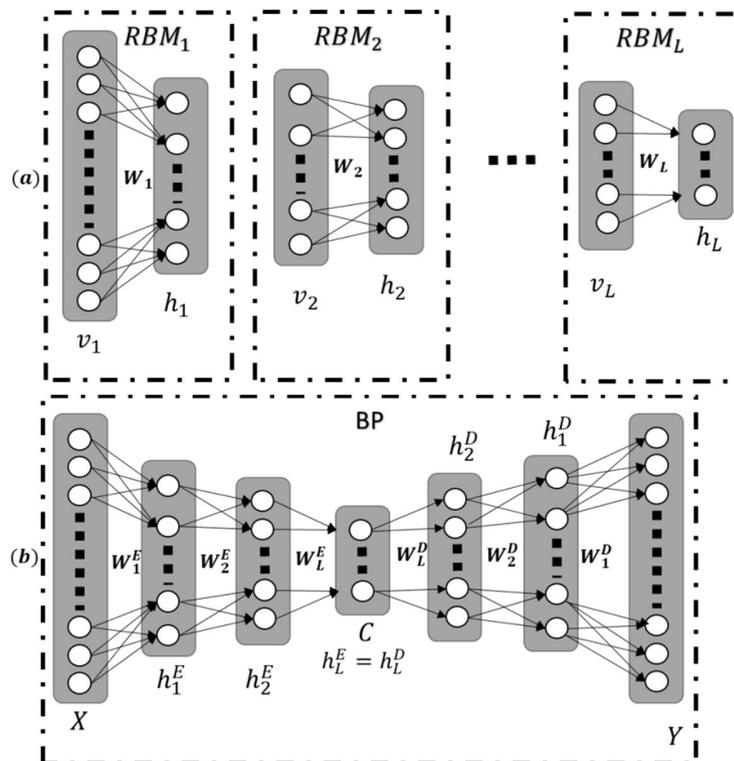


Figure 5.20 SRBM-SBP with (a) multiple RBMs for pre-training (b) stacking and a final BP for fine-tuning

C. Stacked RBM with Single Back Propagation

The second approach for training the SAE was proposed by Hinton et.al. in [109]. The unsupervised training with RBMs is used to obtain all initial weights of the SAE that can be, further, fine-tuned using BP and GD based optimizations.

Figure 5.20 shows the pre-training of a stack of RBMs and a SAE created from the pre-training and stacking of RBMs. The RBM learning can be considered as a pre-training that achieves a good region of the parameter space for the SAE. The initial weights of the SAE are set as follows:

$$\mathbf{W}_1^E = \mathbf{W}_{\text{RBM1}} \dots \mathbf{W}_L^E = \mathbf{W}_{\text{RBM,L}}, \quad \mathbf{W}_1^D = \mathbf{W}_{\text{RBM1}}^T \dots \mathbf{W}_L^D = \mathbf{W}_{\text{RBM,L}}^T \quad (5.40)$$

$$\mathbf{b}_{\text{RBM1}}^E = \mathbf{c}_1 \dots \mathbf{b}_{\text{RBM,L}}^E = \mathbf{c}_L, \quad \mathbf{b}_1^D = \mathbf{b}_{\text{RBM,1}} \dots \mathbf{b}_L^D = \mathbf{b}_{\text{RBM,L}} \quad (5.41)$$

After pre-training, the RBM is considered to be ‘unrolled’ to construct an SAE network where the stochastic activities of the networks are replaced with deterministic training using BP or GD.

D. Extreme learning machine

The ELM proposed by Huang et.al. [129] for SLFN shows that the hidden layer with M nodes can be randomly generated to transform the input data to M dimensional ELM random feature space and the network output is given by the following equation:

$$y(\mathbf{x}) = \sum_{i=1}^M \beta_i z_i(\mathbf{x}) = \mathbf{z}(\mathbf{x})^T \boldsymbol{\beta} \quad (5.42)$$

where $\boldsymbol{\beta} = [\beta_1, \dots, \beta_M]^T$ is the output weight matrix between the hidden nodes and the output nodes. $\mathbf{z}(\mathbf{x}) = [f_1(\mathbf{x}), \dots, f_M(\mathbf{x})]$ are the hidden node outputs (random hidden features) for the input \mathbf{x} and $f_i(\mathbf{x})$ is the output of the i -th hidden node. Given K training samples $\{(\mathbf{x}_i, y_i)\}_{i=1}^K$, the ELM is to resolve the following learning problems:

$$\mathbf{Z}\boldsymbol{\beta} = \mathbf{y} \quad (5.43)$$

where $\mathbf{y} = [y_1, \dots, y_K]^T$ denote the target labels and $\mathbf{Z} = [\mathbf{z}^T(\mathbf{x}_1), \dots, \mathbf{z}^T(\mathbf{x}_K)]^T$ denote the hidden layer outputs. The output weights $\boldsymbol{\beta}$ can be calculated as follows:

$$\boldsymbol{\beta} = \mathbf{Z}^\dagger \mathbf{y} \quad (5.44)$$

where \mathbf{Z}^\dagger denote the Moore-Penrose generalized inverse of matrix \mathbf{Z} .

To achieve a better generalization performance and to make the solution more robust, one can add a regularization term as shown in [130]:

$$\boldsymbol{\beta} = \left(\frac{\mathbf{I}}{C} + \mathbf{Z}^T \mathbf{Z} \right)^{-1} \mathbf{Z}^T \mathbf{y}. \quad (5.45)$$

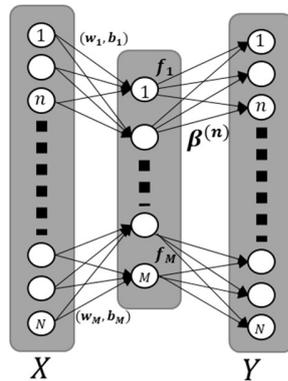


Figure 5.21 an auto-encoder ELM

E. Auto-encoder using Extreme Learning Machine AE-ELM

The AE-ELM was developed by Kasun et.al [131] as an extension to the regular ELM for feature representation learning. To construct an AE-ELM, the ELM is modified as follows to perform supervised learning where input data is used as target output data $\mathbf{y} =$

\mathbf{x} . Therefore, the number of units in the output layer is the same as the input layer $\mathbf{x} = \mathbf{y} \in \mathbb{R}^N$.

Given M units in the hidden layer, the main objective of AE-ELM is to represent the input features meaningfully in three different representations. First, a compressed representation, where $M < N$, transforms features from a higher dimensional input data space to a lower dimensional feature space. Second, a sparse representation, where $M > N$, represents features from a lower dimensional input data space to a higher dimensional feature space. The third representation is the equal dimension representation where $M = N$ that represents features from an input data space to feature space with equal dimensionality [131].

By extending the ELM as a universal approximator [132], the network structure of the AE-ELM for feature representations is shown in Figure 5.21. The random weights and biases of the hidden nodes are chosen to be orthogonal. Even when the random weights and biases are not orthogonal, the AE-ELM is capable of learning a useful feature representation [131]. However by choosing the random weights and random biases to be orthogonal, higher performance tends to be achieved.

In the AE-ELM, the orthogonal random weights and biases of the hidden nodes transform the input data to a different or equal dimension space as shown by Johnson-Lindenstrauss Lemma [133] and calculated by the following equation:

$$\mathbf{z} = \mathbf{f}(\mathbf{w}\mathbf{x} + \mathbf{b}) \tag{5.46}$$

$$\mathbf{W}^T \mathbf{W} = \mathbf{I}, \mathbf{b}^T \mathbf{b} = 1$$

where $\mathbf{W} = [\mathbf{w}_1, \dots, \mathbf{w}_M]$ are the orthogonal random weight and $\mathbf{b} = [\mathbf{b}_1, \dots, \mathbf{b}_M]$ are the orthogonal random bias between the input nodes and hidden nodes.

The output weights $\boldsymbol{\beta}$ of the AE-ELM are responsible of learning the transformation from the feature space to input data. The output weights $\boldsymbol{\beta}$ are then obtained by the following equation:

$$\boldsymbol{\beta} = \left(\frac{\mathbf{I}}{C} + \mathbf{Z}^T \mathbf{Z} \right)^{-1} \mathbf{Z}^T \mathbf{X} \quad (5.47)$$

where $\mathbf{Z} = [\mathbf{z}_1, \dots, \mathbf{z}_N]$ are the hidden layer outputs of AE-ELM and $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_N]$ are both input data and output data of the AE-ELM at the same time.

