Representation learning with efficient extreme learning machine auto-encoders

Zhang, Guanhao

2021

Zhang, G. (2021). Representation learning with efficient extreme learning machine auto-encoders. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/146297>

<https://doi.org/10.32657/10356/146297>

This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License (CC BY-NC 4.0).

Downloaded on 26 Apr 2023 09:04:23 SGT

REPRESENTATION LEARNING WITH EFFICIENT EXTREME LEARNING MACHINE AUTO-ENCODERS

ZHANG GUANGHAO

School of Electrical & Electronic Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirement for the degree of
Doctor of Philosophy

2021

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research, is free of plagiarised materials, and has not been submitted for a higher degree to any other University or Institution.

05 Jan. 2020

.....

Date

Guanghao
.....

ZHANG GUANGHAO

Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is free of plagiarism and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

05 Jan. 2020

.....

Date

.....

Prof. Huang Guang-Bin

Authorship Attribution Statement

This thesis contains material from 3 papers published in the following peer-reviewed journal(s) / from papers accepted at conferences in which I am listed as an author.

Chapter 3 is published as Zhang G, Cui D, Mao S, et al. Sparse Bayesian Learning for Extreme Learning Machine Auto-encoder. International Conference on Extreme Learning Machine. Springer, Cham, 2018: 319-327. Chapter 3 involves extended work accepted by International Journal of Machine Learning and Cybernetics. Zhang G, Cui D, Mao S, et al. Unsupervised Feature Learning with Sparse Bayesian Auto-Encoding based Extreme Learning Machine. The contributions of the co-authors are as follows:

- I prepared the manuscript drafts. The manuscript was revised by Ms Mao Shangbo, Dr Cui Dongshun and Prof Huang Guang-Bin.
- I finished all the required codes and experiments. I also analyzed the data.

Chapter 4 is published as Zhang G, Li Y, et al. R-ELMNet: Regularized extreme learning machine network. Neural Networks (2020). The contributions of the co-authors are as follows:

- I proposed the idea. I prepared the manuscript drafts. The manuscript was revised by Dr Li Yue, Dr Cui Dongshun, Ms Mao Shangbo and Prof Huang Guang-Bin.
- I finished all the required codes and experiments. I also analyzed the data.

05 Jan. 2020

.....

Date

Guanghao

.....
ZHANG GUANGHAO

Acknowledgements

First and foremost, I would like to express my heartfelt thanks and appreciation to my supervisor, Professor Huang Guangbin, for his precious advice, continuous guidance, constructive comments, and invaluable helps throughout my research work.

Many thanks to my group members and colleagues, Dr. Cui Dongshun, Dr. Tu Enmei, Ms. Mao Shangbo, Mr. Han Wei, Mr. Li Yue and all lab mates at the School of Electrical and Electronic Engineering at Nanyang Technological University for their kind assistance, technical support, happy gatherings, and encouraging chats.

I would also like to thank my family for their unconditional support, care, understanding and encouragement.

The last but not the least, I am honored to express my sincere appreciation for the patience and trust of Ms. Sun Hongya from thousands of miles away.

Contents

Acknowledgements	v
Summary	xi
List of Figures	xiv
List of Tables	xix
Symbols and Acronyms	xxi
1 Introduction	1
1.1 Research Background	1
1.2 Objectives and Major Contributions	3
1.3 Organization	5
2 Literature Review	7
2.1 Extreme Learning Machines	8
2.1.1 Overview of Extreme Learning Machines	8
2.1.2 Bayesian Extreme Learning Machine	11

2.1.3	Sparse Bayesian Extreme Learning Machine	13
2.1.4	Extreme Learning Machines for Clustering	14
2.2	Unsupervised Feature Learning	15
2.2.1	Non-negative Matrix Factorization	15
2.2.2	Principal Component Analysis (PCA)	16
2.2.3	Extreme Learning Machine Auto-Encoder	18
2.2.4	Sparse Auto-Encoder	20
2.2.5	Variational Auto-Encoder	22
2.3	Unsupervised Feature Learning-based Multi-Layer Extreme Learning Machine Structure	24
2.3.1	Deep Extreme Learning Machine	25
2.3.2	Multi-Layer Extreme Learning Machines	26
2.3.3	Local Receptive Fields-based Extreme Learning Machine	28
2.3.4	Non-Gradient Convolutional Neural Network	29
2.3.4.1	Principal Component Analysis Network	30
2.3.4.2	Extreme Learning Machine Network	33
2.3.4.3	Hierarchical Extreme Learning Machine Network	35
2.3.4.4	Network Comparison of PCANet, ELMNet, H-ELMNet and LRF-ELM	36
3	Unsupervised Feature Learning with Sparse Bayesian Auto-Encoding based Extreme Learning Machine	39
3.1	Motivation	40

3.2	Sparse Bayesian Learning for Extreme Learning Machine Auto-Encoder	41
3.2.1	Single-Output Sparse Bayesian ELM-AE	41
3.2.2	Batch-Size Training for Multi-Output Sparse Bayesian ELM-AE	46
3.2.3	Hidden Nodes Selection	50
3.3	Experiments	51
3.3.1	Experimental Setup	51
3.3.2	Datasets Preparation	52
3.3.3	Parameter Analysis	54
3.3.4	Performance Comparison	59
3.3.5	Time Efficiency Improvement with Batch-Size Training	61
3.4	Conclusions	61
4	R-ELMNet: Regularized Extreme Learning Machine Network	63
4.1	Motivation	64
4.2	Regularized Extreme Learning Machine Network	66
4.2.1	Regularized ELM Auto-Encoder	66
4.2.2	Learning Details	70
4.2.3	Orthogonality Analysis	71
4.2.4	Overall Pipeline	72
4.3	Experiments	73
4.3.1	Datasets Preparation	73

4.3.2	Experimental Setup	75
4.3.3	Parameter Analysis and Selection	75
4.3.4	Performance Comparison	78
4.3.5	Comparison with Deeper CNN	80
4.3.6	Learning Efficiency Discussion	81
4.3.7	Orthogonality Visualization	82
4.3.8	Feature Map Visualization	83
4.4	Conclusions and Future Work	85
5	Unified ELM-AE for Dimension Reduction and Extensive Appli- cations	87
5.1	Motivation	88
5.2	Proposed Method	89
5.2.1	Unified ELM-AE for Dimension Reduction	89
5.2.2	Comparison with PCA, linear ELM-AE, nonlinear ELM-AE, and SB-ELM-AE	94
5.2.3	Comparison with SAE, VAE, and SOM	95
5.3	Extensive Applications	97
5.3.1	Local Receptive Fields-based Extreme Learning Machine with U-ELM-AE	97
5.3.2	Non-Gradient Convolutional Neural Network with U-ELM-AE	99
5.4	Experiments	100
5.4.1	Experimental Setup	100

5.4.2	Datasets Preparation	101
5.4.3	Sensitivity to Hyper-Parameter L	103
5.4.4	Performance Comparison for Dimension Reduction	103
5.4.5	Performance Comparison as Plug-and-Play Role in LRF-ELM and NG-CNN	112
5.5	Conclusion	115
6	Stacking Projection Regularized ELM-AE with U-ELM-AE for ML-ELM	117
6.1	Proposed Method	118
6.2	Experiments	120
6.3	Conclusion	123
7	Conclusion and Future Work	126
7.1	Conclusion	126
7.2	Future Work	128
	List of Author's Publications	129
	Bibliography	131

Summary

Extreme Learning Machine (ELM) is a specialized Single Layer Feedforward Neural network (SLFN). The traditional SLFN is trained by Back-Propagation (BP), which has the problem of local minimum and slow learning speed. In contrast to that, hidden weights of ELM are randomly generated without any update while learning. The output weights have an analytical solution. And ELM has been successfully explored in classification and regression scenarios.

Extreme Learning Machine Auto-Encoder (ELM-AE) was proposed as a variant of general ELM for unsupervised feature learning. Specifically, ELM-AE introduces linear ELM-AE, nonlinear ELM-AE, and sparse ELM-AE. Multi-Layer Extreme Learning Machine (ML-ELM) is built via stacking multiple nonlinear ELM-AEs, which presents a competitive generalization capability with other multi-layer neural networks such as Deep Boltzmann Machine (DBM) and Deep Belief Network (DBN). Based on ML-ELM, Hierarchical Extreme Learning Machine (H-ELM) mainly presents that the ℓ_1 -regularized ELM-AE variant can improve the performance for various applications. This thesis introduces the Bayesian learning scheme into ELM-AE referred to as Sparse Bayesian Extreme Learning Machine Auto-Encoder (SB-ELM-AE). Also, a parallel training strategy is proposed to accelerate the Bayesian learning procedure. The overall neural network, similar to ML-ELM and H-ELM, is referred to as Sparse Bayesian Auto-Encoding based Extreme Learning Machine (SBAE-ELM). Experimentally, it shows that the neural network with stacking SB-ELM-AEs have better generalization performance on traditional classification and face related challenges.

Principal Component Analysis Network (PCANet), as an unsupervised shallow network, demonstrates noticeable effectiveness on datasets of various volumes.

It carries a two-layer convolution with PCA as a filter learning method, followed by a block-wise histogram post-processing stage. Following the structure of PCANet, ELM-AE variants are employed to replace PCAs role, which comes from the Extreme Learning Machine Network (ELMNet) and Hierarchical Extreme Learning Machine Network (H-ELMNet). ELMNet emphasizes the importance of orthogonal projection. H-ELMNet introduces a specialized ELM-AE variant with complex pre-processing steps. This thesis proposes a Regularized Extreme Learning Machine Auto-Encoder (R-ELM-AE), which combines nonlinear ELM learning and approximately orthogonal characteristic. Based on R-ELM-AE and the pipeline of PCANet, this thesis proposes Regularized Extreme Learning Machine Network (R-ELMNet) accordingly with minimum implementation. Experiments on image classification of various volumes show the effectiveness compared to unsupervised neural networks, including PCANet, ELMNet, and H-ELMNet. Also, R-ELMNet presents competitive performance with supervised convolutional neural networks.

Despite the success of ELM-AE variants, it is not broadly used in the traditional scenarios where PCA is commonly integrated, such as the dimension reduction role in machine learning. There are two main reasons which restrict the propagation of ELM-AE variants. Firstly, the value scale after data transformation is not bounded that we need to add data normalization or value scaling operations to eliminate this problem. Secondly, PCA has only one hyper-parameter, the reduced dimension. While ELM-AE variants generally require additional hyper-parameters. For example, nonlinear ELM-AE needs the ℓ_2 -regularization item, the selection range of which is commonly from $1e^{-8}$ to $1e^8$. The hyper-parameter space would be exponentially expanded due to involving the hyper-parameters from feature post-processing or stacking multiple ELM-AEs. Considering PCA often acts as the plug-and-play role of dimension reduction in machine learning that we expect a simple variant of ELM-AE, which can be verified the adaptability to any model with minimal trials. Hence this thesis proposes a Unified Extreme Learning Machine Auto-Encoder (U-ELM-AE), which presents competitive performance with

other ELM-AE variants and PCA, and importantly involves no additional hyperparameters. Experiments have shown its effectiveness and efficiency for image dimension reduction, compared with PCA and ELM-AE variants. Also, U-ELM-AE can be conveniently integrated into Local Receptive Fields based Extreme Learning Machine (LRF-ELM) and PCANet to present improvements.

The U-ELM-AE is only suitable for dimension reduction case, while nonlinear ELM-AE can be used for dimension expansion. It is necessary to handle scenarios where the input dimension is small. Thus, an effective multi-layer ELM is proposed: 1) if the ELM-AE is used for dimension expansion, then a new regularization is applied to nonlinear ELM-AE to constrain the output value scale; 2) if the ELM-AE is used for dimension reduction, then U-ELM-AE is employed. With such a structure, it can achieve efficiency and competitive performance with ML-ELM, H-ELM, and SBAE-ELM.

List of Figures

2.1	Illustration of a standard ELM network, $\mathbf{X} \in \mathbb{R}^{n \times d}$ denotes the input, $\mathbf{H} \in \mathbb{R}^{n \times L}$ represents the ELM embedding and $\mathbf{Y} \in \mathbb{R}^{n \times t}$ is the target. L is the number of hidden neurons. ELM embedding $\mathbf{H}$, which is learning-free, is computed based on a random matrix $\mathbf{A}$. Only the weights connecting hidden nodes to outputs require learning.	9
2.2	Illustration of various activation functions listed from Equation 2.2 to Equation 2.5.	10
2.3	Illustration of the SAE's network structure.	21
2.4	Illustration of a standard VAE structure.	24
2.5	Illustration of network structure of Deep ELM. The bottom symbols represent the outputs of corresponding neurons.	25
2.6	A simpler inference structure of Deep ELM, as Deep ELM applies no activation on $\mathbf{H}^1 \boldsymbol{\beta}^2$	26
2.7	Illustration of the network of ML-ELM with two stacked ELM-AEs.	27
2.8	Illustration of channel-separable convolution. There are two feature maps in level i , the same convolutional kernel is applied on each map to generate stacked outputs. Then the channels of feature maps increase exponentially.	31

2.9	Illustration of NG-CNN pipeline for one data sample from second filter convolution. In this figure, the resulted channels of the first filter convolution stage are two for simplification. Cs-convolution denotes channel-separable convolution, the post-processing stage consists of binarization and block-wise histogram. The pre-processing stage is not shown here. Symbol h and w show that the height and width of the sliding window for the histogram both are 2. All the outputs are concatenated together to form a final sparse representation. . . .	34
2.10	Framework comparison of PCANet, ELMNet, H-ELMNet, and LRF-ELM. CS-Convolution is short for channel-separable convolution. $ELM-AE^{Or}$ and $ELM-AE^{No}$ represent two ELM auto-encoder variants. The former is orthogonal ELM-AE, and the latter is nonlinear case. The main differences are emphasized with bold font.	37
3.1	The illustration shows the forward connection β_j and the backward γ_i . The β_j relates all hidden nodes with the j -th output node and is independent to β_{-j} . While for ELM-AE, the γ_i links all output dimensions with the i -th hidden node.	41
3.2	Illustration of batch-size matrix operations and corresponding shapes. The upper presents Equation 3.13, which is the batch-size mean matrix $\check{\beta}$. The lower denotes Equation 3.14 for the calculation of batch-size prior variance matrix. The matrix shapes of $\check{\alpha}$, $\check{\Sigma}$, $\check{\beta}$, and $\check{Y}$ are also shown. The subscript i of $\check{\alpha}_i$, $\check{\Sigma}_i$, $\check{\beta}_i$, and $\check{Y}_i$ is ignored for simplification.	48
3.3	Samples of Jaffe dataset. For each row, faces from left to right express angry, disgusting, fear, happy, neutral, sad, and surprised emotion, respectively.	53
3.4	Samples of pre-processed Jaffe faces.	53

3.5	Samples of Orl dataset. The officially provided front faces were directly used.	54
3.6	Illustration of the parameter influence of d_n on all benchmark datasets. Blue line represents accuracy influenced by the first auto-encoder, and red plotting denotes effect from the second auto-encoder. As there is only one SB-ELM-AE for Isolet, Jaffe, Orl, Fashion, and Letters dataset that the red lines are ignored.	56
3.7	Illustration of the influence of the number of hidden neurons on single ELM-AE. The experiments were conducted for each dataset. Note that the performances gain marginal improvement when the number of hidden neurons is larger than 2000. Considering the hyper-parameter searching and implementation efficiency, the maximal number of hidden neurons is set to 2000.	57
4.1	The top figure presents the results of nonlinear ELM-AE for feature reduction. It was performed on the Iris dataset. Note that feature along the x-axis shows a much bigger value scale and variance compared with feature along the y-axis. The bottom figure shows the result of orthogonal ELM-AE. Although it achieves secondary linear-separability, values of each dimension keep comparable scale and variance.	65
4.2	Illustration of the R-ELMNet's network structure.	73
4.3	Accuracy sensitivity to parameter α is illustrated. The blue line presents the effect of varying α of the first convolution stage by fixing the best α in the second convolution stage. The red line denotes the influence of changing α in the second convolution stage.	77
4.4	Robustness comparison on the various training volume. R-ELMNet shows better accuracy while training size is less than 20000.	81