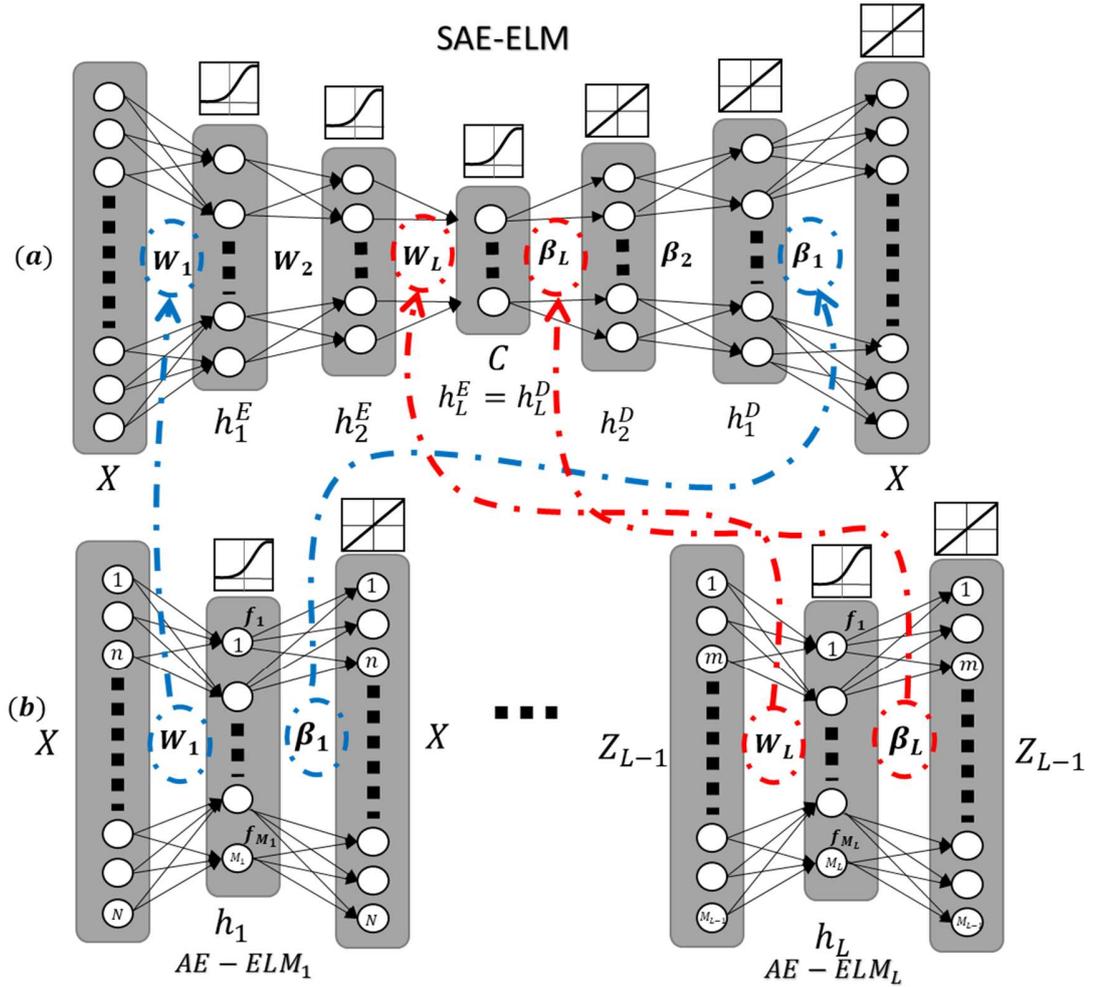


Figure 5.22 SAE-ELM with (a) sequential trainings from $AE-ELM_1$ to $AE-ELM_L$ (b) stacking the AE-ELMs arranging weights $W_1 \dots W_L$ and $\beta_L \dots \beta_1$

5.4.2 Proposed Model and Learning Method

In this sub-section, we introduce an SAE model as depicted by Figure 5.22a with asymmetric encoding-decoding operation for seismic data compression. The asymmetric operation means that all layers in the encoder part of the model use non-linear activation function whereas all layers in the decoder counterpart use linear function without any biases. Given an input vector \mathbf{x} and the reconstructed vector $\hat{\mathbf{x}}$, and SAE model with

random encoder weights and biases $\{\mathbf{W}_1, \dots, \mathbf{W}_L, \mathbf{b}_1, \dots, \mathbf{b}_L\}$, the main goal of the proposed learning method is to find the decoder weights $\{\boldsymbol{\beta}_1, \dots, \boldsymbol{\beta}_L\}$ that minimize the discrepancies between \mathbf{x} and $\hat{\mathbf{x}}$. The proposed asymmetric model allows us to obtain the analytical solution of the decoder weights using AE-ELM. The proposed learning scheme is expected to be faster than aforementioned SAE learning algorithms due to two factors. First, we only need to optimize the decoder weights and, secondly, each weight can be solved analytically without any iteration.

Given an input training data $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_K\}$ with K samples, the output of the l th encoder hidden layer $\mathbf{Z}_l = \{\mathbf{z}_{l,1}, \dots, \mathbf{z}_{l,K}\}$ is given by

$$\mathbf{z}_l = f(\mathbf{W}_l \mathbf{z}_{l-1} + \mathbf{b}_l), \quad l = 1 \dots L \quad (5.48)$$

where $\mathbf{z}_0 = \mathbf{x}$ is the input data for the first hidden layer and \mathbf{z}_{l-1} denotes the output of the preceding hidden layer. This operation is sequentially performed for each layer in the encoder until the code layer. The output of the code layer $\mathbf{Z} = \mathbf{Z}_L$ is then sequentially decoded by each layer in the decoder part. The intermediate decoded vector $\hat{\mathbf{x}}_l$ is the output vector of the l th hidden layer at the decoder part. The decoding operation is given by

$$\hat{\mathbf{x}}_l = \boldsymbol{\beta}_{l+1} \hat{\mathbf{x}}_{l+1}, \quad l = L - 1 \dots 0 \quad (5.49)$$

where $\hat{\mathbf{x}}_L = \mathbf{Z}_L$ is the output of the code layer and $\hat{\mathbf{x}}_l$ denotes the output of the l th hidden layer at the decoder. The output of the final layer $\hat{\mathbf{X}} = \hat{\mathbf{X}}_0$ is the reconstructed data.

Suppose that the weights and biases in the encoder part satisfy the orthonormal condition in (5.46) and each output of the l th encoder hidden layer \mathbf{Z}_l is perfectly reconstructed by

its intermediate decoded data $\hat{\mathbf{X}}_l$. Then, each encoder-decoder parameter set $\{\mathbf{W}_l, \mathbf{b}_l, \boldsymbol{\beta}_l\}$ forms an AE-ELM with input data \mathbf{Z}_l and output target $\hat{\mathbf{X}}_l \approx \mathbf{Z}_l$ as depicted by Figure 5.22b.

Therefore, given an input data \mathbf{X} and the output at the l th encoder hidden layer, $\mathbf{Z}_l, l = 1 \dots L$, the weight $\boldsymbol{\beta}_l$ can be obtained as follows

$$\boldsymbol{\beta}_l = \left(\frac{\mathbf{I}}{C} + \mathbf{Z}_l' \mathbf{Z}_l \right)^{-1} \mathbf{Z}_l' \hat{\mathbf{X}}_{l-1} \quad (5.50)$$

Since $\hat{\mathbf{X}}_{l-1}$ is expected to reconstruct \mathbf{Z}_{l-1} then, equation (5.50) can be expressed as follows

$$\boldsymbol{\beta}_l = \left(\frac{\mathbf{I}}{C} + \mathbf{Z}_l' \mathbf{Z}_l \right)^{-1} \mathbf{Z}_l' \mathbf{Z}_{l-1}. \quad (5.51)$$

Note that $\mathbf{Z}_l, l = 1 \dots L$, must be calculated sequentially. However, once all \mathbf{Z}_l are calculated, each $\boldsymbol{\beta}_l$ can be calculated independently from other $\boldsymbol{\beta}$.

The proposed scheme above is referred to as the Stacked Auto-Encoder Extreme Learning Machine (SAE-ELM). The learning procedure is summarized as follows.

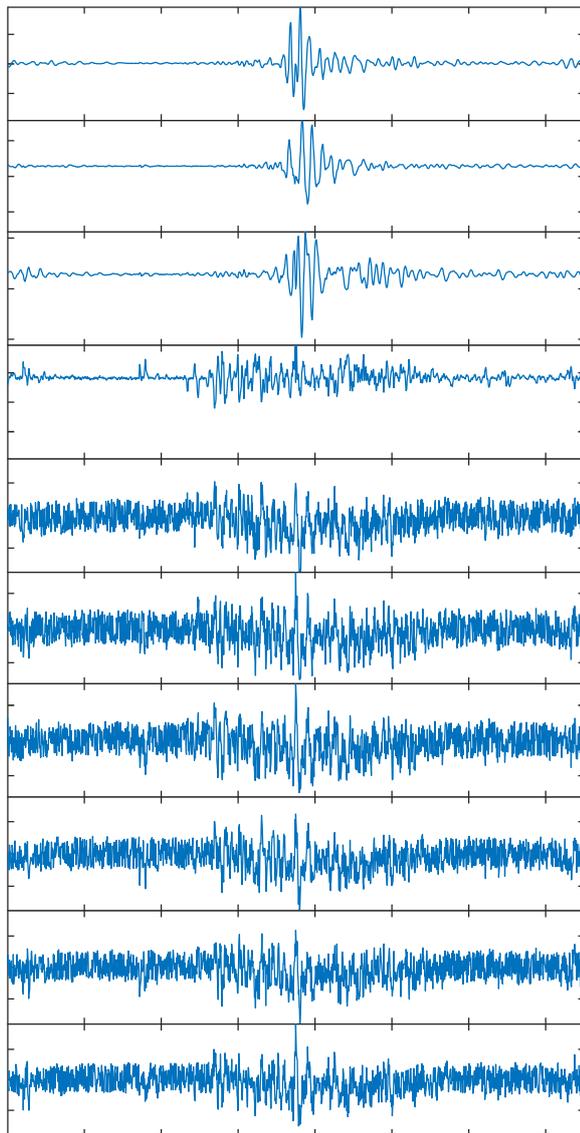
The SAE-ELM algorithm

Input: training samples \mathbf{X} , encoder and decoder hidden layer number L , activation function $f(x)$, regularization parameter C .