4.5	Orthogonality visualization of $Mat = \beta\beta^T$. The upper row demonstrates Mat with directly learned β , while we normalize the row vector of β first before plotting lower figures. The color block within each picture denotes a value close to 1 while it shows white. The difference within rightmost column also shows that the magnitude of corresponding β_1 is huge.	83
4.6	Feature maps from two cs-convolutional layers are shown for filtering learning methods. They came from one same sample of Fashion dataset. Feature map values were clipped with range $[-2,2]$ for equal plotting scheme. The color approaches yellow, while the pixel value is close to 2. There are 8 and 64 feature maps in layer one and two, respectively. Only the first 8 of 64 feature maps in layer two are illustrated.	84
5.1	Illustration of the influence of feature dimension on Coil20 and Coil100 with linear SVM classifier. Note that the upper figure's maximum features are 400 as training split on Coil20 only contains 420 samples. Considering PCA and linear ELM-AE requirement, the maximum feature-length was set to 400.	104
5.2	Illustration of the influence of the feature dimension on Fashion(1) and Fashion with linear SVM classifier.	105
5.3	Illustration of the influence of the feature dimension on Coil20 and Coil100 with ELM classifier. The maximum feature dimension was set to 400 on Coil20.	106
5.4	Illustration of the influence of the feature dimension on Fashion(1) and Fashion with an ELM classifier.	107

5.5	Mean accuracy and standard deviation over all feature dimension choices for each method (U-ELM-AE, PCA, linear ELM-AE, non-linear ELM-AE and SB-ELM-AE) on Coil20 and Coil100 with the linear SVM classifier. The mean accuracy was calculated over the performance of each L and shown by the histogram. The standard deviation is shown by the error bar.	108
5.6	Mean accuracy and standard deviation over all feature dimension choices for each method (U-ELM-AE, PCA, linear ELM-AE, non-linear ELM-AE, and SB-ELM-AE) on Fashion(1) and Fashion with the linear SVM classifier. The mean accuracy is shown by the histogram. The standard deviation is illustrated by the error bar. . . .	109
5.7	Mean accuracy and standard deviation over all feature dimension choices for each method (U-ELM-AE, PCA, linear ELM-AE, non-linear ELM-AE, and SB-ELM-AE) on Coil20 and Coil100 with the ELM classifier. The mean accuracy is shown by the histogram. The standard deviation is illustrated by the error bar.	110
5.8	Mean accuracy and standard deviation over all feature dimension choices for each method (U-ELM-AE, PCA, linear ELM-AE, non-linear ELM-AE, and SB-ELM-AE) on Fashion(1) and Fashion with the ELM classifier. The mean accuracy is shown by the histogram. The standard deviation is illustrated by the error bar.	111
6.1	Illustration of the ML-ELM _U 's network structure.	119
6.2	Illustration of the effect of γ on the classification accuracy.	122

List of Tables

3.1	Datasets summary	52
3.2	Mean accuracy (%) comparison.	58
3.3	Network structure and hyper-parameters	60
3.4	Time-cost (seconds) comparison of single-output training and batch-size training.	61
4.1	Datasets summary.	73
4.2	Parameter selection.	76
4.3	Mean accuracy comparison on scalable classification datasets.	79
4.4	Learning Efficiency (minutes) Comparison on Big Datasets.	82
4.5	Model Complexity Comparison.	82
5.1	The property comparisons of U-ELM-AE, PCA, ELM-AE (linear), ELM-AE (nonlinear), and SB-ELM-AE. A check symbol indicates the method has the corresponding property.	96
5.2	The testing accuracy and training time on Coil20, Coil100, and Fashion datasets of U-ELM-AE, PCA, ELM-AE (linear), ELM-AE (non-linear), and SB-ELM-AE using the linear SVM classifier.	113

5.3	The testing accuracy and training time on Coil20, Coil100, and Fashion datasets of U-ELM-AE, PCA, ELM-AE (linear), ELM-AE (non-linear), and SB-ELM-AE using the ELM classifier.	114
5.4	Mean accuracy comparison as a plug-and-play role in LRF-ELM and NG-CNN. Methods are divided into four groups: single-layer ELM classifier, LRF-ELM related methods, NG-CNNs, and CNN models.	115
6.1	Table shows the testing accuracy and training time comparison on several datasets. ML-ELM ₁ and ML-ELM ₂ represent applying a nonlinear ELM-AE without and with normalization, respectively, as the first ELM-AE followed by a U-ELM-AE.	121
6.2	Table shows the network structure of ML-ELM _{<i>U</i>} . For example, the structure 784-1200(0.2)-500-8000-100 on Coil100 illustrates the input dimension, the first hidden layer (γ), the second hidden layer, the third layer, and the output dimension, respectively.	123
6.3	Mean accuracy comparison on scalable classification datasets. . . .	124
6.4	Training Time (minutes) Comparison on Big Datasets.	125

Symbols and Acronyms

Symbols

$d \in \mathbb{R}$	The data dimension of input
$\mathbf{H} \in \mathbb{R}^{n \times L}$	The hidden activation matrix
$L \in \mathbb{R}$	The number of hidden neurons
$n \in \mathbb{R}$	The number of samples
$\mathbf{X} \in \mathbb{R}^{n \times d}$	The input samples
$\mathbf{Y} \in \mathbb{R}^{n \times t}$	The output targets

Acronyms

AE	Auto-Encoder
B-ELM	Bayesian Extreme Learning Machine
BP	Back-Propagation
CNN	Convolutional Neural Network
ELM	Extreme Learning Machine
ELM-AE	Extreme Learning Machine Auto-Encoder
H-ELM	Hierarchical Extreme Learning Machine
H-ELMNet	Hierarchical Extreme Learning Machine Network
LRF-ELM	Local Receptive Field based Extreme Learning Machine
ML-ELM	Multi-Layer Extreme Learning Machine
ML-ELM _U	Stacking PR-ELM-AE and U-ELM-AE for ML-ELM
NG-CNN	Non-Gradient Convolutional Neural Network

NMF	Non-negative Matrix Factorization
PCA	Principal Component Analysis
PR-ELM-AE	Projection Regularized Extreme Learning Machine Auto-Encoder
SAE	Sparse Auto-Encoder
SB-ELM	Sparse Bayesian Extreme Learning Machine
SB-ELM-AE	Sparse Bayesian Extreme Learning Machine Auto-Encoder
SBAE-ELM	Sparse Bayesian Auto-Encoding based Extreme Learning Machine
SELM-AE	Sparse Extreme Learning Machine Auto-Encoder
SLFN	Single Layer Feedforward Neural network
SVD	Singular Value Decomposition
U-ELM-AE	Unified Extreme Learning Machine Auto-Encoder
VAE	Variational Auto-Encoder

Chapter 1

Introduction

1.1 Research Background

Neural networks have been broadly studied by the machine learning research community. Neural networks can be categorized as fully connected and locally connected network, shallow and deep network, or unsupervised and supervised network, according to the neuron connection type, the number of layers, or whether having training targets. The simplest neural network structure should be a Single Layer Feedforward Neural network (SLFN). It consists of a single hidden layer and necessary input/output layers. The unknown weights include the input weights connecting the input to the hidden layer and the output weights joining the hidden layer with the output layer.

Moreover, activation functions [1] are commonly applied to the output of neurons. Typically, SLFN is trained for supervised tasks, such as classification and regression, with Back-Propagation (BP) [2] learning method. The BP-trained methods require the overall objective should be differentiable or piecewise differentiable. Thus, it is convenient for integrating regularization items or revising objectives. Nevertheless, BP-based SLFN has the drawbacks of low training speed and might converge to a local minimum.

Extreme Learning Machine (ELM) [3–8] is a ‘specialized’ SLFN that the input weights can be just randomly generated and learning-free, hence only the output weights are trained with an analytical solution. The theory of ELM presents the proof of its universal approximation capability as long as the activation function is nonlinear piecewise continuous. ELM shows significant improvement on learning efficiency and generalization on many tasks, such as protein prediction [9], power utility analysis [10], biomedical analysis [11] and remote sensing [12].

The projection via random input weights, followed by the activation function can be regarded as the simple feature learning processing for ELM. However, data might contain noise or meaningless information that could negatively affect the final performance. The random mapping of ELM can not handle such problems. Feature learning generally aims to reduce unwanted dimensions or project data into a more generalized feature space. According to Lee et al. [13], feature learning algorithms can be categorized into holistic-based methods and parts-based algorithms. Principal Component Analysis (PCA) [14, 15] can be treated as holistic-based algorithm. PCA learns the eigenvectors and eigenvalues of the covariance matrix of data. It ranks the eigenvectors according to the eigenvalues from large to small. PCA shows that projecting data along with top eigenvectors can project data into dimensions that describe the most variance. Non-negative Matrix Factorization (NMF) [13, 16] and Tied weight Auto-Encoder (TAE) [17] are two classic parts-based algorithms.

Although ELM was originally proposed for supervised learning, recent research has successfully extended ELM to clustering [18–22] or unsupervised feature learning [23–26]. Extreme Learning Machine Auto-Encoder (ELM-AE) [23, 24] is among the most frequently cited ELM-based feature learning algorithms. ELM-AE utilizes SLFN structure with output the same as input. After learning the output weights, ELM-AE shows projecting data along with the transpose of output weights can obtain more generalized features compared to Restricted Boltzmann Machine (RBM) [27] or TAE.

Multi-layer neural networks, namely stacking multiple layers, could present

more competitive and generalized performance, especially when data is large. Typically Deep Belief Networks (DBN) [28], Deep Boltzmann Machine (DBM) [29] and Multi-Layer Extreme Learning Machine (ML-ELM) [23] can be categorized to same group as they only contain full connections. Inspired by biological discovery [30], the local receptive fields-based connection performs better than a fully connected layer, especially on image-related tasks. Accordingly, Convolutional Neural Networks (CNN) [31–34] were developed for supervised tasks. Meanwhile, it has also shown local receptive fields-based structure [25, 26, 35–37] is effective for unsupervised feature extraction.

1.2 Objectives and Major Contributions

The overall objectives are listed as follows:

- (1) Develop more effective and efficient ELM-AE variants for dimension reduction and dimension expansion.
- (2) Build more generalized multi-layer ELM with fully connected layers for unsupervised feature learning.
- (3) Construct the state-of-the-art local receptive fields-based multi-layer ELM for representation learning.

Inspired by the Bayesian inference-based ELM classifier’s success, this thesis explores the effective and efficient ELM-AE training pipeline based on sparse Bayesian learning. A proper probability scheme is addressed for unsupervised representation learning. To overcome the drawback of learning inefficiency, which is caused by the iterative Bayesian learning scheme, a parallel training framework is also introduced. It is referred to as batch-size training via fully utilizing the CPU cores and memory without explicit multi-threading programming. Furthermore, SB-ELM-AE shows pruning neurons according to the estimated prior

variance could improve performance further. The overall multi-layer ELM-based structure, referred to as SBAE-ELM, is addressed to present an improvement to related fully connected multi-layer ELMs, including ML-ELM and H-ELM.

For the image-related tasks, fully connected multi-layer ELMs commonly present less competitive performance than local receptive fields-based methods. Nevertheless, there exists a bridge that relates these two types. Extracting image patches and forming a two-dimensional matrix, fully connected ways can project image patch into multi-dimensional feature space. After reshaping the projected feature map with the proper size, local receptive fields-based methods complete the convolutional procedure. The convolutional kernels might be a random matrix or generated via PCA or ELM-AEs. Based on the pipeline of PCANet, the R-ELM-AE is proposed with geometrical regularization term to retain the distances of patches. Also, R-ELM-AE avoids the time-consuming LCN or whitening post-processing methods, introduced in H-ELMNet, which achieves a minimal implementation level.

From SB-ELM-AE to R-ELM-AE, although the improvements are shown for fully connected multi-layer ELM and NG-CNN pipeline, the generalization capability of these ELM-AE variants are not well studied. To be more precise, SB-ELM-AE requires long training time and complex implementation skills. R-ELM-AE mainly works within the NG-CNN pipeline. The most desired properties of the ELM-AE variant, highlighted in this thesis, include nonlinear ELM random mapping, restricted projection, learning efficiency, and easy for the extension as a plug-and-play method in other frameworks. Hence, the U-ELM-AE is presented to fulfill these objectives with the condition of orthogonality of output weights. An analytical solution is shown without any hyper-parameters. The experiments on dimension reduction illustrate the effectiveness and efficiency with PCA, linear ELM-AE, nonlinear ELM-AE, and SB-ELM-AE. Meanwhile, the evidence shows U-ELM-AE can be simply integrated into LRF-ELM and NG-CNN for performance improvement and implementation convenience.

Based on the achievements of U-ELM-AE, the focus rolls back to fully connected multi-layer ELMs. As U-ELM-AE only has a solution when the number of hidden neurons is smaller than the output dimension, that U-ELM-AE is only for dimension reduction. Meanwhile, U-ELM-AE requires the input values scale should be comparable with hidden activations (usually fall into $[-1, 1]$ or $[0, 1]$), that U-ELM-AE can not follow a nonlinear ELM-AE or SB-ELM-AE and simply act as a second ELM-AE. Although the output feature of the first ELM-AE can be normalized, there is no observed consistent improvement by such methods. Hence, the PR-ELM-AE is proposed as the first ELM-AE for dimension expansion with regularization term to restrict the output scale. U-ELM-AE then performs dimension reduction to remove unwanted features of the first ELM-AE. The overall structure achieves better performance compared to ML-ELM, H-ELM, and SBAE-ELM.

Among the proposed methods, the U-ELM-AE could be highlighted first, as it summarizes the advantages and advantages of the ELM-AEs proposed in Chapters 3, 4, and other works. It is more concise and elegant, achieves more competitive performance. Nevertheless, the SB-ELM-AE explores the sparse Bayesian learning scheme for ELM-AE, and the R-ELM-AE introduces a simple yet effective auto-encoder learning objective into the NG-CNN framework. All these works have built strong motivation and inspired mathematical derivation.

1.3 Organization

The thesis is presented below:

Chapter 2 reviews the related works from three views: 1) Extreme Learning Machine, its extensions in Bayesian inference and clustering; 2) NMF, PCA, TAE, ELM-AE, and related methods for dimension reduction or feature learning; 3) multi-layer neural network-based unsupervised feature learning, including LRF-ELM, Deep ELM, multi-layer ELM, PCANet and so on.

Chapter 3 introduces the unsupervised feature learning-based ELM-AE within the sparse Bayesian learning framework, referred to as SB-ELM-AE.

Chapter 4 focuses on the filter learning method of NG-CNN and proposes the R-ELM-AE specified for the performance improvement with minimal implementation level compared with PCANet, ELMNet, and H-ELMNet.

Chapter 5 generalizes the R-ELM-AE, referred to as U-ELM-AE with an analytical solution and presents its capability of dimension reduction and feature learning within LRF-ELM and NG-CNN.

Chapter 6 designs the unsupervised feature learning network with stacked ELM-AE variants, incorporating U-ELM-AE for dimension reduction and PR-ELM-AE for dimension expansion. A summary of the performance of all the related methods are compared.

Chapter 7 draws the conclusion of the overall thesis.

Chapter 2

Literature Review

Chapter 2 first reviews the Extreme Learning Machine (ELM) and related extensions, mainly including Bayesian Extreme Learning Machine (B-ELM), Sparse Bayesian Extreme Learning Machine (SB-ELM), and so on, which are associated with supervised classification or regression. Then an overview of feature learning methods is introduced, focusing on bridging ELMs with unsupervised feature learning. Lastly, the unsupervised deep feature learning networks are reviewed in Chapter 2.3.

2.1 Extreme Learning Machines

The Single Layer Feedforward Neural network (SLFN) is usually trained by Back-Propagation (BP) [2] method. However, such neural networks are restricted by low training speed and local minimum problem. Also, it takes a longer time for hyper-parameter tuning. Huang et al. [3–8] proposed the Extreme Learning Machine (ELM), which learns unknown weights with an analytical solution to overcome the drawbacks of low learning speed and local minimum. Bayesian ELM (B-ELM) [38] explains the ELM from the probabilistic view under the Bayesian learning framework. However, the experimental results fail to present a significant improvement in classification and regression tasks. The following sparse Bayesian ELM (SB-ELM) [39] introduces a sparse Bayesian learning pipeline into ELM. Quantitative experiments prove its effectiveness. Beyond the classification and regression tasks, the ELM has also been well studied for clustering, which is briefly reviewed in this chapter.

2.1.1 Overview of Extreme Learning Machines

In contrast to general SLFN, ELM shows the hidden weights can be randomly generated and learning-free. Thus, only the hidden weights are necessary to train. ELM [3, 4, 40] can perform universal approximation as long as the activation function is piecewise continuous. Also, ELM requires very limited hyper-parameters, such as the number of hidden nodes and the activation function type.

Given input data $\mathbf{X} \in \mathbb{R}^{n \times d}$ and targets $\mathbf{Y} \in \mathbb{R}^{n \times t}$, where n, d and t denote the number of samples, the data dimension and the target dimension, respectively. The classification or regression tasks can be expressed in the unified framework by ELM. The network structure consists of two parts: 1) ELM feature mapping and 2) ELM learning. The ELM feature mapping is fulfilled by the multiplication of $\mathbf{X}$ with matrix $\mathbf{A}$, where $\mathbf{A} \in \mathbb{R}^{d \times L}$ is randomly generated and L denotes the dimension after projection. Meanwhile, L represents the number of hidden neurons,

as illustrated in Figure 2.1. Within the overall training procedure, matrix $\mathbf{A}$ keeps fixed and learning-free. The hidden activation matrix $\mathbf{H}$ is computed as follows:

$$\mathbf{H} = g(\mathbf{X}\mathbf{A}), \quad (2.1)$$

where $g(\cdot)$ denotes a nonlinear piecewise continuous activation function. Let $\mathbf{X} = [\mathbf{x}_1^T, \dots, \mathbf{x}_n^T]^T$ and $\mathbf{A} = [\mathbf{a}_1, \dots, \mathbf{a}_L]$, then $g(\mathbf{a}_i, \mathbf{x}_j)$ presents the i -th activation of j -th sample. Commonly used activation functions are listed as below:

FIGURE 2.1: Illustration of a standard ELM network, $\mathbf{X} \in \mathbb{R}^{n \times d}$ denotes the input, $\mathbf{H} \in \mathbb{R}^{n \times L}$ represents the ELM embedding and $\mathbf{Y} \in \mathbb{R}^{n \times t}$ is the target. L is the number of hidden neurons. ELM embedding $\mathbf{H}$, which is learning-free, is computed based on a random matrix $\mathbf{A}$. Only the weights connecting hidden nodes to outputs require learning.

Sigmoid function:

$$g(\mathbf{a}, \mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{x}\mathbf{a})}. \quad (2.2)$$

Tanh function:

$$g(\mathbf{a}, \mathbf{x}) = \frac{\exp(\mathbf{x}\mathbf{a}) - \exp(-\mathbf{x}\mathbf{a})}{\exp(\mathbf{x}\mathbf{a}) + \exp(-\mathbf{x}\mathbf{a})}. \quad (2.3)$$

Gaussian function:

$$g(\mathbf{a}, \mathbf{x}) = \exp(-\|\mathbf{x} - \mathbf{a}^T\|_2^2). \quad (2.4)$$

Multiquadric function:

$$g(\mathbf{a}, \mathbf{x}) = (\|\mathbf{x} - \mathbf{a}^T\|_2^2)^{1/2}. \quad (2.5)$$

FIGURE 2.2: Illustration of various activation functions listed from Equation 2.2 to Equation 2.5.

In the stage of ELM learning, ELM aims to minimize the training error and the norm of the output weights $\boldsymbol{\beta}$, which is different from conventional methods [41, 42]. The generalized objective for ELM learning is illustrated below:

$$\text{Minimize} : \|\mathbf{H}\boldsymbol{\beta} - \mathbf{Y}\|_p^c + C \|\boldsymbol{\beta}\|_q^d, \quad (2.6)$$

where $c > 0$, $d > 0$, $p, q = 0, \frac{1}{2}, 1, \dots, \infty$, C is the trade-off factor and $\mathbf{Y}$ denotes the outputs.

According to the Barlett theory [43], the regularization item in 2.6 can improve the generalization capability. When $c = d = p = q = 2$ and $C = 0$, the

solution of Equation 2.6 is:

$$\boldsymbol{\beta} = \mathbf{H}^\dagger \mathbf{Y}, \quad (2.7)$$

where $\mathbf{H}^\dagger$ is the Moore-Penrose generalized inverse of $\mathbf{H}$.

With the condition $C > 0$ and $c = d = p = q = 2$, the solution is equivalent to the ridge regression [44]. When the number of hidden neurons L is smaller than the number of samples n , the analytical solution of Equation 2.6 is:

$$\boldsymbol{\beta} = (C\mathbf{I} + \mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T \mathbf{Y}. \quad (2.8)$$

When the number of hidden neurons is larger than the number of samples, the corresponding solution is changed as below:

$$\boldsymbol{\beta} = \mathbf{H}^T (C\mathbf{I} + \mathbf{H} \mathbf{H}^T)^{-1} \mathbf{Y}. \quad (2.9)$$

For the sake of clarification, the significant differences and advantages of ELM compared to RVFL [45–49] and QuickNet [42, 50, 51] are summarized. ELM presents the generalized framework for classification and regression. Also, ELM is efficient for clustering or feature learning, while RVFL mainly focuses on regression. QuickNet and RVFL design the direct link between input and output. Apparently, ELM shows a simplified network structure. Especially, ELM [3, 5, 40] proposes the proof of its universal approximation capability. The RVFL’s universal approximation capability is only proved when the hidden activation is semi-random. That is, the input weights $\mathbf{A}$ can be random while the input biases b should be learned. ELM theory [42, 50, 51] shows all the input weights can be randomly generated as long as the activation function is nonlinear piecewise continuous.

2.1.2 Bayesian Extreme Learning Machine

Bayesian linear regression and classification [52] methods optimize output weights within a probability framework instead of fitting to data directly. Hence, they

can gain higher generalization. Bayesian ELM (B-ELM) [38] combines Bayesian methodology with ELM, where the output weights follow a Gaussian prior distribution.

To be more precise, B-ELM uses random feature $\mathbf{H} = [\mathbf{h}_1, \dots, \mathbf{h}_i, \dots, \mathbf{h}_n]^T \in \mathbb{R}^{n \times L}$ to replace data $\mathbf{X}$ as the input of Bayesian inference, where $\mathbf{h}_i$ represents the hidden activation of i -th sample. The output is $\mathbf{y} \in \mathbb{R}^{n \times 1}$, that y_i is scalar value. B-ELM models i -th output y_i with output weights $\boldsymbol{\beta} \in \mathbb{R}^{L \times 1}$ following the Equation (2.10), in which ϵ is independent Gaussian noise.

$$y_i = \mathbf{h}_i^T \boldsymbol{\beta} + \epsilon. \quad (2.10)$$

Assuming ϵ has zero mean and variance δ^{-1} , then Equation (2.10) leads to a conditional definition:

$$p(y_i | \mathbf{h}_i, \boldsymbol{\beta}) = \mathcal{N}(\mathbf{h}_i^T \boldsymbol{\beta}; \delta^{-1}). \quad (2.11)$$

As shown in [38, 53], B-ELM also sets prior distribution of $\boldsymbol{\beta}$ be a normal distribution with zero mean and covariance $\alpha^{-1} \mathbf{I}$. According to [53, 54], the estimation of mean value $\mathbf{m}$ and variance $\boldsymbol{\Sigma}$ of posterior distribution follow:

$$\begin{aligned} \mathbf{m} &= \delta \boldsymbol{\Sigma} \mathbf{H}^T \mathbf{y}, \\ \boldsymbol{\Sigma} &= (\alpha \mathbf{I} + \delta \mathbf{H}^T \mathbf{H})^{-1}. \end{aligned} \quad (2.12)$$

Note that, if α is handcraft parameter and δ is 1, the solution of Equation (2.12) matches ℓ_2 -regularization ELM. And α is exactly related to the ℓ_2 -regularization factor. Nevertheless, parameter α within Bayesian inference can be learned as shown in 2.13 based on ML-II [55] or Evidence Procedure [56].

$$\begin{aligned}\gamma &= n - \alpha \cdot \text{trace}(\mathbf{\Sigma}), \\ \alpha &= \frac{\gamma}{\mathbf{m}^T \mathbf{m}}.\end{aligned}\tag{2.13}$$

Parameters from 2.12 to 2.13 can be iteratively updated until the difference of the norm of $\mathbf{m}$ between successive iterations is smaller than a given gap.

2.1.3 Sparse Bayesian Extreme Learning Machine

Sparse Bayesian ELM (SB-ELM) [39] exploits the sparse Bayesian learning for the output weights of the ELM classifier. It tries to find the sparse estimation of each element of output weights by imposing independent prior distribution. The input and output pair of SB-ELM is $[\mathbf{H} \in \mathbb{R}^{n \times L}, \mathbf{Y} \in \mathbb{R}^{n \times 1}]$.

While in the scenario of classification, y_i denotes the binary label target with 0 or 1 of i -th sample. Hence, SB-ELM models the $p(y_i|\mathbf{h}_i, \boldsymbol{\beta})$ with Bernoulli distribution. The conditional probability is written as:

$$p(y_i|\mathbf{h}_i, \boldsymbol{\beta}) = \sigma\{\Gamma(\mathbf{h}_i; \boldsymbol{\beta})\}^{y_i} [1 - \sigma\{\Gamma(\mathbf{h}_i; \boldsymbol{\beta})\}]^{1-y_i},\tag{2.14}$$

where $\Gamma(\mathbf{h}_i; \boldsymbol{\beta}) = \mathbf{h}_i^T \boldsymbol{\beta}$, $\boldsymbol{\beta} \in \mathbb{R}^{L \times 1}$, and $\sigma(\cdot)$ is sigmoid function:

$$\sigma(\Gamma(\mathbf{h}_i; \boldsymbol{\beta})) = \frac{1}{1 + e^{-\Gamma(\mathbf{h}_i; \boldsymbol{\beta})}}.\tag{2.15}$$

SB-ELM assumes each element of $\boldsymbol{\beta}$ follows zero mean Gaussian prior distribution [57], $p(\beta_k|\alpha_k) = \mathcal{N}(0, \alpha_k^{-1})$. The iterative estimation of mean value of $\boldsymbol{\beta}$ is derived as Equation (2.16).

$$\begin{aligned}
\boldsymbol{\beta} &= [\mathbf{A} + \mathbf{H}^T \mathbf{B} \mathbf{H}]^{-1} \mathbf{H}^T \mathbf{B} \hat{\mathbf{y}}, \\
\mathbf{A} &= \text{diagflat}(\boldsymbol{\alpha}), \\
\mathbf{B} &= \text{diagflat}([t_1(1 - t_1), \dots, t_n(1 - t_n)]), \\
t_i &= \sigma(\Gamma(\mathbf{h}_i; \boldsymbol{\beta}_{old})), \\
\hat{\mathbf{y}} &= \mathbf{H} \boldsymbol{\beta}_{old} + \mathbf{B}^{-1}(\mathbf{y} - \mathbf{t}),
\end{aligned} \tag{2.16}$$

where $\text{diagflat}(\cdot)$ denotes the function to create a two-dimensional matrix with the input as as diagonal.

2.1.4 Extreme Learning Machines for Clustering

Although ELM was originally proposed for supervised classification or regression, many studies have been presented on extending ELM into clustering scenario. He et al. [18] presented that the effective and efficient clustering based on k-means [58] or Non-negative Matrix Factorization (NMF) [13] on ELM feature mapping space compared to clustering on original data space.

Following that, several studies focused on capturing manifold regularization. Un-Supervised ELM (US-ELM) [19] introduced Laplacian Eigenmaps (LE) [59] as the regularization term, related to spectral clustering (SC) [60]. The final objective of US-ELM consists of Laplacian regularization and the norm of $\boldsymbol{\beta}$. To avoid a degenerated solution, it also contains the condition: $(\mathbf{H}\boldsymbol{\beta})^T \mathbf{H}\boldsymbol{\beta} = \mathbf{I}$. Peng et al. [20] proposed that incorporating local manifold structure and global discriminative information can improve clustering performance. Discriminative Embedded Clustering (DEC) [21] is a framework that allows joint embedding and clustering. Inspired by that, ELM-JEC [22] combines Laplacian regularization and DEC with ELM, which has the property of structure-preserving and separability maximizing.

2.2 Unsupervised Feature Learning

Feature learning aims to transform data into a more generalized feature space by removing redundant dimensions or increasing the data dimension. Principal Component Analysis (PCA) [14, 15] and Non-negative Matrix Factorization (NMF) [13, 16] are two most frequently cited dimension reduction methods. PCA and NMF can be categorized into holistic-based and parts-based algorithms by Lee et al. [13], respectively. Generalized Relevance Learning Vector Quantization (GRLVQ) [61] learns the prototype positions with the given number of prototypes and weights each feature dimension with a relevance weight. Thus, GRLVQ could select more important features with big relevance. Tied weight Auto-Encoder (TAE) [17] and Extreme Learning Machine Auto-Encoder (ELM-AE) [23, 24] all take single-layer neural network for unsupervised feature learning (dimension reduction, dimension expansion and equal dimension projection). Sparse Auto-Encoder (SAE) [62] presents a sparsity-regularized auto-encoder with back-propagation learning method. Variational Auto-Encoder (VAE) [63] proposes a stochastic variation inference and learning algorithm, the encoding process of which follows a posterior probabilistic distribution. The details are reviewed in subsequent subsections.

2.2.1 Non-negative Matrix Factorization

Non-negative Matrix Factorization (NMF) [13, 16, 64, 65] factorizes given non-negative matrix $\mathbf{X}$ into two non-negative matrices $\mathbf{H}$ and $\mathbf{W}$, the former is referred to as coefficient matrix and the latter is called basis matrix. NMF requires all elements of $\mathbf{X}$ are positive. It is consistent with biological observation: neurons have only positive firing rates. Unlike other feature learning methods, which allow the sign of neurons to be positive or negative, NMF constrains the input, the coefficient matrix, and the basis matrix to contain non-negative values. The square

loss objective of NMF is defined as below:

$$\begin{aligned} \text{Minimize : } & \| \mathbf{X} - \mathbf{H}\mathbf{W} \|^2, \\ \text{Subject to : } & \mathbf{H} \geq 0, \mathbf{W} \geq 0. \end{aligned} \quad (2.17)$$

With multiplicative update rules [16], the coefficient matrix $\mathbf{H}$ and basis matrix $\mathbf{W}$ can be iteratively learned as follows:

$$\begin{aligned} \mathbf{H}^{k+1} &= \mathbf{H}^k \odot \frac{\mathbf{X}\mathbf{W}^T}{\mathbf{H}^k\mathbf{W}^k(\mathbf{W}^k)^T}, \\ \mathbf{W}^{k+1} &= \mathbf{W}^k \odot \frac{(\mathbf{H}^k)^T\mathbf{X}}{(\mathbf{H}^k)^T\mathbf{H}^k\mathbf{W}^k}. \end{aligned} \quad (2.18)$$

Note that matrix division in Equation 2.18 is actually element-wise, and the symbol $\odot$ denotes the element-wise matrix multiplication. NMF has shown its effectiveness in many applications, such as recommender systems [66, 67], document clustering [68], or bioinformatics [69]. NMF can perform feature learning by projecting data $\mathbf{X}$ along with the basis matrix $\mathbf{W}$ as follows:

$$\mathbf{X}^{proj} = \mathbf{X}\mathbf{W}^T. \quad (2.19)$$

2.2.2 Principal Component Analysis (PCA)

Principal Component Analysis (PCA) [14, 15] projects data onto the transformed feature space via the orthogonal transformation matrix. The objective of PCA is to remove dimensions with low variance and keep dimensions with high variance. Firstly, it aims to find a matrix $\hat{\mathbf{V}}$ based on the objective (2.20).

$$\begin{aligned} \text{Maximize : } & \underset{\hat{\mathbf{V}}}{\text{trace}(\hat{\mathbf{V}}^T\mathbf{X}^T\mathbf{X}\hat{\mathbf{V}})}, \\ \text{Subject to : } & \hat{\mathbf{V}}^T\hat{\mathbf{V}} = \mathbf{I}. \end{aligned} \quad (2.20)$$

The objective (2.20) can be solved by applying spectral decomposition on the covariance matrix $\mathbf{X}^T \mathbf{X}$, assuming $\mathbf{X}$ is centered with subtracted column means. We get eigenvectors $\hat{\mathbf{V}} = [\mathbf{v}_1, \dots, \mathbf{v}_d]$ and corresponding eigenvalues $\mathbf{E} = [e_1, \dots, e_d]$. Here, we may assume columns of $\hat{\mathbf{V}}$ are sorted according to their eigenvalues from big to small. The final matrix $\hat{\mathbf{V}}$ for dimension reduction is learned based on the knowledge that the first column vector of $\hat{\mathbf{V}}$ can describe the most variance direction of data, the second column vector can represent the second most variance, and so on. Given the reduced dimension L ($L < d$), PCA selects the top L eigenvectors to form $\mathbf{V}$. Hence, eigenvectors with low eigenvalues are removed, and data is projected via the remaining eigenvectors $\mathbf{V}$:

$$\mathbf{X}^{proj} = \mathbf{X}\mathbf{V}. \quad (2.21)$$

There exists a strong relationship between PCA and Multidimensional Scaling (MDS) [70]. The task of MDS is to find a low-dimensional embedding $\mathbf{Y} \in \mathbb{R}^{n \times L}$, given the $n \times n$ matrix $\mathbf{D}$ of pairwise distances between n samples of $\mathbf{X}$. The corresponding objective of MDS is to minimize:

$$\| \mathbf{D} - \mathbf{D}_{\mathbf{Y}} \|^2, \quad (2.22)$$

where $\mathbf{D}_{\mathbf{Y}}$ denotes the matrix of pairwise distances based on $\mathbf{Y}$.