- 1: Initialize an SAE with orthonormal random hidden unit parameters $(\mathbf{W}_i, \mathbf{b}_i), i = 1, 2, \dots, L$.

A. Representation learning

To test the feature representation learning, we follow the setups in [131] by creating a mini dataset with 100 traces. Next, the mini dataset is used to train an AE-ELM (network configuration 1501-20-1501) and the contents of the output weights β are compared with BP and RBM with BP fine-tuning.



(a)

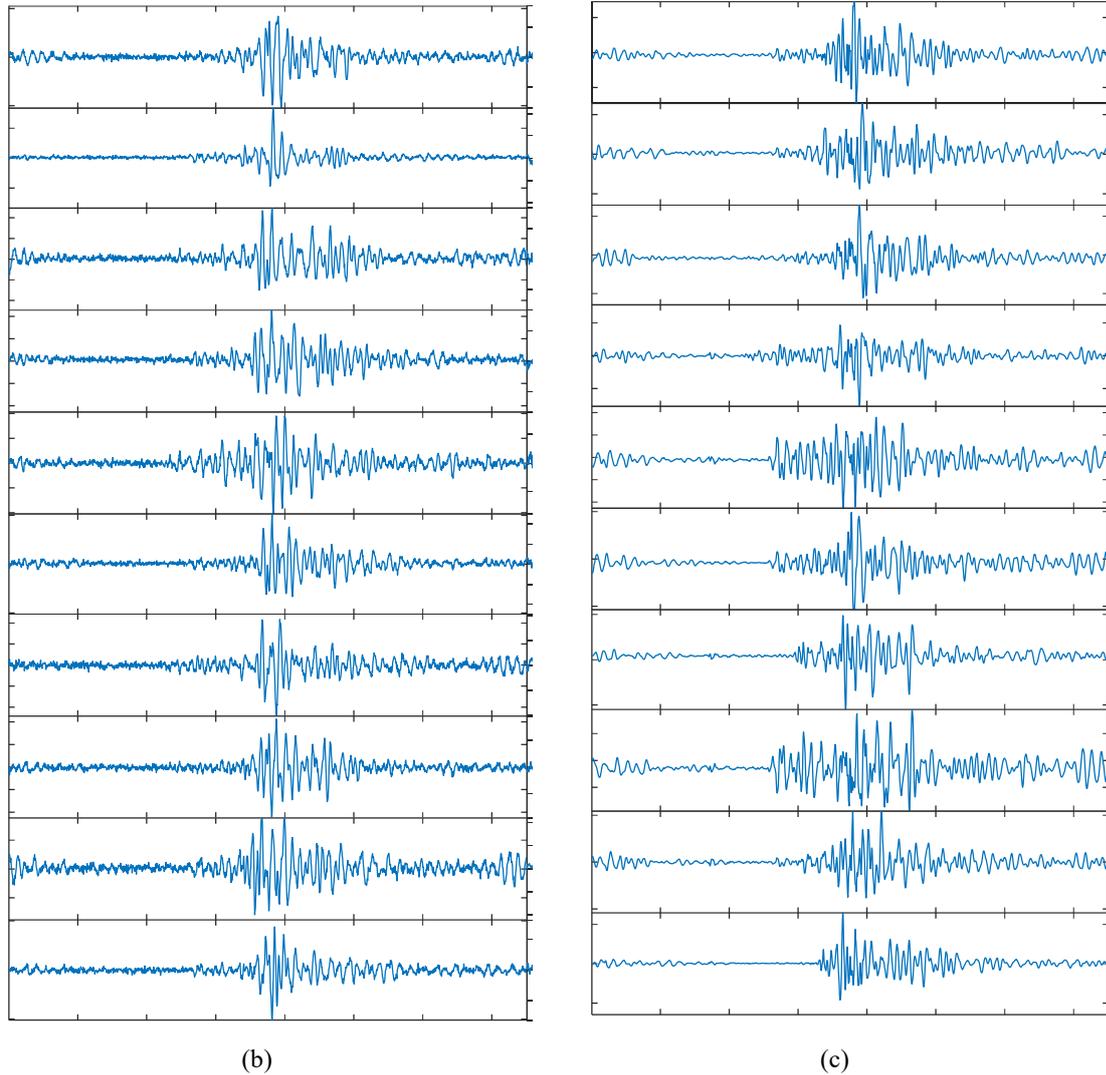


Figure 5.24. Weights of (a) BP (b) RBM-BP (b) (c) ELM

The weights learned by the BP and RBM are shown in Figure 5.24(a) and (b), respectively. In Figure 5.24(c), we plot ELM output weights β instead of the input layer weights since the learning procedure is performed at the output layer weights and the input weights are regular random orthonormal transform basis. From top to down, all features are sorted based on their signal energy. It can be seen that the output weights β represent the features better than that of the RBM or BP based learning. The features learned using ELM show less noisy plots than that of the RBM or BP. The weights from

our proposed method are more adapted to the dataset, therefore, it is expected to achieve a lower reconstruction error than that of other methods.

B. Optimal architectures

To find the optimal architecture for each compression ratio, we utilize 80% of traces for training and 20% for testing. We use the NMSE to measure the signal distortion. The NMSE can be computed as

$$\text{NMSE} = \frac{\sum_{i=1}^K \|\mathbf{x}_i - \hat{\mathbf{x}}_i\|^2}{\sum_{i=1}^K \|\mathbf{x}_i\|^2}, \quad (5.52)$$

where K is the number of traces in the dataset. \mathbf{x} and $\hat{\mathbf{x}}$ denote the input vector and its corresponding reconstruction.

Table 5.7 Performance of SAE-ELM with different architectures and compression ratios

Architectures	Time (s)	NMSE (10^{-3})	CR
1501-15	0.0586	10.01	100:1
1501-1000-15	1.303	10.91	
1501-4000-15	6.555	10.43	
1501-2500-1500-15	7.755	10.37	
1501-25	0.104	8.27	60:1
1501-1000-25	1.299	8.31	
1501-4000-25	7.024	7.89	
1501-2500-1500-25	7.909	7.998	
1501-30	0.0987	7.602	50:1
1501-1000-30	1.280	7.11	

1501-4000-30	6.445	7.202	
1501-2500-1500-30	7.994	7.534	
1501-50	0.07	5.017	30:1
1501-1000-50	1.308	4.904	
1501-4000-50	6.147	5.020	
1501-2500-1500-50	8.005	4.955	
1501-75	0.0919	3.5	
1501-1000-75	1.3193	3.63	
1501-4000-75	7.606	3.444	
1501-2500-1500-75	8.049	3.357	
1501-100	0.1133	2.515	15:1
1501-500-100	1.2756	2.599	
1501-4000-100	7.1938	2.449	
1501-2500-1500-100	8.205	2.323	
1501-150	0.1427	1.512	10:1
1501-1000-150	1.13	1.506	
1501-4000-150	5.94	1.374	
1501-2500-1500-150	8.235	1.282	
1501-300	0.1901	0.495	5:1
1501-1000-300	1.3567	0.337	
1501-4000-300	6.889	0.302	
1501-2500-1500-300	8.6607	0.393	

1501-500	0.3954	0.175	3:1
1501-1000-500	1.3579	0.13	
1501-4000-500	6.92	0.125	
1501-2500-1500-500	8.8095	0.201	
1501-750	0.830	0.072	2:1
1501-1000-750	1.9921	0.077	
1501-4000-750	7.9148	0.082	
1501-2500-1500-750	9.1623	0.143	

Table 5.7 summarizes the experiment results to identify the optimum layer architecture for different compression ratios. The first column indicates the input and the encoder hidden layer sizes. For example, the layer size $m_1-m_2-m_3-m$ indicates an auto-encoder with the architecture of $m_1-m_2-m_3-m-m_3-m_2-m_1$. The compression ratio is determined by the number of units at the code layer m . For example, architectures with $m = 150$ and $m = 75$ yield 10:1 and 20:1 compression ratios, respectively.

From Table 5.7, one can notice that the training duration depends heavily on the number of units in the hidden layer. Obviously, architectures with many hidden layers and large number of units tend to require more time to be trained. However, for almost the same number of units, architectures of 1501-2500-1500- m -1500-2500-1501, on average, take longer training durations than that of the architectures of 1501-4000- m -4000-1501. It can also be noticed that increasing number of units at code layer does not increase the training duration significantly.