The classic MDS, also known as Torgerson MDS, replaces the $\mathbf{D}$ by Gram matrix $\mathbf{K} = \mathbf{X}\mathbf{X}^T$, as the Equation 2.22 holds no closed formula. The objective is then transformed as $\| \mathbf{X}\mathbf{X}^T - \mathbf{Y}\mathbf{Y}^T \|^2$. After running singular value decomposition of the $\mathbf{X}$, we have $\mathbf{X} = \hat{\mathbf{U}}\mathbf{E}\hat{\mathbf{V}}^T$. The Gram matrix $\mathbf{K}$ is then expressed as $\hat{\mathbf{U}}\mathbf{E}^2\hat{\mathbf{U}}^T$. Obviously, the solution of the least squared approximation to $\mathbf{K}$ via $\mathbf{Y} \in \mathbb{R}^{n \times L}$ comes from $\hat{\mathbf{U}}\hat{\mathbf{E}}$. Recalling that $\mathbf{X}\hat{\mathbf{V}} = \hat{\mathbf{U}}\hat{\mathbf{E}}$, we could have $\mathbf{Y} = \mathbf{X}\mathbf{V}$, where $\mathbf{V}$ denotes the first L vectors of $\hat{\mathbf{U}}$. Finally, the conclusion is that the classic MDS is equivalent to PCA.

2.2.3 Extreme Learning Machine Auto-Encoder

Auto-Encoder (AE) [17, 63, 71–76] can perform dimension reduction or dimension expansion for original input data. AE is a 'specialized' network structure whose output is the same as the input. Single hidden layer AE is the most simplified AE network structure, which uses the input mapping as the encoder and the output mapping as the decoder. The AE can be conveniently extended into deep neural networks, such as Variational Auto-Encoder (VAE) [63], with two subnetworks for encoder and decoder, respectively. Tied weight Auto-Encoder (TAE) [17] shows that the input weights and output weights can be shared. Depending on the size of hidden neurons, TAE can perform dimension reduction, equal dimension projection, and feature expansion. Extreme Learning Machine Auto-Encoder (ELM-AE) [23, 24] follows the same structure as TAE. Nevertheless, the input weights are randomly generated, originated from ELM.

To be more precise, ELM-AE can be categorized into linear ELM-AE and nonlinear ELM-AE depending on the activation function. Meanwhile, ELM-AE can be linear Sparse ELM-AE (SELM-AE) or nonlinear SELM-AE conditional on the type of random weights. Furthermore, ELM-AE can perform dimension reduction ($L < d$), equal dimension projection ($L = d$), and feature expansion ($L > d$).

For compressed structure ($L < d$), the input is projected onto lower-dimensional ELM feature space via the orthogonal random matrix, which is calculated as follows:

$$\begin{aligned} \mathbf{H}(\mathbf{X}) &= g(\mathbf{X}\mathbf{A} + \mathbf{b}) = [g(\mathbf{X}\mathbf{a}_1 + b_1), \dots, g(\mathbf{X}\mathbf{a}_L + b_L)], \\ \mathbf{A}^T \mathbf{A} &= \mathbf{I}, \\ \mathbf{b}^T \mathbf{b} &= 1. \end{aligned} \tag{2.23}$$

The input weights or biases are orthogonal matrices or vectors, which are different from standard ELM. Vincent et al. [77] proposed that the hidden layer should preserve the information of input data. According to Johnson-Lindenstrauss

Lemma [78], orthogonal random projection can retain the Euclidean distance of the input data. Thus, the objective of general ELM-AE follows:

$$\text{Minimize : } \|\mathbf{H}\boldsymbol{\beta} - \mathbf{X}\|^2 + C \|\boldsymbol{\beta}\|^2, \quad (2.24)$$

where C indicates the ℓ_2 -regularization item. According to Equation 2.7, the solution is changed as below:

$$\boldsymbol{\beta} = (C\mathbf{I} + \mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T \mathbf{X}. \quad (2.25)$$

After the training procedure, ELM-AE projects the data onto a more generalized feature space via the following transformation:

$$\mathbf{X}^{proj} = f(\mathbf{X}\boldsymbol{\beta}^T), \quad (2.26)$$

where $f(\cdot)$ denotes activation function, which can be sigmoid or tanh function.

The linear ELM-AE differs at two points: the activation function $g(\cdot)$ is linear and the biases $\mathbf{b}$ are zeros. Meanwhile, the solution of linear ELM-AE should be changed as follows, according to the Theorem 2 of [24].

$$\boldsymbol{\beta} = \mathbf{A}^T \mathbf{V} \mathbf{V}^T, \quad (2.27)$$

where $\mathbf{V}$ is the set of eigenvectors of covariance matrix $\mathbf{X}^T \mathbf{X}$.

According to the statement of linear ELM-AE, the solution is verified by Ding et al.[79], which shows that projecting input along between-class scatter matrix $\mathbf{X} \mathbf{M} \mathbf{M}^T$ ($\mathbf{M} = [\mathbf{m}_1, \dots, \mathbf{m}_t]$ and $\mathbf{m}_i$ is the center vector of class i) reduces the distances of samples from the same cluster. Therefore, linear ELM-AE projects data along the between-class scatter matrix multiplied by the orthogonal matrix, and that is $\mathbf{X} \mathbf{V} \mathbf{V}^T \mathbf{A}$.

The SELM-AE comes from the motivation of sparse coding [80]. Similar to the orthogonal random projection $\mathbf{X}\mathbf{A}$ of general ELM-AE, the sparse random matrix also preserves the Euclidean distance between data points. Accordingly, the hidden activation matrix $\mathbf{H}$ is calculated as follows and different from 2.23.

$$\mathbf{H}(\mathbf{X}) = g(\mathbf{X}\mathbf{A} + \mathbf{b}) = [g(\mathbf{X} \cdot \mathbf{a}_1 + b_1), \dots, g(\mathbf{X} \cdot \mathbf{a}_L + b_L)],$$

$$a_{ij} = b_i = 1/\sqrt{L} \begin{cases} +\sqrt{3}, & p = 1/6, \\ 0, & p = 2/3, \\ -\sqrt{3}, & p = 1/6, \end{cases} \quad (2.28)$$

where $\mathbf{A}$ is the sparse random matrix and $\mathbf{b}$ is the bias vector. The output weights β can be computed by objective 2.24 or Equation 2.27.

For the feature expansion case ($L > d$), the ELM-AE can also be efficiently calculated as objective 2.24 with relaxing the generation of orthogonal random matrix $\mathbf{A}$. ELM-AE has inspired the following works [36, 37, 81–84]. We mainly focus on unsupervised feature learning in this thesis.

2.2.4 Sparse Auto-Encoder

Sparse Auto-Encoder (SAE) [62], which has a similar network structure with ELM-AE, applies back-propagation to learn unknown weights. Given the training set $\mathbf{X} \in \mathbb{R}^{n \times d}$, where n and d denote the number of samples and the data dimension respectively, SAE tries to learn a function $h_{\mathbf{W}, \mathbf{b}}(x) \approx x$. The $\mathbf{W}$ represents the weights set, where W_{ij}^l denotes the parameter associated with the connection between i -th neuron in the layer l , and j -th neuron in the layer $l + 1$. Moreover, b is the bias term. The network structure is illustrated in Figure 2.3.

The overall cost function is:

FIGURE 2.3: Illustration of the SAE's network structure.

$$J(\mathbf{W}, \mathbf{b}) = \frac{1}{n} \sum_{i=1}^n \frac{1}{2} \| h_{\mathbf{W}, \mathbf{b}}(x^i) - x^i \|^2 + \frac{\lambda}{2} \sum_{l=1}^{n_l-1} \sum_i \sum_j (W_{ij}^l)^2, \quad (2.29)$$

where n_l represents the number of neurons in layer l .

The first term of $J(\mathbf{W}, \mathbf{b})$ is the average sum-of-squares reconstruction error, and the second term is the regularization term. The λ is the weight decay hyper-parameter. Although the cost function shares similarity with Equation 2.24, it learns the parameters with back-propagation.

SAE treats a neuron as active if its output is close to one, and inactive while its output is close to zero. SAE encourages more neurons to be inactive with the following sparsity constraint. Therefore, the weight decay term in Equation 2.29 would be replaced. Firstly, let

$$\hat{\rho}_j = \frac{1}{n} \sum_{i=1}^n [a_j^l(x_i)] \quad (2.30)$$

be the average activation of j -th neuron in the layer l . SAE enforces the constraint

$$\hat{\rho}_j = \rho, \quad (2.31)$$

where ρ is the sparsity parameter. Generally ρ is a small value close to zero, such as 0.05.

Then the sparsity constraint is

$$\sum_{j=1} \rho \log \frac{\rho}{\hat{\rho}_j} + (1 - \rho) \log \frac{1 - \rho}{1 - \hat{\rho}_j}. \quad (2.32)$$

Then the overall cost function of SAE is

$$J(\mathbf{W}, \mathbf{b}) = \frac{1}{n} \sum_{i=1}^n \frac{1}{2} \| h_{\mathbf{W}, \mathbf{b}}(x^i) - x^i \|^2 + \sum_{j=1} [\rho \log \frac{\rho}{\hat{\rho}_j} + (1 - \rho) \log \frac{1 - \rho}{1 - \hat{\rho}_j}]. \quad (2.33)$$

As the Objective 2.33 has no analytical solution, SAE is typically solved by back-propagation.

2.2.5 Variational Auto-Encoder

Variational Auto-Encoder (VAE) [63] proposes a stochastic variational inference and learning algorithm, which assumes the data are generated by the latent variable $\mathbf{z}$. The data generation process consists of two steps: 1) sampling a $\mathbf{z}_i$ from prior distribution $p_{\theta}(\mathbf{z})$; 2) generating a x^i based on the conditional distribution $p_{\theta}(\mathbf{x}|\mathbf{z})$, and the parameters θ are unknown. VAE makes no assumptions about the marginal and posterior probabilities that the $p_{\theta}(\mathbf{z}|\mathbf{x}) = p_{\theta|\mathbf{z}}(\mathbf{x})p_{\theta}(\mathbf{z})/p_{\theta}(\mathbf{x})$ is intractable.

VAE proposes a solution to efficient approximate maximal likelihood estimation for the parameters θ and posterior distribution of the latent variable $\mathbf{z}$ given $\mathbf{x}$. First of all, it introduces the recognition model $q_{\phi}(\mathbf{z}|\mathbf{x})$ to approximate true posterior $p_{\theta}(\mathbf{z}|\mathbf{x})$. We refer to the process $p_{\phi}(\mathbf{z}|\mathbf{x})$ as the encoder and the $p_{\theta}(\mathbf{x}|\mathbf{z})$ as decoder. Typically, the encoding term is a multivariate Gaussian distribution, and the decoding term follows Gaussian or Bernoulli distribution.

The variational bound is derived from a sum over the marginal likelihoods of individual samples, $\log p_{\theta}(\mathbf{x}_1, \dots, \mathbf{x}_n)$, which can also be written as:

$$\log p_{\theta}(\mathbf{x}_i) = D_{KL}(q_{\phi}(\mathbf{z}|\mathbf{x}_i)||p_{\theta}(\mathbf{z}|\mathbf{x}_i)) + \mathbb{E}_{\mathbf{z} \sim q_{\phi}(\mathbf{z}|\mathbf{x})} \log \frac{p_{\theta}(\mathbf{x}, \mathbf{z})}{q_{\phi}(\mathbf{z}|\mathbf{x})}, \quad (2.34)$$

where the first RHS term is non-negative KL-divergence of the approximate to the true posterior, and the second RHS term is the so-called variational lower bound on the marginal likelihood of $\mathbf{x}_i$, which can be written as:

$$\mathbb{L}(\mathbf{x}_i) = -D_{KL}(q_{\phi}(\mathbf{z}|\mathbf{x}_i)||p_{\theta}(\mathbf{z})) + \mathbb{E}_{\mathbf{z} \sim q_{\phi}(\mathbf{z}|\mathbf{x}_i)} \log p_{\theta}(\mathbf{x}_i|\mathbf{z}). \quad (2.35)$$

However, the gradient of $\mathbb{L}(\mathbf{x}_i)$ w.r.t parameters ϕ is not straightforward. Thus, the VAE introduces stochastic gradient variational Bayes estimator $\tilde{\mathbb{L}}(\mathbf{x}_i) \approx \mathbb{L}(\mathbf{x}_i)$:

$$\tilde{\mathbb{L}}(\mathbf{x}_i) = \frac{1}{L} \sum_{l=1}^L \log p_{\theta}(\mathbf{x}_i, \mathbf{z}_{i,l}) - \log q_{\phi}(\mathbf{z}_{i,l}|\mathbf{x}_i), \quad (2.36)$$

where $\mathbf{z}_{i,l} = g_{\phi}(\epsilon_{i,l}, \mathbf{x}_i)$, $\epsilon_l \sim p(\epsilon)$. The function $g_{\phi}(\cdot)$ is a differentiable transformation to reparameterize the distribution $\mathbf{z} \sim q_{\phi}(\mathbf{z}|\mathbf{x})$ and ϵ is auxiliary noise.

Generally, the q_{ϕ} follows a multivariate Gaussian with a diagonal covariance structure. And the $p_{\theta}(\mathbf{z})$ is the centered isotropic multivariate Gaussian. Therefore the formula of Equation 2.36 is changed to:

$$\tilde{\mathbb{L}}(\mathbf{x}_i) \approx \frac{1}{2} \sum_{j=1}^J [1 + \log(\delta_i^j)^2 - (\mu_i^j)^2 - (\delta_i^j)^2] + \frac{1}{L} \sum_{l=1}^L \log p_{\theta}(\mathbf{x}_i|\mathbf{z}_{i,l}), \quad (2.37)$$

where $\mathbf{z}_{i,l} = \boldsymbol{\mu}_i + \boldsymbol{\delta}_i \odot \boldsymbol{\epsilon}_l$, $\boldsymbol{\epsilon}_l \sim \mathbb{N}(0, \mathbf{I})$, J denotes the dimensionality of $\mathbf{z}$, and L represents the sampling size per datapoint $\mathbf{x}_i$.

After learning with BP methods, the q_{ϕ} could perform the encoding role. A standard pipeline for VAE is illustrated in Figure 2.4.

FIGURE 2.4: Illustration of a standard VAE structure.

2.3 Unsupervised Feature Learning-based Multi-Layer Extreme Learning Machine Structure

Extending ELM into the unsupervised feature learning scenario has received attention and contributions. Among the efforts, building a multi-layer ELM feature learning structure can be categorized into two directions based on the connection type between layers. Deep Extreme Learning Machine [85], Multi-Layer Extreme Learning Machine [23] and Hierarchical Extreme Learning Machine [81] all use fully connected weights while Local Receptive Fields-based Extreme Learning Machine (LRF-ELM) [26], Extreme Learning Machine Network (ELMNet) [36] and Hierarchical Extreme Learning Machine Network (H-ELMNet) [37] emphasize the importance of local connection, especially on image-related tasks. The details are reviewed in the following subsections.

2.3.1 Deep Extreme Learning Machine

ELM is popular due to its simple network structure and efficiency for the extensions. While along with the increase in the volume of datasets, researchers focused on developing a deeper ELM structure. Deep Extreme Learning Machine [85] was proposed with a stacked ELMs structure, which reducing the number of hidden neurons compared to ELM and presenting better generalization capability. Deep ELM trains the overall network with a layer-wise method. The illustration starts from the first layer:

$$\mathbf{H}^1 = \text{sigmoid}(\mathbf{X}\mathbf{A}^1), \quad (2.38)$$

where $\mathbf{A}^1$ represents the first hidden weights and is generated randomly. The following second hidden weights β^2 are calculated as follows:

$$\beta^2 = (C^1\mathbf{I} + (\mathbf{H}^1)^T\mathbf{H}^1)^{-1}(\mathbf{H}^1)^T\mathbf{X}. \quad (2.39)$$

FIGURE 2.5: Illustration of network structure of Deep ELM. The bottom symbols represent the outputs of corresponding neurons.

Note the solution matches the objective of nonlinear ELM-AE, while Deep ELM utilizes β^2 with different manner. To be more precise, it multiplies $\mathbf{H}^1$ with β^2 rather than transforms $\mathbf{X}$ with the transpose of β^2 . This procedure is illustrated

FIGURE 2.6: A simpler inference structure of Deep ELM, as Deep ELM applies no activation on $\mathbf{H}^1 \beta^2$.

as below:

$$\mathbf{H}^3 = \text{sigmoid}(\mathbf{H}^1 \beta^2 \mathbf{A}^3), \quad (2.40)$$

where $\mathbf{A}^3$ denotes the hidden weights of the third layer, which is also randomly generated.

The last hidden layer, connected to output nodes, performs the regression or classification tasks. Unknown weights, presented with β^4 , can be learned as follows:

$$\beta^4 = (C^3 \mathbf{I} + (\mathbf{H}^3)^T \mathbf{H}^3)^{-1} (\mathbf{H}^3)^T \mathbf{Y}, \quad (2.41)$$

where $\mathbf{Y}$ is the output target.

The full network of Deep ELM during the learning procedure is illustrated in Figure 2.5, while the feedforward structure can be simplified for inference, as shown in Figure 2.6.

2.3.2 Multi-Layer Extreme Learning Machines

Based on the ELM-AE, Multi-Layer Extreme Learning Machine (ML-ELM) [23] proposed a deeper network structure via stacking multiple ELM-AEs, as illustrated

FIGURE 2.7: Illustration of the network of ML-ELM with two stacked ELM-AEs.

in Figure 2.7. Here original input data is denoted with the symbol $\mathbf{X}^1$. Then the transformed data into the second ELM-AE is represented by $\mathbf{X}^2$, which comes from $\mathbf{X}^2 = g(\mathbf{X}^1[\beta^1]^T)$. The output weights β^1 are computed following Equation 2.25. The activation function $g(\cdot)$ represents the linear, sigmoid, or tanh function.

Then, $\mathbf{X}^2 \in \mathbb{R}^{n \times L}$ acts as the input data for the next ELM-AE. After learning the second output weights β^2 via Equation 2.25 again, $\mathbf{X}^3$ is calculated with $\mathbf{X}^3 = g(\mathbf{X}^2[\beta^2]^T)$. Given a pre-defined network structure $[L_1, \dots, L_k]$, where L_i denotes the number of hidden nodes of i -th ELM-AE, k different ELM-AEs are trained sequentially to get final output feature $\mathbf{X}^k$, where $\mathbf{X}^k = g(\mathbf{X}^{k-1}[\beta^{k-1}]^T)$. After the last auto-encoder, the ridge regression is applied for the training data pairs $(\mathbf{X}^k, \mathbf{T})$.

Based on ML-ELM, Hierarchical ELM (H-ELM) [81] was proposed accordingly with two main differences: 1) adopting ℓ_1 -regularization rather than ℓ_2 -regularization. 2) using ELM classifier with L_k hidden nodes for $\mathbf{X}^{k-1}$ while ML-ELM applies a linear regression for $\mathbf{X}^k$.

2.3.3 Local Receptive Fields-based Extreme Learning Machine

Local Receptive Fields-based Extreme Learning Machine (LRF-ELM) [25, 26] addresses the question: *Can local receptive fields be applied in ELM?*. The thesis has mainly reviewed the works about ELMs with the fully connected input layer. In image-related applications, one hidden node should benefit from a local connection rather than a full connection with all input pixels. That is consistent with CNNs [86], while the difference is that the convolutional kernels of LRF-ELM can be randomly generated without learning. LRF-ELM emphasizes the importance of orthogonalization of the input matrix $\mathbf{A}$, which can extract a complete set of features (it is also verified in [23, 34]).

The detailed steps of generating $\mathbf{A}$ can be enumerated as below:

- (1) Given the receptive size $k \times k$, generate initial random matrix $\hat{\mathbf{A}} \in \mathbb{R}^{k^2 \times k^2}$. Each element is sampled from Gaussian or uniform distribution.
- (2) Applying singular value decomposition on $\hat{\mathbf{A}}$, we get $\mathbf{U}$, $\mathbf{\Sigma}$, and $\mathbf{V}$. Next, the first L columns of $\mathbf{U}$ are selected to form final $\mathbf{A} \in \mathbb{R}^{k^2 \times L}$.
- (3) Each column $\mathbf{a}_i$ of $\mathbf{A}$ represents a convolutional kernel. The $\mathbf{a}_i$ accepts local receptive window $k \times k$ and produces the i -th feature map.

The r -th feature map $\mathbf{c}_r$ is computed via the r -th column, $\mathbf{a}_r \in \mathbb{R}^{k^2 \times 1}$, of $\mathbf{A}$. Practically, $\mathbf{a}_r$ is reshaped into two-dimensional convolutional kernel $\bar{\mathbf{a}}_r \in \mathbb{R}^{k \times k}$. After that, $c_{i,j,r}$ is calculated as below Equation, where $*$ denotes the convolutional operation:

$$\mathbf{c}_r = \mathbf{x} * \bar{\mathbf{a}}_r. \quad (2.42)$$

Hence, LRF-ELM produces a $(d - k + 1) \times (d - k + 1)$ feature map without padding. The square/square-root pooling is applied on feature map to achieve

nonlinearity and generalization. The pooling output value, $h_{i,j,r}$, at position (i, j) in the r -th feature map, is calculated as below:

$$h_{i,j,r} = \sqrt{\sum_{p=i-e/2}^{i+e/2} \sum_{q=j-e/2}^{j+e/2} c_{p,q,r}^2}, \quad (2.43)$$

where e is the pooling window size.

The square/square-root has the properties of rectification nonlinearity and translation invariance [87, 88]. The final feature matrix $\mathbf{H}$ is shaped into proper shape; the solution 2.7 or 2.8 can be directly applied to learn unknown weights β .

Multi-Scale LRF-ELM (MSLRF-ELM) [89] and Multi-Modal LRF-ELM (MMLRF-ELM) [90] extend LRF-ELM to multi-scale and multi-modal scenarios, respectively. As the convolutional kernels are randomly generated that expanded feature maps impose limited influence on learning speed. The time-consuming part happens in the stage of training the ELM classifier. Furthermore, online sequential ELM [91] can be integrated into MSLRF-ELM and MMLRF-ELM to handle the memory problem and online requirement.

2.3.4 Non-Gradient Convolutional Neural Network

Principal Component Analysis Network (PCANet) [35] was proposed as a simple shallow neural network for unsupervised feature extraction. It shows competitive performance compared to supervised convolutional networks just with a linear SVM classifier on big datasets. Also, PCANet performs well as a BP-free method on small datasets. ELMNet [36] and H-ELMNet [37] were proposed following PCANet's network structure. Thus, the overall pipeline of PCANet, ELMNet, and H-ELMNet is referred to as Non-Gradient Convolutional Neural Network (NG-CNN) for convenience.

Based on the achievement of ELM-AE [23, 24], which shows ELM-AE outperforms PCA, NMF, and related methods on dimension reduction challenges,

ELMNet and H-ELMNet improve PCANet's performance mainly by replacing PCA with ELM-AE. ELMNet and H-ELMNet share a similar network structure with PCANet except for the ELM-AE variants. Generally, the NG-CNN pipeline can be summarized into three steps: pre-processing, filter learning, and post-processing. H-ELMNet adopts a complicated pre-processing step, including local contrast normalization (LCN) [32] and ZCA whitening for each convolutional layer, while PCANet and ELMNet only use simple patch-mean removal. All three methods apply two-layer channel-separable convolution (as shown in Figure 2.8) due to the feature dimension and memory limitation. These methods differ mainly in the filter convolution progress. H-ELMNet and ELMNet all use ELM-AEs as the filter learning method, while H-ELMNet takes a nonlinear ELM-AE variant, and ELMNet employs the linear case. Generally, nonlinear ELM-AE should outperform the linear across all datasets, while the experiment shows that a slight improvement of ELMNet on the MNIST dataset.

2.3.4.1 Principal Component Analysis Network

PCANet includes three main learning steps: pre-processing, filter learning, and post-processing. In the pre-processing stage, PCANet extracts image patches with sliding window size $k_1 \times k_2$, commonly k_1 equals k_2 . For image volume $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_i, \dots, \mathbf{x}_n]$, where $\mathbf{x}_i \in \mathbb{R}^{H \times W \times 1}$ represents each sample's shape and n denotes the number of samples, we have the accordingly extracted patches $\mathbf{P}_i = [\mathbf{p}^1, \dots, \mathbf{p}^j, \dots, \mathbf{p}^t]$, where $\mathbf{p}^j \in \mathbb{R}^{k_1 \times k_2}$ stands for one patch, and t is the number of patches. Within the rest of this thesis, we use k to denote k_1 and k_2 for simplification. After patch extraction, flatten patches are stacked into a two-dimensional matrix $\mathbf{M}$. Patch-mean removal is then applied to matrix $\mathbf{M}$ to eliminate the illumination effect.

The patch-mean removal for pre-processing is illustrated as:

$$\hat{\mathbf{p}}^i = \mathbf{p}^i - \frac{\sum_{j=1}^{k^2} \mathbf{p}_j^i}{k^2} \mathbf{1}, \quad (2.44)$$

FIGURE 2.8: Illustration of channel-separable convolution. There are two feature maps in level i , the same convolutional kernel is applied on each map to generate stacked outputs. Then the channels of feature maps increase exponentially.

where $\mathbf{1}$ stands for vector with all ones.

The patch-mean removed matrix $\hat{\mathbf{M}}$ is then formed by stacking $\hat{\mathbf{p}}^i$. This operation is performed once before each filter learning. Considering all NG-CNNs use two-layer filter convolution, patch-mean removal will be conducted two times.

After patch-mean removal, PCANet steps into filter learning and filter convolution. Filters are calculated by the PCA dimension reduction method. The stacked patch matrix $\mathbf{M}$ (simplification for $\hat{\mathbf{M}}$) follows matrix shape $k^2 \times t$, and k^2 represents the length of one flatten patch. PCA learns the orthogonal projection matrix $\mathbf{V}$ from covariance matrix $\mathbf{M}^T \mathbf{M}$ based on the knowledge that the first column of $\mathbf{V}$ describes the most variance direction of $\mathbf{M}$, the second column represents the second most variance direction, and so on. This progress can be simply expressed as:

$$[\mathbf{E}, \mathbf{V}^\S] = \text{pca}(\mathbf{M}), \quad (2.45)$$

where $\text{pca}(\cdot)$ represents the PCA learning function as shown in Chapter 2.2.2, $\mathbf{E} = [e_1, \dots, e_{k^2}]$ denotes the eigenvalues, and $\mathbf{V}^\S = [\mathbf{v}_1, \dots, \mathbf{v}_{k^2}]$ describes the corresponding eigenvectors, respectively. Assuming the reduced dimension is set to L , the selected top L eigenvectors, according to top L eigenvalues, are combined

to form the projection matrix $\mathbf{V} = [\mathbf{v}_1, \dots, \mathbf{v}_L]$. Then each column in $\mathbf{V}$ is treated as one convolutional kernel. Suppose there is only one input channel, the convolutional output channels are determined by hyper-parameter L . Each column $\mathbf{v}_j$ produces one output feature map.

$$\mathbf{M}^{proj} = \mathbf{M}\mathbf{V}. \quad (2.46)$$

After 2.46, all image patches are projected to L -dimensional feature space by a tensor re-organization. If there are L input channels, then the resulting channels are $L \times L$, which is different from the usual convolution. In this thesis, it is called channel-separable convolution, as illustrated in Figure 2.8. The main disadvantage of channel-separable convolution is the exponential memory explosion problem. For example, the output channels will be L^3 while going into third filter convolution.

Since the reduced feature dimension must be less than k^2 , there is a condition that the output channels should be less than pixel numbers within the local receptive field. Generally, PCANet sets k to 7, and the output channel L is set to 8.

Due to the limitation of memory, all NG-CNNs implement a two-stage channel-separable convolutional network structure. The post-processing stage, consisting of binarization and block-wise histogram, is applied to the feature maps from the second stage. Assuming the number of filters in the first and second channel-separable convolution stages are L_1 and L_2 respectively, input feature maps $\mathbf{I}$ for post-processing step are denoted as:

$$\{[\mathbf{I}_1^1, \dots, \mathbf{I}_1^{L_2}], [\mathbf{I}_2^1, \dots, \mathbf{I}_2^{L_2}], \dots, [\mathbf{I}_{L_1}^1, \dots, \mathbf{I}_{L_1}^{L_2}]\}. \quad (2.47)$$

The element-wise unit step function $S(\cdot)$ is adopted to binarize $\mathbf{I}$ and obtain the binary feature maps $\mathbf{B}$, where unit step function $S(\cdot)$ denotes $S(v) = 1$ when $v \geq 0$, and $S(v) = 0$ otherwise. After binarization, the resulted feature maps are

shown as:

$$\{[\mathbf{B}_1^1, \dots, \mathbf{B}_1^{L2}], [\mathbf{B}_2^1, \dots, \mathbf{B}_2^{L2}], \dots, [\mathbf{B}_{L1}^1, \dots, \mathbf{B}_{L1}^{L2}]\}. \quad (2.48)$$

Next, the block-wise histogram feature encoding is applied. The instance on the first binary feature map group $\mathbf{B}_1 \in \{0, 1\}^{H \times W \times L2}$ demonstrates the example. Each depth-wise vector $\mathbf{B}_{1,i,j,-}$ from $\mathbf{B}_1$ is treated as an $L2$ -bit binary number. A simple hash function is employed to convert $\mathbf{B}_{1,i,j,-}$ to a decimal number as Equation 2.49:

$$\mathbf{D}_{1,i,j} = \sum_{r=1}^{L2} 2^{L2-r} \mathbf{B}_{1,i,j,r}. \quad (2.49)$$

After carrying the same operations on each feature map group, binary feature maps $\mathbf{B} \in \{0, 1\}^{L1 \times H \times W \times L2}$ are converted to three-dimensional decimal feature maps $\mathbf{D} \in \mathbb{R}_+^{L1 \times H \times W}$. The latter takes advantage of lower memory requirement and better robustness.