It can be noticed in Table 5.7, that the best results for each compression ratio are achieved by different architectures (as indicated by bold faces). The lowest NMSE for 100:1 and 2:1 of compression ratios are achieved by architectures with only single hidden layer but for the remaining compression ratios, the lowest NMSE can be achieved by architectures with multiple hidden layers. The architectures with $m = 30$ and $m = 50$ corresponding to compression ratio of 50:1 and 30:1, respectively, exhibit the best performance when the first hidden layer is with $m_2 = 1000$ units and do not show any improvement despite the first hidden layer uses $m_2 = 4000$ units. To the contrary, the best performance for architectures with $m = 25, 300$, and 500 is achieved when $m_2 = 4000$ units. It can be seen that for compression ratios of 20:1, 15:1, and 10:1, the architectures with five hidden layers (1501-2500-1500- m -1500-2500-1501) achieve lower NMSE than that of three hidden layers (1501-4000- m -4000-1501) despite having almost the same number of units in hidden layers (> 8000). A sample of the original and the reconstructed seismic trace using SAE-ELM with 10:1 of compression ratio and 1.28×10^{-3} of NMSE is shown in Figure 5.25.

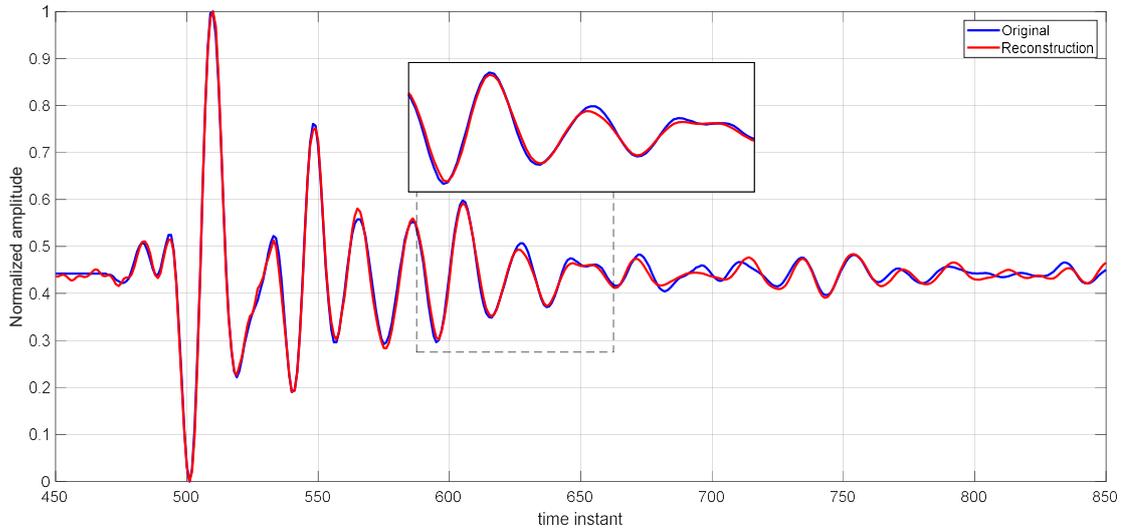


Figure 5.25. A reconstruction sample with 10:1 compression ratio

C. Comparison with other methods

In this section, we compare the proposed method with existing SAE learning methods like SRBM-SBP and SAE-MBP for data compression. In addition to that, we also use the Discrete Cosine Transform (DCT) as a comparative method for seismic data compression with linear transformations. In [37], the DCT was shown to achieve the closest performance to the optimal linear transform for seismic data.

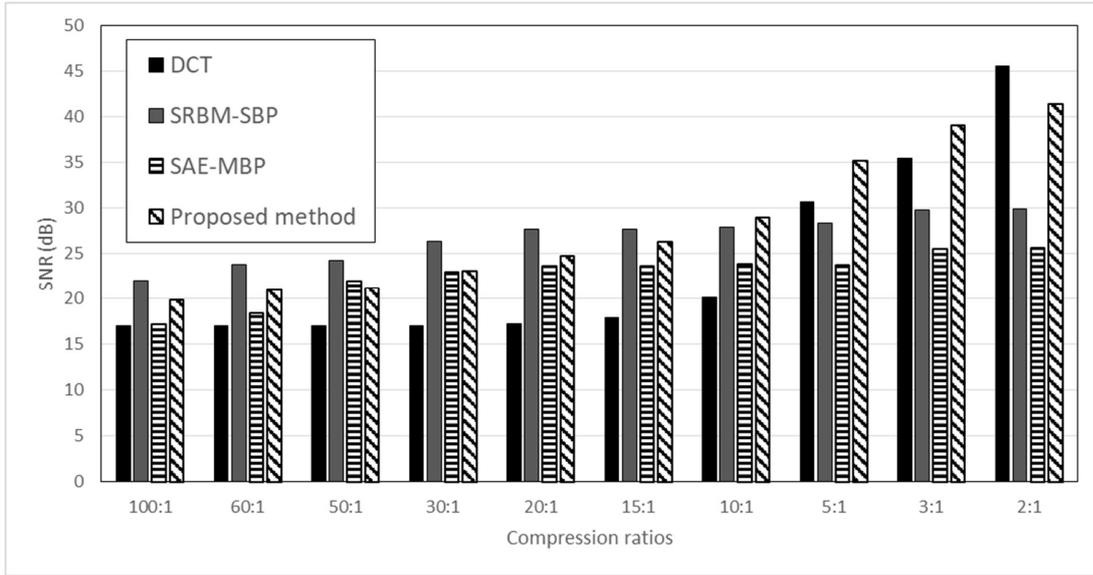


Figure 5.26. SNR as a function of compression ratios for all methods

Figure 5.26 shows the signal to noise ratio (SNR) in decibel (dB) from all techniques for different compression ratios. The SNR is related to NMSE using the following equation

$$\text{SNR (dB)} = -\log_{10} \text{NMSE}. \quad (5.53)$$

It can be seen that for compression ratios 100:1 to 15:1, the SRBM-SBP achieves the highest SNR. However, the SRBM-SBP fails to improve the SNR for less than 10:1 of compression ratio. For the DCT, the SNR doesn't increase significantly with number of frequency components increment, especially for high compression ratios. However, for a very low compression ratio of 2:1, the DCT achieves the best result. The SAE-MBP exhibits stagnant performance for all compression ratios. Our proposed method exhibits comparable performance with other methods for different compression ratios and shows the best performance for compression ratio of 10:1 to 3:1.

Table 5.8 Training duration comparison of the SAE-ELM using east Texas testing data for 10:1 compression ratio

Compression ratios	Times(s)		
	SRBM-SBP	SAE-MBP	Proposed method
100:1	18.48	8.14	0.0586
60:1	41.71	15.18	7.02
50:1	55.01	42.66	1.280
30:1	57.98	56.69	1.308
20:1	69.91	69.75	8.049
15:1	96.94	99.68	8.205
10:1	159.42	138.2	8.235
5:1	233.41	231.3	6.889
3:1	361.22	411.1	6.92
2:1	558.34	550.3	0.830

In Table 5.8, the training durations are taken from the architectures that achieve the best performances corresponding to Figure 5.26. High compression ratios tend to require more training times. For high compression ratio like 100:1, the SAE-MBP tends to terminate the training prematurely due to low gradient that result in short training duration. However, for each compression ratio, it can be seen that our proposed method achieves the shortest training duration.

5.4.4 Concluding remarks for compression using the SAE-ELM

In this section, we presented an asymmetric auto-encoder neural networks model seismic data compression namely the Stacked Auto-encoder Extreme Learning Machine (SAE-ELM). The encoder part of the model used non-linear activation function whereas the

decoder part used linear activation function. Second, all encoder hidden layer output values were sequentially calculated where the first hidden layer used training data as inputs. Third, the decoder weights were calculated using AE-ELM given the encoder hidden layer output values. Data compression is achieved by transforming the data to the code layer where the size is lower than the original data size. We compared our proposed method with other state of the art SAE models for seismic data compression using a real dataset. Our proposed method showed comparable performances to the other methods in term of reconstruction quality by achieving 1.28×10^{-3} of NMSE for 10:1 of compression ratio, in a significantly short training time with only 8.23s. For the same compression ratios, averagely, the SAE-ELM also achieved a better reconstruction quality than that of the linear transform like the DCT.

CHAPTER 6

CONCLUSIONS AND FUTURE WORKS

6.1 Conclusions

This thesis developed a suit of algorithms for seismic data compression over large seismic sensor networks. First, a distributed PCA algorithm (DPCA) was developed based on formulating a set of global PCs using the accumulated second order statistics propagated through the network sensor-by-sensor. The resulting global PCs are then broadcasted to all sensors so that local data is projected over these PCs instead of the local PCs. The resulting weights from the projection are transmitted for reconstruction. By selecting a suitable number of PCs, the DPCA offered compression ratios ranging from 2:1 to 200:1. The reconstruction quality of the DPCA was shown to be significantly better than the DCT and LPCA. The DPCA achieved high reconstruction quality with more than 20dB in SNR for a 7:1 compression ratio while classical methods only achieved less than 5:1 for the same level of reconstruction quality.

The second developed compression algorithm was a model based approach using a sum of exponentially decaying sinusoids which are called EDSWs. Each EDSW component was represented with a small number of parameters. The model parameters were sequentially estimated using a PSO optimization algorithm. The residual signals from the model were then compressed using fixed-length code-word quantization to enhance the reconstruction quality. The model based compression algorithm offered a high range of compression ratios from 5:1 to more than 100:1 with significantly better reconstruction

qualities than that of the LPCA and the DCT. Furthermore, when enhanced with residual encoding, the model based technique achieved near lossless reconstruction with a compression ratio of 3.4:1.

The third developed compression algorithm is based on a DNN trained with either RBM or ELM. With RBM based training, the DNN was decomposed into pairs of layers. Each pair of layers was pre-trained using RBM in an unsupervised fashion. The DNN was then fine-tuned using back-propagation techniques to achieve better results. In the scheme with the ELM, we introduced an alternative architecture for data compression using DNN. This scheme proposed an asymmetric model to perform seismic data compression that allows the parameters to be optimized using ELM over a short time. The DNN based approaches significantly outperformed the above mentioned methods. For similar ranges of compression ratios, the DNN-based prediction compression technique achieved an improvement of more than 20 dB compared with the LPC. The DNN was shown to achieve a near lossless reconstruction of 58dB in SNR at a compression ratio of 7:1. Furthermore, the use of ELM improved the training time of the DNN significantly as compared with RBM.

Our extensive experimental results showed that transform based and model based techniques are appropriate when the underlining assumptions, such as Gaussian distribution, linear models, finite number of parameters and white noise, are valid. However, such assumptions may not be appropriate for the case of seismic data which is subjected to many effects from the random nature of the earth substructure. For this reason, we have shown in this thesis that the best results can be achieved using machine

learning techniques. However, the computational cost of using such techniques may be substantial. For this reason, a fast learning based compression technique using deep networks trained with ELM was developed with very promising compression performance.

6.2 Future Research Directions

From the conclusions above, the possible research directions are listed below:

- 1. Analyzing the impact of seismic image quality on the accuracy of event detection.**

An insight look into the seismic image based geophysical interpretation from a different aspect, i.e., to comprehensively analyze how the signal quality affects the process of event detection will be performed. This analysis has not been done yet, to the best of our knowledge. The result of this task will give researchers working on seismic data compression a solid understanding of geophysical event detection.

- 2. Developing a new distortion metric specific for seismic image event detection**

Based on the previous research direction, a comprehensive mathematical relation between the raw seismic data and the final output of the image event detection will be investigated. Achieving this task requires numerical analysis of a large volume of seismic data and qualitative analysis on each step of image event detection. The perspective output of this task is a specifically designed distortion metric for geophysical interpretation.

3. Re-evaluating some existing benchmark data compression methods by applying the designed metric.

Based on the achievement of the previous objectives, re-evaluation to some of the existing lossy data compression methods by applying the designed metric will be performed. Successfully achieving this task is significant for the geophysical signal processing field. Researchers will have a relevant metric to evaluate the different lossy compression methods.

4. Developing a near lossless data compression algorithm for seismic data while preserving the seismic image based interpretation performance.

All the achievements of the previous tasks will be integrated and a novel algorithm specifically for the seismic image event detection oriented seismic data compression can be developed. The perspective output of this work will potentially be commercialized for the petroleum industry. Meanwhile, the theoretical output would be publishable.

REFERENCES

- [1] US EIA, "International Energy Outlook 2016," 2016.
- [2] BP Stats, "BP Statistical Review of World Energy June 2016," 2016.
- [3] H. J. Longwell, "The future of the oil and gas industry: past approaches, new challenges," *World Energy*, vol. 5, no. 3, pp. 100-104, 2002.
- [4] P. Kearey, M. Brooks and I. Hill, *An introduction to geophysical exploration*, John Wiley & Sons, 2013.
- [5] A. Al-Shuhail, S. Al-Dossary and W. Mousa, *Seismic Data Interpretation Using Digital Image Processing*, John Wiley & Sons., 2017.
- [6] O. Yilmaz, *Seismic data analysis: Processing, inversion, and interpretation of seismic data*, Society of exploration geophysicists, 2001.
- [7] B. Liu, M. Mohandes, H. Nuha, D. Mohamed and F. Faramarz, "A Distributed Principal Component Analysis Compression for Smart Seismic Acquisition Networks," *IEEE Transactions on Geoscience and Remote Sensing*, 2018.
- [8] Q. Liao, F. Gao and C. Rivera, *Wavelet-based seismic data compression*, Google Patents, 2015.
- [9] M. Girard, P. Faure and C. Paulet, "Onshore Seismic Acquisition: Updating an Efficient QC on Increasing Seismic Data Volume," in *EAGE workshop on Developments in Land Seismic Acquisition for Exploration*, Cairo, 2010.
- [10] R. Ergas, P. Donoho and J. Villasenor, "Method for reducing data storage and transmission requirements for seismic data. Chevron USA Inc.". U.S. Patent Patent 5,745,392., 1998.
- [11] K. Boman, "RigZone," [Online]. Available: https://www.rigzone.com/news/oil_gas/a/140418/big_data_growth_continues_in_seismic_surveys/?all=hg2. [Accessed 10 October 2017].
- [12] R. A. Ergas, P. L. Donoho and J. Villasenor, *Method for reducing data storage and*

transmission requirements for seismic data, Google Patents, 1998.

- [13] T. Rosten, V. A. Marthinussen, T. A. Ramstad and A. Perkis, "Filter bank optimization for high-dimensional compression of pre-stack seismic data," in *Acoustics, Speech, and Signal Processing, 1999. Proceedings., 1999 IEEE International Conference on*, 1999.
- [14] T. Rosten, T. A. Ramstad and L. Amundsen, "Lossless compression of seismic trace headers," in *62nd EAGE Conference & Exhibition*, 2000.
- [15] N. S. Jayant and P. Noll, "Digital coding of waveforms: principles and applications to speech and video," *Englewood Cliffs, NJ*, pp. 115-251, 1984.
- [16] L. C. Wood, "SEISMIC DATA COMPRESSION METHODS," *Geophysics*, vol. 39, pp. 499-525, 1974.
- [17] Sercel, "428XL Land Seismic Acquisition System," Houston, 2015.
- [18] K. M. Barry, D. A. Cavers and C. W. Kneale, "Recommended standards for digital tape formats," *Geophysics*, vol. 40, pp. 344-352, 1975.
- [19] H. Li, X. Tuo, T. Shen, M. J. Henderson, J. Courtois and M. Yan, "An improved lossless group compression algorithm for seismic data in SEG-Y and MiniSEED file formats," *Computers & Geosciences*, vol. 100, pp. 41-45, 2017.
- [20] B. G. Nickerson, P. A. Judd and L. A. Mayer, "Data structures for fast searching of SEG-Y seismic data," *Computers & Geosciences*, vol. 25, pp. 179-190, 1999.
- [21] D. Salomon, *A concise introduction to data compression*, Springer Science & Business Media, 2007.
- [22] D. Salomon, *Data compression: the complete reference*, Springer Science & Business Media, 2004.
- [23] M. J. Rubin, M. B. Wakin and T. Camp, "A comparison of on-mote lossy compression algorithms for wireless seismic data acquisition," in *Distributed Computing in Sensor Systems (DCOSS), 2014 IEEE International Conference on*, 2014.
- [24] M. J. Rubin, M. B. Wakin and T. Camp, "Lossy compression for wireless seismic data acquisition," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 9, pp. 236-252, 2016.
- [25] A. Payani, A. Abdi and F. Fekri, "Memory-assisted compression of seismic data: Tackling a large alphabet-size problem by statistical methods," in *SEG Technical Program Expanded Abstracts 2017*, Society of Exploration Geophysicists, 2017,

pp. 5567-5571.

- [26] D. L. Donoho, "Compressed sensing," *IEEE Transactions on information theory*, vol. 52, no. 4, pp. 1289--1306, 2006.
- [27] P. Donoho, R. Ergas and R. Polzer, "Development of seismic data compression methods for reliable, low-noise, performance," *SEG Technical Program Expanded Abstracts*, vol. 486, pp. 1903-1906, 1999.
- [28] P. Donoho, R. Ergas and J. Villasenor, "High-performance seismic trace compression," Houston, 1995.
- [29] J. D. Villasenor, R. A. Ergas and P. L. Donoho, "Seismic data compression using high-dimensional wavelet transforms," in *Data Compression Conference, 1996. DCC'96. Proceedings*, 1996.
- [30] T. Rosten, T. A. Ramstad and L. Amundsen, "Optimization of sub-band coding method for seismic data compression," *Geophysical Prospecting*, vol. 52, pp. 359-378, 2004.
- [31] C. A. Fajardo, C. A. Angulo, J. G. Mantilla, I. F. Obregon, J. Castillo, C. Pedraza and M. Reyes, "Computational Architecture for Fast Seismic Data Transmission between CPU and FPGA by Using Data Compression," in *Data Compression Conference (DCC), 2016*, 2016.
- [32] A. Kiely and F. Pollara, "Subband coding methods for seismic data compression," 1995.
- [33] J. M. Lervik, T. Rosten and T. A. Ramstad, "Subband seismic data compression: optimization and evaluation," in *Digital Signal Processing Workshop Proceedings, 1996.*, IEEE, 1996.
- [34] K. Xie, Z. Bai and W. Yu, "Fast seismic data compression based on high-efficiency SPIHT," *Electronics Letters*, vol. 50, pp. 365-367, 2014.
- [35] S. B. Jonsson and A. S. Spanias, "Seismic data compression," in *Computers and Communications, 1990. Conference Proceedings., Ninth Annual International Phoenix Conference on*, 1990.
- [36] A. S. Spanias, S. B. Jonsson and S. D. Stearns, "Transform coding algorithms for seismic data compression," in *Circuits and Systems, 1990., IEEE International Symposium on*, 1990.
- [37] A. S. Spanias, S. B. Jonsson and S. D. Stearns, "Transform methods for seismic data compression," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 29, pp. 407-416, 1991.