The block-wise histogram is computed as the last step for unsupervised feature extraction. A sliding window with shape $h \times w$ and step size s goes through all feature maps in $\mathbf{D}$ to extract patches $\mathbf{P}_D$. For each patch $\mathbf{P}_D^{i,j}$, corresponding histogram $\mathbf{H}_i^j$ is estimated with pre-defined bins b , we set b equal to 2^L for simplification. Considering the number of decimal values in $\mathbf{P}_D^{i,j}$ is much less than 2^L ; thus, $\mathbf{H}_i$ is a sparse feature vector. The final representation $\mathbf{H}_i$ is formed by stacking all histograms $\mathbf{H}_i^j$.

As the length of $\mathbf{H}_i$ could be tens of thousands, determined by the image shape, L , s , and b , PCANet takes Support Vector Machine (SVM) with the linear kernel as the classifier for time efficiency. A simple illustration of the NG-CNN pipeline is shown in Figure 2.9.

2.3.4.2 Extreme Learning Machine Network

Extreme Learning Machine Network (ELMNet) [36] was proposed based on the pipeline of the PCANet. ELMNet highlights the property of orthogonal $\beta \in \mathbb{R}^{L \times d}$

FIGURE 2.9: Illustration of NG-CNN pipeline for one data sample from second filter convolution. In this figure, the resulted channels of the first filter convolution stage are two for simplification. Cs-convolution denotes channel-separable convolution, the post-processing stage consists of binarization and block-wise histogram. The pre-processing stage is not shown here. Symbol h and w show that the height and width of the sliding window for the histogram both are 2. All the outputs are concatenated together to form a final sparse representation.

($L < d$). That is $\beta\beta^T = \mathbf{I}$, which means the row vectors of β are orthonormal. Although ELMNet fails to present the analytical solution of orthogonal β , it uses linear ELM-AE 2.27 to achieve approximately orthogonal property. To be compatible with the abbreviation in the following section, ELM-AE^{Or} represents linear ELM-AE within the scope of ELMNet. Given the image patches $\mathbf{M}$, the output feature maps $\mathbf{M}^{proj}$ are calculated as follows:

$$\mathbf{M}^{proj} = \mathbf{M}\mathbf{V}\mathbf{V}^T\mathbf{A}. \quad (2.50)$$

Projecting $\mathbf{M}$ along the scatter matrix $\mathbf{V}\mathbf{V}^T$ can reduce the distances of samples from the same cluster, namely clustering patches with similar patterns. Then the orthogonal matrix $\mathbf{A}$ ($\mathbf{A}^T\mathbf{A} = \mathbf{I}$) performs dimension reduction. Except for the filtering learning method, ELMNet shares the same structure with PCANet.

2.3.4.3 Hierarchical Extreme Learning Machine Network

Hierarchical Extreme Learning Machine Network (H-ELMNet) [37] has three main differences: 1) using local contract normalization (LCN) [32] and whitening to preprocess the image patches, 2) adopting a nonlinear ELM-AE variant for filtering learning, 3) concatenating the feature map of first convolution layer to second convolution layer.

After extracting the image patches $\mathbf{M}$, H-ELMNet applies LCN first. The illustration of LCN on i -th patch is shown below:

$$\begin{aligned} \mathbf{Y}_j^i &= (\mathbf{M}_j^i - \frac{1}{k^2} \sum_{l=1}^{k^2} \mathbf{M}_l^i) / \\ &\sqrt{\frac{1}{k^2} \sum_{l=1}^{k^2} (\mathbf{M}_l^i - \frac{1}{k^2} \sum_{l=1}^{k^2} \mathbf{M}_l^i)^2 + c}, \\ j &= 1, \dots, k^2; i = 1, \dots, t, \end{aligned} \quad (2.51)$$

where c is constant for robustness.

With the output $\mathbf{Y}$ from LCN processing, the whitening operation is used successively:

$$\begin{aligned} [\mathbf{D}, \mathbf{U}] &= eig(\mathbf{Y}^T \mathbf{Y}), \\ \mathbf{Z}^i &= \mathbf{U}(\mathbf{D} + diag(\epsilon))^{-1/2} \mathbf{U}^T (\mathbf{Y}^i)^T, \end{aligned} \quad (2.52)$$

where $eig(\cdot)$ denotes the eigen-decomposition function, $\mathbf{D}$ and $\mathbf{U}$ are eigenvalues and eigenvectors respectively, ϵ is set as 1. After LCN and whitening, the nonlinear ELM-AE is specialized as follows:

$$\begin{aligned} \mathbf{H}^\S &= g(\alpha_1(\mathbf{Z}\mathbf{A} + \mathbf{b})), \\ \boldsymbol{\beta} &= \alpha_2(C\mathbf{I} + (\mathbf{H}^\S)^T \mathbf{H}^\S)^{-1} (\mathbf{H}^\S)^T \mathbf{Z}, \end{aligned} \quad (2.53)$$

where α_1 and α_2 are two scaling factors. ELM-AE^{No} is short for this ELM-AE variant within the scope of H-ELMNet.

Same to PCANet and ELMNet, H-ELMNet uses $\boldsymbol{\beta}$ as the convolutional filters. H-ELMNet specially designs the direct link from the first feature maps to the second, as shown in 2.47, feeding into the post-processing step concurrently. Hence there are $L_1 \times (L_2 + 1)$ feature maps for post-processing.

2.3.4.4 Network Comparison of PCANet, ELMNet, H-ELMNet and LRF-ELM

The network framework comparison of PCANet, ELMNet, H-ELMNet, and LRF-ELM is shown in Figure 2.10. Note they all implement two-stage convolutional learning. PCANet proposes and designs the original NG-CNN structure, followed by ELMNet and H-ELMNet. ELMNet replaces the PCA filter learning method with an ELM-AE variant named ELM-AE^{Or} in this thesis. H-ELMNet utilizes another ELM-AE variant, called ELM-AE^{No}. It also introduces LCN and whitening

FIGURE 2.10: Framework comparison of PCANet, ELMNet, H-ELMNet, and LRF-ELM. CS-Convolution is short for channel-separable convolution. *ELM-AE^{No}* and *ELM-AE^{No}* represent two ELM auto-encoder variants. The former is orthogonal ELM-AE, and the latter is nonlinear case. The main differences are emphasized with bold font.

as the pre-processing method and designs a skip concatenation from feature map 1 to feature map 2.

LRF-ELM extends ELM classifier with local-receptive-field random projection. Especially it emphasizes the importance of orthogonal random projection. Although LRF-ELM also shows a two-stage channel-separable convolution, it is featured with a learning-free random kernel. A simple flatten operation to second feature maps forms the corresponding post-processing step. Thus, LRF-ELM achieves less-effective nonlinear learning capability compared with NG-CNNs. As the supplement, it relies on the ELM classifier rather than linear-SVM to realize nonlinear requirements.

Chapter 3

Unsupervised Feature Learning with Sparse Bayesian Auto-Encoding based Extreme Learning Machine

This chapter presents the effective ELM-AE variant based on the sparse Bayesian inference framework. Considering the learning efficiency, a parallel training scheme, referred to as batch-size training, is addressed.

3.1 Motivation

ELM-AE has inspired many exertions [82, 83, 92] for unsupervised feature learning. Nevertheless, the insight into its network structure is rarely discussed. Firstly, output weights and the transposed shape of a single auto-encoder are denoted as β and γ , respectively. Then corresponding column vectors are $[\beta_1, \dots, \beta_j, \dots, \beta_d]$ and $[\gamma_1, \dots, \gamma_i, \dots, \gamma_L]$. Generally, for single layer networks or ELM, β_j correlates all hidden nodes to the j -th output node and is independent to β_{-j} , where β_{-j} represents all columns except β_j . While for feature projection of Extreme Learning Machine Auto-Encoder (ELM-AE) [23, 24], γ_i links all data dimensions to the i -th hidden node. That is shown in Figure 3.1. H-ELM shows that sparsity of β with ℓ_1 -regularization can improve the effectiveness of γ , while in this chapter the sparse Bayesian learning is verified its effectiveness for unsupervised ELM auto-encoding.

While transferring the knowledge of sparse Bayesian Extreme Learning Machine (SB-ELM) [39] to ELM-AE, there are still two concerns. Firstly, the Bernoulli distribution in Equation 2.14 is not suitable for unsupervised feature learning with continuous targets. Secondly, Bayesian Extreme Learning Machine (B-ELM) [38] and SB-ELM both solve objective sequentially for each output dimension $\mathbf{y} \in \mathbb{R}^{n \times 1}$. Thus, it will be time-consuming if the output dimension is high. Accordingly, the sparse Bayesian ELM-AE (SB-ELM-AE) is addressed, which attempts to present effectiveness and improved efficiency. A network structure with stacked SB-ELM-AEs constructs the new unsupervised feature learning method, which is referred to as Sparse Bayesian Auto-Encoding-based ELM (SBAE-ELM). Evidence of pruning hidden nodes based on Bayesian estimation presents the upgraded result. Importantly, the advantage of pruning might be highlighted on learning efficiency. That is assuming we can drop d_n of L hidden nodes without sacrificing generalization and performance. The time-cost of training the following SB-ELM-AE is approximately reduced to $[(L - d_n)/L]\%$ of implementation without pruning. The technical details are discussed in Section 3.2.

FIGURE 3.1: The illustration shows the forward connection β_j and the backward γ_i . The β_j relates all hidden nodes with the j -th output node and is independent to β_{-j} . While for ELM-AE, the γ_i links all output dimensions with the i -th hidden node.

3.2 Sparse Bayesian Learning for Extreme Learning Machine Auto-Encoder

3.2.1 Single-Output Sparse Bayesian ELM-AE

In SB-ELM, the target outputs are discrete class labels modeled by Bernoulli distribution. On the other hand, ELM-AE requires that the outputs are as same as the inputs. Therefore, the outputs can be treated accordingly by continuous Gaussian distribution. Firstly, this section begins with single-output sparse Bayesian learning, in which the output $\mathbf{y} \in \mathbb{R}^{n \times 1}$ denotes one column vector of input $\mathbf{X} \in \mathbb{R}^{n \times d}$, such as $\mathbf{y} = \mathbf{X}_{:,j}$ ($\mathbf{X}_{:,j}$ is the j -th column).

The likelihood is expressed as Equation 3.1, assuming outputs follow independent Gaussian distribution with variance δ^{-1} same to Equation 2.11.

$$p(\mathbf{y}|\mathbf{H}, \boldsymbol{\beta}) = \prod_{i=1}^n \mathcal{N}(\mathbf{h}_i^T \boldsymbol{\beta}; \delta^{-1}), \quad (3.1)$$

where $\mathbf{h}_i$ represents the hidden activation of i -th sample, the output is $\mathbf{y} \in \mathbb{R}^{n \times 1}$, $\boldsymbol{\beta} \in \mathbb{R}^{L \times 1}$ connects the hidden activation $\mathbf{h}_i$ with the output $\mathbf{y}$.

Each individual dimension of $\boldsymbol{\beta}$ can be treated as an independent Gaussian sample. The Gaussian prior distribution over parameter $\boldsymbol{\beta}$ is given by Equation 3.2, where $\boldsymbol{\alpha} = [\alpha_1, \dots, \alpha_k, \dots, \alpha_L]$.

$$p(\boldsymbol{\beta}|\boldsymbol{\alpha}) = \prod_{k=1}^L \mathcal{N}(\beta_k|0; \alpha_k^{-1}). \quad (3.2)$$

Based on the definition of conditional likelihood and prior distribution, the core learning procedure is derived. From Bayesian theory, we learn that posterior over all unknown variables follows Equation 3.3.

$$p(\boldsymbol{\beta}, \boldsymbol{\alpha}|\mathbf{y}, \mathbf{H}) = \frac{p(\mathbf{y}|\boldsymbol{\beta}, \boldsymbol{\alpha}, \mathbf{H})p(\boldsymbol{\beta}, \boldsymbol{\alpha})}{p(\mathbf{y})}. \quad (3.3)$$

The posterior can not be computed directly as the integral $p(\mathbf{y}) = \int p(\mathbf{y}|\boldsymbol{\beta}, \boldsymbol{\alpha}) p(\boldsymbol{\beta}, \boldsymbol{\alpha}) d\boldsymbol{\beta} d\boldsymbol{\alpha}$ can not be estimated analytically. In practice, the inference can be estimated via calculating posterior $p(\boldsymbol{\beta}|\mathbf{y}, \boldsymbol{\alpha}, \mathbf{H})$. For easy access, the Laplace approximation method is adopted to approximate posterior over $\boldsymbol{\beta}$ with a Gaussian distribution, which is achieved by quadratic Taylor expansion of log-posterior [93]. Since $p(\boldsymbol{\beta}|\mathbf{y}, \boldsymbol{\alpha}) \propto p(\mathbf{y}|\boldsymbol{\beta})p(\boldsymbol{\beta}|\boldsymbol{\alpha})$, this is equivalent to maximize objective 3.4 over $\boldsymbol{\beta}$.

$$\begin{aligned}
& \ln\{p(\mathbf{y}|\boldsymbol{\beta})p(\boldsymbol{\beta}|\boldsymbol{\alpha})\} \\
&= \ln\left\{\prod_{i=1}^n \frac{1}{\sqrt{2\pi\delta^{-1}}} \exp\left\{-\frac{\delta(\mathbf{h}_i^T \boldsymbol{\beta} - y_i)^2}{2}\right\} \prod_{k=1}^L \frac{1}{\sqrt{2\pi\alpha_k^{-1}}} \exp\left\{-\frac{\alpha_k \beta_k^2}{2}\right\}\right\} \\
&= n \ln \frac{1}{\sqrt{2\pi\delta^{-1}}} + \sum_{k=1}^L \frac{1}{\sqrt{2\pi\alpha_k^{-1}}} + \sum_{i=1}^n \left\{-\frac{\delta(\mathbf{h}_i^T \boldsymbol{\beta} - y_i)^2}{2}\right\} \\
&\quad + \sum_{k=1}^L \left\{-\frac{\alpha_k \beta_k^2}{2}\right\} \\
&= \mathbf{const} - \frac{1}{2} \boldsymbol{\beta}^T \mathbf{A} \boldsymbol{\beta} - \frac{\delta}{2} (\mathbf{H} \boldsymbol{\beta} - \mathbf{y})^T (\mathbf{H} \boldsymbol{\beta} - \mathbf{y}),
\end{aligned} \tag{3.4}$$

where $\mathbf{A}$ denotes diagonal matrix with element $\mathbf{A}_{kk} = \alpha_k$. Note the item $\mathbf{const} = n \ln \frac{1}{\sqrt{2\pi\delta^{-1}}} + \sum_{k=1}^L \frac{1}{\sqrt{2\pi\alpha_k^{-1}}}$ in Equation 3.4 is irrelevant to $\boldsymbol{\beta}$.

According to the iteratively-reweighted least-squares' method (IRLS) [94], Laplace's mode $\hat{\boldsymbol{\beta}}$ of posterior can be computed efficiently. Accordingly, the gradient $\nabla_{\boldsymbol{\beta}}$ with respect to $\boldsymbol{\beta}$ and Hessian matrix $\boldsymbol{\phi}$ are the basis of mean and covariance matrix of approximated posterior, respectively. They are illustrated in Equation 3.5 and 3.6.

$$\begin{aligned}
\nabla_{\boldsymbol{\beta}} &= \nabla_{\boldsymbol{\beta}} \ln\{p(\mathbf{y}|\mathbf{H}, \boldsymbol{\beta})p(\boldsymbol{\beta}|\boldsymbol{\alpha})\} \\
&= -\mathbf{A}\boldsymbol{\beta} - \delta(\mathbf{H}^T \mathbf{H} \boldsymbol{\beta} - \mathbf{H}^T \mathbf{y}) \\
&= -\mathbf{A}\boldsymbol{\beta} - \delta \mathbf{H}^T (\mathbf{t} - \mathbf{y}),
\end{aligned} \tag{3.5}$$

$$\boldsymbol{\phi} = \nabla_{\boldsymbol{\beta}} \nabla_{\boldsymbol{\beta}} = -\mathbf{A} - \delta \mathbf{H}^T \mathbf{H}. \tag{3.6}$$

Using IRLS, mean $\hat{\beta}$ is updated from following equation.

$$\begin{aligned}
\beta^{new} &= \beta^{old} - \phi^{-1} \nabla_{\beta} = \beta^{old} + \phi^{-1} [\mathbf{A}\beta + \delta \mathbf{H}^T (\mathbf{H}\beta^{old} - \mathbf{y})] \\
&= \phi^{-1} [\phi \beta^{old} + \delta \mathbf{H}(\mathbf{t} - \mathbf{y}) + \mathbf{A}\beta^{old}] \\
&= \phi^{-1} [-\mathbf{A}\beta^{old} - \delta \mathbf{H}^T \mathbf{H}\beta^{old} + \mathbf{A}\beta^{old} + \delta \mathbf{H}^T (\mathbf{H}\beta^{old} - \mathbf{y})] \quad (3.7) \\
&= -\delta \phi^{-1} \mathbf{H}^T \mathbf{y} \\
&= [\mathbf{A} + \delta \mathbf{H}^T \mathbf{H}]^{-1} \delta \mathbf{H}^T \mathbf{y}.
\end{aligned}$$

Now the mean $\hat{\beta}$ and covariance Σ are shown from Equation 3.7.

$$\hat{\beta} = \delta \Sigma \mathbf{H}^T \mathbf{y}, \quad (3.8)$$

$$\Sigma = [\mathbf{A} + \delta \mathbf{H}^T \mathbf{H}]^{-1}. \quad (3.9)$$

Considering that $\mathbf{A}$ is the diagonal matrix of α and δ is the hyper-parameter of $p(\mathbf{y}|\mathbf{H}, \beta)$, if we set $\mathbf{A} = C \cdot \mathbf{I}$ and $\delta = 1$, the resulted solution is the same with ℓ_2 -regularized ELM.

One step further for estimating α , the hyper-parameter posterior over α can be represented by $p(\alpha|\mathbf{y}) \propto p(\mathbf{y}|\alpha)p(\alpha)$. Assuming α follows the uniform distribution, only $p(\mathbf{y}|\alpha)$ requires maximization. As shown in [95], the according log-marginal likelihood is computed as Equation 3.10.

$$\begin{aligned}
\mathcal{L}(\alpha) &= \ln p(\mathbf{y}|\alpha) \\
&= \frac{1}{2} \ln |\delta^{-1} \mathbf{I} + \mathbf{H} \mathbf{A}^{-1} \mathbf{H}^T| \\
&\quad - \frac{\mathbf{y}^T (\delta^{-1} \mathbf{I} + \mathbf{H} \mathbf{A}^{-1} \mathbf{H}^T)^{-1} \mathbf{y}}{2} \quad (3.10) \\
&\quad + \text{const.}
\end{aligned}$$

Let the differential of $\mathcal{L}(\boldsymbol{\alpha})$ with respect to $\boldsymbol{\alpha}$ equal to zero [95, 96], α_k is updated by Equation 3.11, where Σ_{kk} denotes k -th diagonal element of $\boldsymbol{\Sigma}$.

$$\alpha_k = \frac{1 - \alpha_k^{old} \Sigma_{kk}}{\hat{\beta}_k^2}. \quad (3.11)$$

Given initial values for δ and $\boldsymbol{\alpha}$, with Equation 3.8, 3.9, and 3.11, the $\boldsymbol{\beta}$ and $\boldsymbol{\alpha}$ could be updated iteratively. The $\boldsymbol{\beta}$ could converge within limited steps with a pre-defined gap value τ . A quick summary of single-output SB-ELM-AE is presented in Algorithm 1.

Algorithm 1 Sparse Bayesian learning for single-output ELM-AE

Input:

- The randomly projected feature $\mathbf{H}$;
- The target vector $\mathbf{y}$;
- The number of hidden nodes L ;
- The pre-defined δ and convergence factor τ ;
- The initialized $\boldsymbol{\alpha}$ vector and $\boldsymbol{\beta}$ vector.

Output:

- The estimated $\boldsymbol{\alpha}$, and output weights $\boldsymbol{\beta}$.

- 1: **repeat**
 - 2: Calculate covariance $\boldsymbol{\Sigma}$ according to Equation 3.9.
 - 3: Calculate new estimated mean $\boldsymbol{\beta}^{new}$ according to Equation 3.8.
 - 4: Estimate $\boldsymbol{\alpha}$ based on Equation 3.11.
 - 5: **until** $\|\boldsymbol{\beta}^{old} - \boldsymbol{\beta}^{new}\|_2^2 < \tau$, where $\boldsymbol{\beta}^{old}$ denotes $\boldsymbol{\beta}$ in previous iteration.
 - 6: Return $\boldsymbol{\alpha}$, $\boldsymbol{\beta}^{new}$ as $\boldsymbol{\beta}$.
-

For convenience, this thesis uses SB-ELM-AE as the abbreviation of sparse Bayesian ELM-AE in the whole thesis. Above SB-ELM-AE is introduced for the single-output case, in which output $\mathbf{y} \in \mathbb{R}^{n \times 1}$ is one column of $\mathbf{X} \in \mathbb{R}^{n \times d}$. If one repeats the above learning progress for each output dimension, the time efficiency of the proposed method can not be highlighted. Thus, a parallel implementation is proposed, called batch-size training for multi-output SB-ELM-AE, in the following subsection.

3.2.2 Batch-Size Training for Multi-Output Sparse Bayesian ELM-AE

The time-cost of training a single-output SB-ELM-AE mainly stems from Equations 3.8 and 3.9, the time complexity of which are $\mathcal{O}(n^2L^3)$ and $\mathcal{O}(L^3)$, respectively. The algorithm in the former subsection only presents the solution for the single-output SB-ELM-AE. The data dimension d of input requires d repeats of training single-output SB-ELM-AE to fulfill the overall objective. Then final time complexity would be d times of single-output SB-ELM-AE. Instead of simply repeating Algorithm 1 for multi-output SB-ELM-AE, multi-output learning is accelerated via parallel implementation. The proposed method, called batch-size training, can learn the objectives efficiently based on the following observation.

Firstly, we emphasize mathematical notation in case of misunderstanding scalar, vector, or matrix symbols. The n input samples are represented by $\mathbf{X} \in \mathbb{R}^{n \times d}$, and $\mathbf{H} \in \mathbb{R}^{n \times L}$ is the corresponding ELM activation matrix. In the previous chapters, the $\mathbf{y} \in \mathbb{R}^{n \times 1}$ denotes one column of input matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$, such as the j -th column $\mathbf{X}_{:,j}$. Then the $\boldsymbol{\beta}$ used in single-output SB-ELM-AE denotes one column vector. In comparison, the final output weights matrix $\bar{\boldsymbol{\beta}}$ of a complete SB-ELM-AE, follow the shape $[\boldsymbol{\beta}_1, \dots, \boldsymbol{\beta}_i, \dots, \boldsymbol{\beta}_d] \in \mathbb{R}^{L \times d}$, where d comes from $\mathbf{X} \in \mathbb{R}^{n \times d}$. Hence the $\boldsymbol{\beta}$ from Equation 2.10 to 3.11 represents one column vector of $\bar{\boldsymbol{\beta}}$, such as $\boldsymbol{\beta}_i$. Meanwhile, we take a similar notation to $\bar{\boldsymbol{\alpha}}$. To learn a complete SB-ELM-AE, individual vector $\boldsymbol{\alpha}_i$, $\boldsymbol{\beta}_i$, and matrix $\boldsymbol{\Sigma}_i$ for each output $\mathbf{X}_{:,i}$ should be calculated.

Note all elements in $\bar{\boldsymbol{\alpha}}$ are initialized with the same scalar value, such as 1. Thus, the first iteration of estimating all $\boldsymbol{\Sigma}_i$ follows time complexity $\mathcal{O}(L^3)$ instead of $\mathcal{O}(dL^3)$. Furthermore, $\boldsymbol{\Sigma}_i$, $\boldsymbol{\beta}_i$, and $\boldsymbol{\alpha}_i$ can be computed faster with batch-size matrix operations. Before the statement of how to implement parallel calculation, the batch-size matrix operations are introduced briefly.

Batch-size matrix operation follows a broadcasting rule ¹ of general cases.

¹www.numpy.org/devdocs/user/theory.broadcasting.html

For the mini-batch neural networks trained by the stochastic gradient descent, samples within one mini-batch share the same weights. All operations connecting individual sample with shared weights are broadly supported by toolboxes, such as tensorflow [97].

For example, given two 3-dimensional batch matrices $\mathbf{E} \in \mathbb{R}^{batch \times a \times b}$ and $\mathbf{F} \in \mathbb{R}^{batch \times b \times c}$, we get a matrix $\mathbf{G} \in \mathbb{R}^{batch \times a \times c}$ after batch-size multiplication. Each $\mathbf{G}_j \in \mathbb{R}^{a \times c}$ is the matrix multiplication result of $\mathbf{E}_j \in \mathbb{R}^{a \times b}$ and $\mathbf{F}_j \in \mathbb{R}^{b \times c}$. For convolutional neural networks, hyper-parameter *batch* denotes how many samples within one mini-batch, which is a compromise due to memory limitation. While in our case, the *batch* denotes the number of output dimensions within current batch-size training. Although the time complexity of Equation 3.8 is still $\mathcal{O}(n^2 L^3)$, it can be efficiently accelerated via large-scale tensor computation and supported by toolboxes (e.g. tensorflow [97]). That brings convenient implementation without explicit multi-threading coding or memory optimization.

Definitions of matrix operation symbols used in this thesis are defined first:

1. $\ominus$: Batch-size matrix subtraction.
2. $\otimes$: Batch-size matrix multiplication.
3. $\oslash$: Batch-size matrix division.
4. $\odot$: Element-size multiplication.

Then the difference from single-output SB-ELM-AE is illustrated in details. Extract a *batch* of output dimensions, e.g. $\mathbf{y} = \mathbf{X}_{:,1:batch}$, then transpose and reshape it to batch-first matrix $\check{\mathbf{Y}} \in \mathbb{R}^{batch \times n \times 1}$. Select according subset $\bar{\alpha}_{1:batch,1:L}$ and diagonalize second dimension to a three-dimensional matrix $\mathbf{A} \in \mathbb{R}^{batch \times L \times L}$. For simplification, δ is ignored. Based on the statement above, equations in Algorithm 1 can be re-written. The batch-size covariance matrix $\check{\Sigma}$ of Equation 3.9 is expressed as Equation 3.12, where *BI* denotes a batch-size matrix inverse operation

$$\check{\Sigma} = BI([\mathbf{A}_1 + \mathbf{H}^T \mathbf{H}]^{-1}, \dots, [\mathbf{A}_{batch} + \mathbf{H}^T \mathbf{H}]^{-1}). \quad (3.12)$$

FIGURE 3.2: Illustration of batch-size matrix operations and corresponding shapes. The upper presents Equation 3.13, which is the batch-size mean matrix $\tilde{\beta}$. The lower denotes Equation 3.14 for the calculation of batch-size prior variance matrix. The matrix shapes of $\tilde{\alpha}$, $\tilde{\Sigma}$, $\tilde{\beta}$, and $\tilde{Y}$ are also shown. The subscript i of $\tilde{\alpha}_i$, $\tilde{\Sigma}_i$, $\tilde{\beta}_i$, and $\tilde{Y}_i$ is ignored for simplification.

Algorithm 2 Batch-size parallel training for multi-output SB-ELM-AE**Input:**

- The randomly projected feature $\mathbf{H}$;
- The pre-defined δ and convergence factor τ ;
- The initialized $\bar{\alpha}$ matrix and $\bar{\beta}$ matrix;
- The hyper-parameter *batch*.

Output:

- The estimated $\bar{\alpha}$, and learned output weights $\bar{\beta}$.
- 1: Compute the first estimation of $\Sigma \in \mathbb{R}^{L \times L}$ with Equation 3.9 and use that for all batches.
- 2: **for** $i = 1$ to $\lceil L/batch \rceil$ **do**
- 3: Prepare batch-size output $\check{Y}_i$.
- 4: Compute the first estimation of batch-size $\check{\alpha}_i$ with Equation 3.14.
- 5: **repeat**
- 6: Calculate the batch-size covariance matrix $\check{\Sigma}_i$, according to Equation 3.12.
- 7: Calculate the new batch-size $\check{\beta}_i^{new}$ according to Equation 3.13.
- 8: Calculate the batch-size $\check{\alpha}_i$, according to Equation 3.14.
- 9: **until** $\|\check{\beta}_i^{old} - \check{\beta}_i^{new}\|^2 < \tau$, where $\check{\beta}_i^{old}$ denotes estimated $\check{\beta}_i$ in previous iteration.
- 10: **end for**
- 11: Stack all $\check{\alpha}_i$ and reshape as $\bar{\alpha}$.
- 12: Stack all $\check{\beta}_i$ and reshape as $\bar{\beta}$.
- 13: Return the transpose of $\bar{\alpha}$ and $\bar{\beta}$.

The estimated $\check{\beta}$ is then shown in Equation 3.13.

$$\check{\beta} = \check{\Sigma} \otimes \mathbf{H}_{dup}^T \otimes \check{Y}, \quad (3.13)$$

where $\mathbf{H}_{dup}^T \in \mathbb{R}^{batch \times L \times n}$ is created by stacking *batch* duplicates of $\mathbf{H}^T$.

Then we use Equation 3.14 to learn batch-size $\check{\alpha} \in \mathbb{R}^{batch \times L \times 1}$

$$\check{\alpha} = (\mathbf{1} \ominus \check{\alpha}^{old} \otimes \text{diag}(\check{\Sigma})) \oslash (\check{\beta} \odot \check{\beta}), \quad (3.14)$$

where $\text{diag}(\cdot)$ denotes the function to extract all diagonal vectors of $\check{\Sigma}$. $\check{\alpha}^{old}$ is the learned $\check{\alpha}$ in the previous iteration.

For easy access to batch-size operations, the data flow and matrix shape of

Equation 3.13 and 3.14 are illustrated in Figure 3.2. A brief summary of batch-size training is shown in Algorithm 2. Note that the batch-size training method sacrifices memory for the time-efficiency.

3.2.3 Hidden Nodes Selection

In this section, detailed steps to prune less important hidden nodes are introduced. ELM-AE calculates projected feature with transposed $\bar{\beta}$, which is $\bar{\beta}^T = \gamma = [\gamma_1, \dots, \gamma_L]$ as shown in Figure 3.1, that γ_j connects all output dimensions with the j -th hidden node. Thus, the output weights of ELM-AE perform differently from feedforward ELMs. We take the assumption that some hidden nodes may affect ELM-AE negatively.

To reduce the effect of the random projection matrix and extract the most important hidden nodes, the proposed method prunes the hidden neurons according to α_i in Equation 3.2. If α_i is close to a small value, a flat Gaussian distribution regularization is applied to β_i . If α_i is a large value, β_i tends to be zero and less important. This work assumes that pruning k -th hidden node with a low summation of $\bar{\alpha}_{:,k}$ could improve performance. Once $\bar{\alpha}$ is estimated, the importance of hidden nodes can be evaluated by $\bar{\alpha}_{:,+} \in \mathbb{R}^L$, which is the vector of summation along the second dimension of $\bar{\alpha}$. Sorting the $\bar{\alpha}_{:,+}$ from large to small, top d_n hidden nodes can be dropped to complete pruning. A quick summary of hidden nodes selection is shown in Algorithm 3.

Specifically, a pruning scheme is proposed not only for performance improvement, but also important for time efficiency. Assuming dropping d_n of L hidden nodes without sacrificing generalization and performance, the time-cost of training the following SB-ELM-AE is approximately reduced to $[(L - d_n)/L]\%$ of implementation without pruning.

Algorithm 3 Hidden nodes selection

Input:

- The estimated $\bar{\alpha}$ matrix and $\bar{\beta}$ matrix;
- The number of hidden nodes L ;
- The number of hidden nodes to drop d_n .