- [38] A. Z. Averbuch, F. Meyer, J.-O. Stromberg, R. Coifman and A. Vassiliou, "Low bit-rate efficient compression for seismic data," *IEEE Transactions on Image Processing*, vol. 10, pp. 1801-1814, 2001.
- [39] A. Payani, A. Abdi, X. Tian, F. Fekri and M. Mohandes, "Advances in Seismic Data Compression-Compression for seismic data acquisition," *IEEE Signal Processing Magazine*, pp. 51-61, 2018.
- [40] H. R. Khan and S. A. Zummo, "Functional Quantization-Based Data Compression in Seismic Acquisition," *Arabian Journal for Science and Engineering*, pp. 1-13, 2018.
- [41] A. Vassiliou and V. Wickerhauser, "Comparison of wavelet image coding schemes for seismic data compression," in *SEG Technical Program Expanded Abstracts 1997*, Society of Exploration Geophysicists, 1997, pp. 1334-1337.
- [42] D. A. Huffman, "A method for the construction of minimum-redundancy codes," *Proceedings of the IRE*, vol. 40, pp. 1098-1101, 1952.
- [43] W. Wu, Z. Yang, Q. Qin and F. Hu, "Adaptive seismic data compression using wavelet packets," in *Geoscience and Remote Sensing Symposium, 2006. IGARSS 2006. IEEE International Conference on*, 2006.
- [44] M. Fargues, S. D. Stearns and G. Coutu, "Seismic data compression using adaptive filters," in *Circuits and Systems, 1994., Proceedings of the 37th Midwest Symposium on*, 1994.
- [45] C. Angulo, C. Fajardo, O. Reyes and J. Castillo, "FPGA Implementation of a Huffman Decoder for High Speed Seismic Data Decompression," *Snowbird*, 2014.
- [46] G. Mandyam, N. Magotra and W. McCoy, "Lossless seismic data compression using adaptive linear prediction," in *Geoscience and Remote Sensing Symposium, 1996. IGARSS'96. 'Remote Sensing for a Sustainable Future.', International*, 1996.
- [47] A. O. Abanmi, S. A. Alshebeili and T. H. Alamri, "Lossless compression of seismic data," *Journal of the Franklin Institute*, vol. 343, pp. 340-351, 2006.
- [48] F. G. Meyer, "Fast compression of seismic data with local trigonometric bases," in *Proceedings of SPIE- The International Society for Optical Engineering*, 1999.
- [49] I. H. Witten, R. Neal and J. Cleary, "Arithmetic coding for data compression," *Communications of the ACM*, vol. 30, no. 6, pp. 520-540, 1987.
- [50] J. Ziv and A. Lempel, "A universal algorithm for sequential data compression," *IEEE Transactions on information theory*, vol. 23, pp. 337-343, 1977.

- [51] M. A. Razzaque, C. Bleakley and S. Dobson, "Compression in wireless sensor networks: A survey and comparative evaluation," *ACM Transactions on Sensor Networks (TOSN)*, vol. 10, p. 5, 2013.
- [52] S. D. Steams, "A technique for lossless compression of seismic data," in *Geoscience and Remote Sensing Symposium, 1992. IGARSS'92. International, 1992.*
- [53] N. Ahmed, T. Natarajan and K. R. Rao, "Discrete Cosine Transform," *IEEE TRANSACTIONS ON COMPUTERS*, vol. 23, pp. 90-93, 1974.
- [54] K.-Y. Huang, "Neural networks for seismic principal components analysis," *IEEE transactions on geoscience and remote sensing*, vol. 37, pp. 297-311, 1999.
- [55] I. F. Jones, "Applications of the Karhunen-Loeve transform in reflection seismology," 1985.
- [56] B. Liu, H. Nuha, M. Mohandes, M. Deriche and F. Fekri, "Distributed principal component analysis for data compression of sequential seismic sensor arrays," in *SEG Technical Program Expanded Abstracts 2016*, Society of Exploration Geophysicists, 2016, pp. 250-254.
- [57] L. C. Wood, "Seismic data compression methods," *Geophysics*, vol. 39, pp. 499-525, 1974.
- [58] P. Wang, M. Zhang, H. Song and N. Jiang, "A Semi-lossless Seismic Data Compression Algorithm Based on Discrete Cosine Transform," *JOURNAL OF INFORMATION & COMPUTATIONAL SCIENCE*, vol. 7, pp. 1331-1336, 2010.
- [59] G. Bernasconi and M. Vassallo, "Efficient data compression for seismic-while-drilling applications," *IEEE transactions on geoscience and remote sensing*, vol. 41, pp. 687-696, 2003.
- [60] R. L. De Queiroz, T. Q. Nguyen and K. R. Rao, "The GenLOT: Generalized linear-phase lapped orthogonal transform," *IEEE Transactions on Signal Processing*, vol. 44, pp. 497-507, 1996.
- [61] R. L. Queiroz, T. Q. Nguyen and K. R. Rao, "Generalized linear-phase lapped orthogonal transforms," in *Circuits and Systems, 1994. ISCAS'94., 1994 IEEE International Symposium on*, 1994.
- [62] L. Duval, T. Q. Nguyen and T. D. Tran, "On Progressive Seismic Data Compression using GenLOT," in *Proc. 33rd Conf. on Information Sciences and Systems (CISS)*, Baltimore, 1999.
- [63] L. C. Duval and T. Q. Nguyen, "Seismic data compression: a comparative study between GenLOT and wavelet compression," in *Proceedings of SPIE- The*

International Society for Optical Engineering, 1999.

- [64] L. C. Duval, T. Q. Nguyen and T. D. Tran, "Seismic data compression and QC using GenLOT," in *Proc. 61th EAGE Conference*, 1999.
- [65] L. C. Duval, V. Bui-Tran, T. Q. Nguyen and T. D. Tran, "GenLOT optimization techniques for seismic data compression," in *Acoustics, Speech, and Signal Processing, 2000. ICASSP'00. Proceedings. 2000 IEEE International Conference on*, 2000.
- [66] L. C. Duval, V. Bui-Tran, T. Q. Nguyen and T. D. Tran, "Seismic data compression using GenLOT: towards" optimality"?,," in *Data Compression Conference, 2000. Proceedings. DCC 2000*, 2000.
- [67] T. Nagai, M. Ikehara, M. Kaneko and A. Kurematsu, "Generalized unequal length lapped orthogonal transform for subband image coding," *IEEE transactions on signal processing*, vol. 48, pp. 3365-3378, 2000.
- [68] L. C. Duval and T. Nagai, "Seismic data compression using GULLOTS," in *Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on*, 2001.
- [69] C. Bosman and E. Reiter, "Seismic Data Compression Using Wavelet Transforms," in *SEG Technical Program Expanded Abstracts*, 1993.
- [70] P. L. Donoho, R. A. Ergas and R. S. Polzer, "Improved data management for seismic interpretation using compression," in *SPIE's International Symposium on Optical Science, Engineering, and Instrumentation*, 1998.
- [71] M. A. Simaan, "Wavelet compression in marine seismic data," in *OCEANS'98 Conference Proceedings*, 1998.
- [72] B. Yang, L. Liu and Q. Qin, "Research of seismic data compression method based on morphological wavelet," *Geomatics and Information Science of Wuhan University*, vol. 36, pp. 785-788, 2011.
- [73] Z. Feng, X. Zhang and Y. Li, "Some practical aspects of seismic data compression based on wavelet transform," *JOURNAL-TSINGHUA UNIVERSITY*, vol. 41, pp. 170-173, 2001.
- [74] A. .. Wu and S. .. Cao, "Noise-elimination and compression of seismic data using balanced biorthogonal multi-wavelets transform," *Oil Geophysical Prospecting*, vol. 39, p. 635, 2004.
- [75] L. C. Duval and T. Rosten, "Filter bank decomposition of seismic data with application to compression and denoising," in *SEG Technical Program Expanded*

Abstracts 2000, Society of Exploration Geophysicists, 2000, pp. 2055-2058.

- [76] J. Stigant, R. Ergas, P. Donoho, A. Minchella and P. and Galibert, "Field trial of seismic compression for real-time transmission," in *SEG Technical Program Expanded Abstracts*, 1995.
- [77] M. F. Khène and S. H. Abdul-Jauwad, "Efficient seismic compression using the lifting scheme," in *SEG Technical Program Expanded Abstracts 2000*, Calgary, 2000.
- [78] M. F. Khene and S. H. Abdul-Jauwad, "Adaptive seismic compression by wavelet shrinkage," in *Proceedings of the Tenth IEEE Workshop on Statistical Signal and Array Processing*, Pocono Manor, 2000.
- [79] H. Tang and Yang, "Seismic data compression using M-band wavelet transform," *Journal of the Chengdu Institute of Technology*, vol. 26, pp. 183-186, 1999.
- [80] L. Duval and V. Bui-Tran, "Compression denoising: using seismic compression for uncoherent noise removal," in *63rd European Assoc. of Geophysicists and Engineers (EAGE) Conference*, Amsterdam, 2001.
- [81] A. Wu and Y. Mu, "SEISMIC DATA COMPRESSION BY WAVELET PACKET METHOD," *JOURNAL OF THE UNIVERSITY OF PETROLEUM, CHINA*, vol. 6, 1996.
- [82] W. Z. He and A. D. Wu, "Seismic data compression and denoising by balanced orthogonal multiwavelet packet," in *Proceedings of the International Computer Congress on Wavelet Analysis and Its Applications, and Active Media Technology*, 2004.
- [83] M. A. Al-Moohimeed, "Towards an efficient compression algorithm for seismic data," in *Radio Science Conference, 2004. Proceedings. 2004 Asia-Pacific*, 2004.
- [84] Y. Luo and G. T. Schuster, "Wave packet transform and data compression," in *SEG Technical Program Expanded Abstracts 1992*, Society of Exploration Geophysicists, 1992, pp. 1187-1190.
- [85] F. Xu, Z. Zhang and Z. Gui, "Seismic data compression based on EZW algorithm," *Oil Geophysical Prospecting*, vol. 50, pp. 881-889, 2015.
- [86] K. Xie, H. Q. Yu and G. Y. Lu, "GPU-based large seismic data parallel compression," *Intelligence Computation and Evolutionary Computation*, Springer, Berlin, Heidelberg, pp. 339-345, 2013.
- [87] L. Duval, "Simultaneous seismic compression and denoising using a lapped transform coder," in *International Conference on Acoustics, Speech and Signal*

Processing (ICASSP), Orlando, 2002.

- [88] R.-S. Wu and Y. Wang, "New flexible segmentation technique in seismic data compression using local cosine transform," in *Proceedings of SPIE- The International Society for Optical Engineering*, 1999.
- [89] G. Matviyenko, "Optimized local trigonometric bases," *Applied and Computational Harmonic Analysis*, vol. 3, no. 4, pp. 301-323, 1996.
- [90] Y. Wang and R.-S. Wu, "Seismic data compression by an adaptive local cosine/sine transform and its effects on migration," *Geophysical Prospecting*, vol. 48, pp. 1009-1031, 2000.
- [91] Y. Geng, R. S. Wu and J. Gao, "Dreamlet transform applied to seismic data compression and its effects on migration," in *2009 SEG Annual Meeting*, 2009.
- [92] E. Liu, A. Payani and F. Fekri, "Seismic data compression using online double-sparse dictionary learning schemes," in *Data Compression Conference (DCC)*, 2017.
- [93] X. Tian, A. Abdi, E. Liu and F. Fekri, "Seismic Signal Compression Through Delay Compensated and Entropy Constrained Dictionary Learning," in *the 19th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, Kalamata, 2018.
- [94] A. Abdi, A. Payani and F. Fekri, "Learning dictionary for efficient signal compression," in *International Conference In Acoustics, Speech and Signal Processing (ICASSP)*, New Orleans, 2017.
- [95] P. Lindstrom, P. Chen and E.-J. Lee, "Reducing disk storage of full-3d seismic waveform tomography (f3dt) through lossy online compression," *Computers & Geosciences*, vol. 93, pp. 45-54, 2016.
- [96] K.-Y. Huang, "Seismic principal components analysis using neural networks," in *SEG Technical Program Expanded Abstracts 1996*, Society of Exploration Geophysicists, 1996, pp. 1259-1262.
- [97] K.-Y. Huang, "Seismic principal components analysis using neural networks," in *Geophysical Applications of Artificial Neural Networks and Fuzzy Logic*, Springer, 2003, pp. 103-122.
- [98] T. A. Reddy, K. R. Devi and S. V. Gangashetty, "Nonlinear principal component analysis for seismic data compression," in *Recent Advances in Information Technology (RAIT), 2012 1st International Conference on*, 2012.
- [99] R. Ergas, R. Polzer, P. Donoho and J. Villasenor, "Measuring seismic data compression: What losses are acceptable?," in *SEG Technical Program Expanded*

Abstracts, 1996.

- [100] W. A. Mousa and A. A. Al-Shuhail, "Processing of Seismic Reflection Data Using MATLAB™," *Synthesis Lectures on Signal Processing*, vol. 5, pp. 1-97, 2011.
- [101] Q. He, L. Wang and B. Liu, "Parameter estimation for chaotic systems by particle swarm optimization," *Chaos, Solitons & Fractals*, vol. 34, pp. 654-661, 2007.
- [102] Q. Guo, H. Zhang, J. Tian, L. Liang and Z. Shang, "A nonlinear multiparameter prestack seismic inversion method based on hybrid optimization approach," *Arabian Journal of Geosciences*, vol. 11, p. 48, 2018.
- [103] M. Schwaab, E. C. Biscaia Jr, J. L. Monteiro and J. C. Pinto, "Nonlinear parameter estimation through particle swarm optimization," *Chemical Engineering Science*, vol. 63, pp. 1542-1552, 2008.
- [104] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*, 1998.
- [105] D. Liu, C. Cheng, Q. Fu, Y. Zhang, Y. Hu, D. Zhao, M. I. Khan and M. A. Faiz, "Complexity measurement of precipitation series in urban areas based on particle swarm optimized multiscale entropy," *Arabian Journal of Geosciences*, vol. 11, p. 83, 2018.
- [106] J. Kennedy and R. C. Eberhart, "A discrete binary version of the particle swarm algorithm," in *Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on*, 1997.
- [107] E. W. Forgy, "Cluster analysis of multivariate data: efficiency versus interpretability models," *Biometrics*, vol. 61, pp. 768-769, 1965.
- [108] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath and others, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, pp. 82-97, 2012.
- [109] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, pp. 504-507, 2006.
- [110] Y. Bengio, P. Lamblin, D. Popovici and H. Larochelle, "Greedy layer-wise training of deep networks," in *Advances in neural information processing systems*, 2007.
- [111] Y. Bengio, Y. LeCun and others, "Scaling learning algorithms towards AI," *Large-scale kernel machines*, vol. 34, pp. 1-41, 2007.

- [112] W. Huang, G. Song, H. Hong and K. Xie, "Deep Architecture for Traffic Flow Prediction: Deep Belief Networks With Multitask Learning," *IEEE Trans. Intelligent Transportation Systems*, vol. 15, no. 5, pp. 2191-2201, 2014.
- [113] A. Koesdwiady, R. Soua and F. Karray, "Improving traffic flow prediction with weather information in connected cars: a deep learning approach," *IEEE Transactions on Vehicular Technology*, , vol. 65, no. 12, pp. 9508-9517, 2016.
- [114] A. Said, A. Mohamed, T. Elfouly, K. Harras and Z. Wang, "Multimodal deep learning approach for joint EEG-EMG data compression and classification," in *In Wireless Communications and Networking Conference (WCNC)*, 2017.
- [115] M. T. M. Al-Said, A. Abdellatif, A. Mohamed, T. Elfouly, K. Harras and M. O'Connor, "A deep learning approach for vital signs compression and energy efficient delivery in mHealth systems," *IEEE Access*, vol. 6, pp. 33727-33739, 2018.
- [116] N. Sriraam and C. Eswaran, "Performance evaluation of neural network and linear predictors for near-lossless compression of EEG signals," *IEEE Transactions on Information Technology in Biomedicine*, vol. 12, no. 1, pp. 87-93, 2008.
- [117] C. Tan and C. Eswaran, "Using autoencoders for mammogram compression," *Journal of medical systems*, vol. 35, no. 1, pp. 49-58, 2011.
- [118] N. Srivastava and R. R. Salakhutdinov, "Multimodal learning with deep boltzmann machine," in *Advances in neural information processing systems*, 2012.
- [119] Y. Ollivier, "Auto-encoders: reconstruction versus compression," *arXiv preprint*, 2014.
- [120] D. Del Testa and M. Rossi, "Lightweight lossy compression of biometric patterns via denoising autoencoders," *IEEE Signal Processing Letters*, vol. 22, no. 12, pp. 2304-2308, 2015.
- [121] M. F. Moller, "A scaled conjugate gradient algorithm for fast supervised learning," *Neural networks*, vol. 6, pp. 525-533, 1993.
- [122] X. Zhang, H. Liu, X. Wang, L. Dong, Q. Wu and R. Mohan, "Speed and convergence properties of gradient algorithms for optimization of IMRT," *Medical physics*, vol. 31, pp. 1141-1152, 2004.
- [123] UTAM, "UTAM Seismic Data Library (Part Three)," 2016.
- [124] R. Kannan and C. Eswaran, "Lossless compression schemes for ECG signals using neural network predictors," *EURASIP Journal on Applied Signal Processing*, vol. 2007, pp. 102-102, 2007.

- [125] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath and others, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, pp. 82-97, 2012.
- [126] W. Huang, G. Song, H. Hong and K. Xie, "Deep architecture for traffic flow prediction: deep belief networks with multitask learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, pp. 2191-2201, 2014.
- [127] D. Harnik, E. Khaitzin, D. Sotnikov and S. Taharlev, "A fast implementation of deflate," in *Data Compression Conference (DCC), 2014*, 2014.
- [128] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio and P. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion.," *Journal of machine learning research*, pp. 3371-3408, 2010.
- [129] G.-B. Huang, Q.-Y. Zhu and C.-K. Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, pp. 489-501, 2006.
- [130] G. Huang, H. Zhou, X. Ding and R. Zhang, "Extreme learning machine for regression and multiclass classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 42, no. 2, pp. 513-529, 2012.
- [131] L. L. C. Kasun, H. Zhou, G.-B. Huang and C. M. Vong, "Representational Learning with Extreme Learning Machine for Big Data," *IEEE intelligent systems*, vol. 28, no. 6, pp. 31-34, 2013.
- [132] G. Huang, L. Chen and C. Siew, "Universal approximation using incremental constructive feedforward networks with random hidden nodes," *IEEE Trans. Neural Networks*, vol. 17, no. 4, pp. 879-892., 2006.
- [133] W. Johnson and J. Lindenstrauss, "Extensions of Lipschitz mappings into a Hilbert space," *Contemporary mathematics*, vol. 26, no. 1, pp. 189-206, 1984.
- [134] Y. Wang and R.-S. Wu, "Improvements on seismic data compression and migration using compressed data with the flexible segmentation scheme for local cosine transform," in *SEG Technical Program Expanded Abstracts 2000*, Society of Exploration Geophysicists, 2000, pp. 2048-2051.
- [135] J. Shlens, "A tutorial on principal component analysis," *arXiv preprint arXiv:1404.1100*, 2014.
- [136] H. S. Malvar and D. H. Staelin, "The LOT: Transform coding without blocking effects," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, pp. 553-559, 1989.

- [137] E. Liu, A. Payani and F. Fekri, "Seismic Data Compression Using Online Double-Sparse Dictionary Learning Schemes," in *Data Compression Conference (DCC), 2017*, 2017.
- [138] C. Liu, M. Han and L. Han, "Application of principal component analysis for frequency-domain full waveform inversion," *SEG Technical Program Expanded Abstracts*, vol. 564, pp. 1-5, 2012.
- [139] J. Li, K. Das, G. Fu, R. Li and R. Wu, "The Bayesian lasso for genome-wide association studies.," *Bioinformatics (Oxford, England)*, vol. 27, pp. 516-23, 2011.
- [140] A. Krizhevsky, I. Sutskever and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012.
- [141] G. HONGSHENG and C. H. E. N. JUNLIANG, "REAL-TIME SEISMIC WAVE DATA COMPRESSION METHODS [J]," *Acta Seismologica Sinica*, vol. 1, p. 006, 1989.
- [142] T. Hennig, P. Maass, J. Hayano and S. Heinrichs, "Exponential distribution of long heart beat intervals during atrial fibrillation and their relevance for white noise behaviour in power spectrum.," *Journal of biological physics*, vol. 32, pp. 383-92, 11 2006.
- [143] D. C. Hagen, "The application of principal components analysis to seismic data sets," *Geoplotation*, vol. 20, pp. 93-111, 1982.
- [144] I. Guvenc, "Towards Practical Design of Impulse Radio Ultrawideband Systems:," 2006.
- [145] A. Graves, A.-r. Mohamed and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, 2013.
- [146] A. Gersho and R. M. Gray, *Vector quantization and signal compression*, vol. 159, Springer Science & Business Media, 2012.
- [147] R. Gallager, "Variations on a theme by Huffman," *IEEE Transactions on Information Theory*, vol. 24, pp. 668-674, 1978.
- [148] A. J. Davis, "Linear prediction coding for compressing of seismic data," *The Journal of the Acoustical Society of America*, vol. 78, pp. 1153-1153, 1985.
- [149] G. W. Cottrell and P. Munro, "Principal components analysis of images via back propagation," in *Visual Communications and Image Processing'88: Third in a*

Series, 1988.

- [150] G. W. Cottrell, P. Munro and D. Zipser, "Learning internal representations from gray-scale images: An example of extensional programming," in *Ninth annual conference of the cognitive science society*, 1987.
- [151] T. Chen, "Seismic data compression," 1995.
- [152] C. M. Bishop and M. E. Tipping, "Variational relevance vector machines," in *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, 2000.
- [153] Y. Bengio, P. Simard and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, vol. 5, pp. 157-166, 1994.
- [154] M. N. Al-Ali, "Land seismic data acquisition and preprocessing," 2007.
- [155] A. Fischer and C. Igel, "Training restricted Boltzmann machines: An introduction," *Pattern Recognition*, vol. 47, pp. 25-39, 2014.
- [156] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Micro Machine and Human Science, 1995. MHS'95., Proceedings of the Sixth International Symposium on*, 1995.
- [157] R. Mehboob, S. A. Khan, Z. Ahmed, H. Jamal and M. Shahbaz, "Multigig lossless data compression device," *IEEE Transactions on Consumer Electronics*, vol. 56, 2010.
- [158] T. Nagai, M. Ikehara, M. Kaneko and A. Kurematsu, "Generalized unequal length lapped orthogonal transform for subband image coding," in *Proceedings IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP).*, Istanbul, 2000.
- [159] Y. W. Tee and G. E. Hinton, "Rate-coded Restricted Boltzmann machines for face recognition," *Neural Information Processing Systems*, vol. 13, p. 908–914, 2000.
- [160] H. Nuha, M. Mohandes and B. Liu, "Seismic data compression using auto-associative neural network and restricted Boltzmann machine," in *SEG Technical Program Expanded Abstracts*, Anaheim, 2018.

VITAE

- Name : Hilal Hudannuha
- Nationality : Indonesia
- Date of Birth : 2 December 1986
- Email : hilalnuha@gmail.com
- Permanent Address : Pradha Ciganitri A24, Bojongsoang, Bandung, Indonesia
40288

Journal Publications

1. Hilal Nuha, Adil Balghonaim, Bo Liu, Mohamed Mohandes, and Faramarz Fekri. "Deep Neural Networks with Extreme Learning Machine for Seismic Data Compression" Submitted to Arabian Journal for Science and Engineering on December 2018.
2. Hilal Nuha, Bo Liu, Mohamed Mohandes, and Faramarz Fekri. "Seismic Data Modelling and Compression using Particle Swarm Optimization", Submitted to Arabian Journal of Geosciences on March 2018
3. Mohamed Mohandes, Shafiqur Rehman, Hilal Nuha, M. Saiful Islam. "Wind Speed Predictability Accuracy with Height using LiDAR based Measurements and Artificial Neural Networks". Submitted to Applied Artificial Intelligence on February 2019.
4. Bo Liu, Mohamed Mohandes, Hilal Nuha, Mohamed Deriche, Faramarz Fekri and James McClellan. "A Multitone Model-Based Seismic Data Compression", Submitted to IEEE Transactions on Systems, Man, and Cybernetics: Systems on August 2017.
5. Bo Liu, Mohamed Mohandes, Hilal Nuha, Mohamed Deriche, and Faramarz Fekri. "A Distributed Principal Component Analysis Compression for Smart Seismic Acquisition Networks." *IEEE Transactions on Geoscience and Remote Sensing* 56, no. 6 (2018): 3020-3029.

Conference Publications

1. Hilal Nuha, Mohamed Mohandes, and Bo Liu. "Seismic-data compression using auto-associative neural networks and restricted Boltzmann machine." In *SEG*

Technical Program Expanded Abstracts 2018. Society of Exploration Geophysicists, 2018. Anaheim

2. Hilal Nuha, Bo Liu, M. Mohandes, and M Deriche. Seismic Data Compression using Signal Alignment and PCA. *The 9th IEEE-GCC conference*. 08 May - 11 May 2017. Bahrain.
3. Hilal Nuha, M. Mohandes, M Deriche, and N. Iqbal. "Near Lossless Seismic Data Compression Using Signal Projection Technique." *The 4th International Geoscience & Geomatics Conference*. 23-25 November 2015. Bahrain.
4. Bo Liu, Mohamed Mohandes, and Hilal Nuha. "Seismic model estimation using particle swarm optimization." In *SEG Technical Program Expanded Abstracts 2018*. Society of Exploration Geophysicists, 2018. Anaheim
5. Bo Liu, Hilal Nuha, Mohamed Mohandes, Mohamed Deriche, and Faramarz Fekri. "Disributed principal component analysis for data compression of sequential seismic sensor arrays." In *SEG Technical Program Expanded Abstracts 2016*, pp. 250-254. Society of Exploration Geophysicists, 2016. Dallas

Patents

1. A Novel Data Compression Method for Sequential Seismic Sensor Arrays (submitted);
2. A Novel Method for Seismic Sensor Network Data Compression (submitted)
3. A Novel Model-based Compression Method for Seismic Data (submitted)