Output:

- The pruned output weights $\bar{\beta}$.
 - 1. Compute reduced summation of $\bar{\alpha}_{:,+}$.
 - 2. Select the d_n largest values of $\bar{\alpha}_{:,+}$, and record corresponding indexes.
 - 3. Drop corresponding rows in $\bar{\beta}$.
 - 4. Return updated $\bar{\beta}$.
-

3.3 Experiments

3.3.1 Experimental Setup

Experiments compared the performance of the proposed method SBAE-ELM with H-ELM, ML-ELM, PCA [14, 15], NMF [13], Variational Auto-Encoder (VAE) [98], and Sparse Auto-Encoder (SAE) [62] for unsupervised feature learning followed by ELM classifier.

For ML-ELM, H-ELM, and SBAE-ELM, the number of ELM-AE is hyper-parameter. Here we only consider 1 or 2 auto-encoders. Hyper-parameters $L1, L2, L3$ for ML-ELM, H-ELM, and SBAE-ELM denote first auto-encoder's hidden neurons, second auto-encoder and third classifier/auto-encoder, respectively. The hyper-parameter $L3$ is a special case for ML-ELM because ML-ELM uses the last auto-encoder for both feature auto-encoding and classification. The selection of the number of hidden nodes comes from set $[100, 200, \dots, 2000]$. The upper limit is set to 2000 for the consideration of both efficiency and effectiveness. Because the hyper-parameter combinations grow rapidly if increasing selections for each hyper-parameter, that means the implementation of ML-ELM, H-ELM, and SBAE-ELM will be inefficient. Besides, the experimental results for single ELM-AE along with hidden neurons from 500 to 8000 are illustrated in the following Section 3.3.3. Note that the performances of single ELM-AE gain marginal improvement when

TABLE 3.1: Datasets summary

Dataset	Features	Samples
Abalone	8	4,177
Madelon	500	2,600
Diabetes	8	768
Isolet	617	6,238
Jaffe	900	213
Orl	1,080	400
Fashion	784	70,000
Letters	784	145,600

the number of hidden neurons is large. Hence the upper limit is 2000 empirically. Regularization factor C is used in H-ELM and ML-ELM, while the former adopts ℓ_1 -regularization, and the latter uses ℓ_2 -regularization. The value of C is within $[1e-5, 1e-4, \dots, 1e4, 1e5]$. Hyper-parameter d_n only works for SBAE-ELM, which denotes the number of hidden nodes to prune, the range is $[0, 50, 100, 200, \dots, 2000]$. The experiments were conducted on the platform with Ubuntu-16.04 and a 12-core CPU. All codes are based on Python-2.7.

For SAE and VAE, the upper bound of training epochs is 40. SAE has same hyper-parameter space for the number of hidden neurons with SBAE-ELM. The sparse hyper-parameter was chosen from $[0.5, 0.05, 0.005]$. The encoder and decoder network of VAE are three-layer fully connected networks, the latent dimension choices of VAE were the same as PCA, which are $[5, 10, 20, 50, \dots, 500]$.

3.3.2 Datasets Preparation

The performance of the proposed method was evaluated on eight datasets. They are selected classification datasets from Openml ², the face emotion recognition dataset Jaffe [99], face recognition dataset Orl [100], image classification datasets Fashion [101] and Letters [102]. The summary of all datasets is listed in Table 3.1.

Openml: Abalone, Madelon, Diabetes, and Isolet datasets were chosen from Openml. They are non-image datasets, and flatten as vector per sample. Data

²www.openml.org

FIGURE 3.3: Samples of Jaffe dataset. For each row, faces from left to right express angry, disgusting, fear, happy, neutral, sad, and surprised emotion, respectively.

FIGURE 3.4: Samples of pre-processed Jaffe faces.

dimensions of Abalone and Diabetes are both 8. Samples of Madelon and Isolet have higher dimensions, which are 500 and 617, respectively. Data for SBAE-ELM, ML-ELM, H-ELM, and PCA normalized to have zero mean and unit variance, while normalized between 0 and 1 for NMF. The training and testing subsets were divided with proportion of 0.7 and 0.3, respectively. The partitions were repeated five times.

Jaffe: It contains seven types of facial emotions that are: angry, disgusting, fear, happy, neutral, sad, and surprised. Original examples are shown in Figure 3.3. In this thesis, face areas were cropped using a public face detector [103] with 68 registered landmarks and then normalized. While for NMF, samples were normalized between 0 and 1. Selected pre-processed faces are shown in Figure 3.4. The dataset was partitioned into the training set and testing set with proportions 0.7 and 0.3, respectively. And categories were equally split. Each sample is flattened into vector, and the same operation is applied for the following datasets.

Orl: It contains 40 subjects for face recognition. The officially cropped front face was directly used. Samples of Orl database are illustrated in Figure 3.5. It was randomly split into training and testing subset with amounts 240 and 160,

FIGURE 3.5: Samples of Orl dataset. The officially provided front faces were directly used.

respectively. The pixel values were scaled to $[0, 1]$ for NMF and normalized for other methods.

Fashion: It consists of a training set of 60,000 samples and a testing set of 10,000 samples. Each sample is 28×28 grayscale image and comes from 10 classes, such as trouser, shirt, and so on.

Letters: It is a bigger image dataset on character classification, which has 124,800 samples for training and 20,800 samples for testing. It contains 26 classes, and each class includes upper and lower cases. Data samples follow 28×28 grayscale shape.

3.3.3 Parameter Analysis

The effect of changing the number of dropping hidden neurons and the influence of the number of hidden neurons of ELM-AE are discussed in this section.

Firstly, the accuracy sensitivity to parameter d_n is analyzed. Details are presented in Figure 3.6. Two SB-ELM-AEs are utilized for Abalone, Madelon, Diabetes datasets. In contrast, a single SB-ELM-AE is suitable for Isolet, Jaffe, Orl, Fashion, and Letters datasets. The blue line represents the influence of varying d_n to the first SB-ELM-AE by fixing the second. Red line denotes sensitivity from the second SB-ELM-AE, as there is only one SB-ELM-AE for Isolet, Jaffe, Orl, Fashion, and Letters datasets that the red lines are therefore ignored. The experiments show that, for example, pruning hidden nodes can improve accuracy on Orl from 0.962 to 0.9675 by dropping 300 hidden nodes. There is a 0.89 percentage point

improvement on Jaffe via pruning 200 hidden nodes. Based on such observations on other datasets, we can conclude that dropping nodes can improve performance further.

The advantages of pruning hidden nodes also include time-efficiency while stacking multiple SB-ELM-AEs. For example, given the first SB-ELM-AE with 1000 hidden nodes, dropping 500 at least without sacrificing performance will reduce the time-cost of training the second SB-ELM-AE by half. For effectiveness and efficiency, the proposed pruning scheme is presented as a supplemental contribution.

Secondly, the effect of increasing the number of hidden nodes of the single ELM-AE was shown. Figure 3.7 illustrates the accuracy plotting of ℓ_2 -ELM-AE, ℓ_1 -ELM-AE, and SB-ELM-AE while increasing the hidden nodes from 500 to 8000. Note the curves are distinctive across datasets. For example, it approximates to be monotonic on Abalone and Madelon. Nevertheless, the curves show the apparent oscillation on other datasets. Although not all the best accuracies for ℓ_2 -, ℓ_1 -ELM-AE and SB-ELM-AE were reported with fewer hidden nodes, such as less than 2000. The choices of the number of hidden nodes were still limited to fewer than 2000 based on several considerations: 1) less hidden nodes can definitely bring implementation efficiency, 2) much hidden neurons of first ELM-AE will reduce the performance while feeding the transformed feature into the second ELM-AE, 3) the final searched structures and hyper-parameters by random grid search show better or equal performance compared to corresponding single ELM-AE reported in Figure 3.7, except for ML-ELM on Diabetes. Single l_2 -regularized ELM-AE with 8000 hidden nodes performs better than the reported stacked l_2 -ELM-AEs. Hence the number of hidden nodes was limited under 2000 based on the experiments and former motivations.

(a) Abalone

(b) Madelon

(c) Diabetes

(d) Isolet

(e) Jaffe

(f) Orl

(g) Fashion

(h) Letters

FIGURE 3.6: Illustration of the parameter influence of d_n on all benchmark datasets. Blue line represents accuracy influenced by the first auto-encoder, and red plotting denotes effect from the second auto-encoder. As there is only one SB-ELM-AE for Isolet, Jaffe, Orl, Fashion, and Letters dataset that the red lines are ignored.

(a) Abalone

(b) Madelon

(c) Diabetes

(d) Isolet

(e) Jaffe

(f) Orl

(g) Fashion

(h) Letters

FIGURE 3.7: Illustration of the influence of the number of hidden neurons on single ELM-AE. The experiments were conducted for each dataset. Note that the performances gain marginal improvement when the number of hidden neurons is larger than 2000. Considering the hyper-parameter searching and implementation efficiency, the maximal number of hidden neurons is set to 2000.

TABLE 3.2: Mean accuracy (%) comparison.

Dataset	NMF	PCA	ML-ELM	H-ELM	SAE	VAE	SBAE-ELM
Abalone	64.65	64.41	64.93	65.17	63.27	64.37	65.65
Madelon	59.61	52.46	60.46	61.23	59.95	53.76	64.37
Diabetes	74.28	71.29	76.12	77.04	76.34	72.52	78.01
Isolet	94.75	92.06	92.18	90.78	92.41	91.23	93.20
Jaffe	83.15	83.51	84.21	85.62	83.85	82.81	86.98
Orl	94.33	94.16	95.08	95.83	92.63	91.86	96.75
Fashion	86.04	87.51	87.53	87.36	87.42	86.83	87.57
Letters	81.87	86.77	88.25	87.97	86.81	90.14	88.32
<i>map</i>	79.82	79.02	81.09	81.38	80.33	79.19	82.61

3.3.4 Performance Comparison

The results are shown in Table 3.2. Accuracies were evaluated five repeats for each training/testing partition of each dataset. The best performance on each dataset is emphasized with boldface. For Openml datasets, results show the consistent improvement of SBAE-ELM compared with PCA, H-ELM, ML-ELM, SAE, and VAE. Except for Isolet, SBAE-ELM is also better than NMF. Table 3.2 also shows the accuracy on image datasets. One may observe that SBAE-ELM performs better than all the baseline methods NMF, PCA, ML-ELM, H-ELM, and SAE. On the Letters dataset, VAE presents the best performance. Nevertheless, the mean average performance (MAP) of VAE is the lowest. Obviously, SBAE-ELM achieves the best MAP across all datasets, which verified its generalization on scalable size dataset.

On Madelon, ELM-AE related methods have the most significant leading over PCA. Nevertheless, ML-ELM, H-ELM, and SBAE-ELM illustrate the smallest superiority compared to PCA on Abalone. H-ELM presents better results compared to ML-ELM on all datasets except Isolet. On Isolet, H-ELM performs the worst accuracy.

Selected hyper-parameters are listed in Table 3.3, where $L1(C)$ - $L2(C)$ - $L3(C)$ denotes hidden nodes and ℓ_1/ℓ_2 -regularization of first ELM-AE, hidden nodes, and ℓ_1/ℓ_2 -regularization of second ELM-AE, hidden nodes and ℓ_2 -regularization of last classifier/auto-encoder, respectively. Note that the *skip* in Table 3.3 means it only uses one ELM-AE. One ELM-AE for ML-ELM and H-ELM achieves the best performance on Madelon. And one ELM-AE is most effective for SBAE-ELM on Isolet, Jaffe, Orl, Fashion, and Letters. The hyper-parameters were chosen by two steps: 1) evaluating the performance of single ELM-AE to reduce the selections of hyper-parameters on each dataset individually, 2) finding the best combinations of hyper-parameters via random grid search based on a three-folder cross-validation on the training dataset. For the δ of SBAE-ELM, it is ten on Jaffe and Orl; other datasets use 1.

TABLE 3.3: Network structure and hyper-parameters

Dataset	ML-ELM	H-ELM	SBAE-ELM
	$L1(C)-L2(C)-L3(C)$	$L1(C)-L2(C)-L3(C)$	$L1(d_n)-L2(d_n)-L3(C)$
Abalone	$500(1e2)-500(1e2)-800(1)$	$600(1e4)-600(1e3)-800(1)$	$600(300)-600(300)-800(1)$
Madelon	$100(1)-skip-2000(1)$	$500(1e3)-skip-2000(1)$	$800(300)-600(300)-2000(1)$
Diabetes	$400(1e3)-400(1e2)-2000(1)$	$400(1e3)-400(1e2)-2000(1)$	$800(300)-400(100)-2000(1)$
Isotlet	$400(1e4)-500(1e3)-2000(1)$	$400(1e4)-500(1e3)-2000(1)$	$600(200)-skip-2000(1)$
Jaffe	$500(1e4)-500(1e3)-2000(1)$	$600(1e3)-500(1e3)-2000(1)$	$600(200)-skip-2000(1)$
Orl	$2000(1e2)-500(1e3)-2000(1)$	$1000(1e3)-500(1e3)-2000(1)$	$600(300)-skip-2000(1)$
Fashion	$500(1)-skip-10000(1)$	$1000(1e3)-skip-10000(1)$	$1000(100)-skip-10000(1)$
Letters	$500(1)-skip-10000(1)$	$500(1e3)-skip-10000(1)$	$1000(400)-skip-10000(1)$

3.3.5 Time Efficiency Improvement with Batch-Size Training

To validate the time-efficiency of batch-size training over single-output SB-ELM-AE, the time-cost of an SB-ELM-AE with 1000 hidden nodes was evaluated. Each dataset was estimated with five repeats. The batch-size is set to the maximum number according to the memory limitation.

Time spent (seconds) for all datasets are listed in Table 3.4 for comparison. The *ratio* column presents statistical time-efficient improvement. For example, *ratio* 2.14 of single-output training on Abalone represents it takes 1.14 more running time than batch-size training. One may say the time efficiency of batch-size training is achieved on all datasets compared with the single-output case.

The improvements *ratio* on Abalone and Diabetes are not as significant as on other datasets, because the input dimensions are only 8. While the *ratio* can exceed eight if the input dimension is larger than 500.

TABLE 3.4: Time-cost (seconds) comparison of single-output training and batch-size training.

Dataset	Single-output training	Batch-size training (<i>batch</i>)	Ratio
Abalone	1.95	0.91(8)	2.14
Madelon	270.74	20.17(100)	13.42
Diabetes	1.26	0.44(8)	2.86
Isolet	426.39	44.71(100)	9.53
Jaffe	382.87	23.06(100)	16.60
Orl	510.81	27.85(120)	18.34
Fashion	490.22	58.06(56)	8.44
Letters	1096.81	121.14(56)	9.05

3.4 Conclusions

In this chapter, the intrinsic feature of ELM-AE, which relates to multi-layer Extreme Learning Machine (ML-ELM) and hierarchical Extreme Learning Machine

(H-ELM), is analyzed. A sparse Bayesian learning-based ELM-AE (SB-ELM-AE) is proposed in this chapter. To overcome the time-inefficiency of multi-output SB-ELM-AE, a parallel implementation is addressed. Also, we show that pruning part of hidden nodes can improve performance further. Because ELM-AEs rely on the back-propagated connection for feature learning that redundant hidden neurons may reduce generalization. Experiments illustrate that the proposed method shows competitive performance than NMF, PCA, ML-ELM, H-ELM, SAE, and VAE for unsupervised feature learning.

Chapter 4

R-ELMNet: Regularized Extreme Learning Machine Network

Fully connected multi-layer ELMs present less competitive performance compared to local receptive fields based methods on image datasets. This chapter shows the improved ELM-AE, which is called R-ELM-AE, for the NG-CNN pipeline.

4.1 Motivation

As discussed in Section 2.3.4, Hierarchical Extreme Learning Machine Network (H-ELMNet) [37] and Extreme Learning Machine Network (ELMNet) [36] share most in their network structure except the ELM-AE selection. This chapter takes an insight into ELM-AE variants and discusses their effectiveness specifically within the Non-Gradient Convolutional Neural Network (NG-CNN) [35–37] framework.

ELM-AE [23, 24] was initially introduced as a non-gradient optimized auto-encoder to learn compact features. Experiments [24] show its improved performance on dimension reduction, especially compared with PCA, which is also the motivation of H-ELMNet and ELMNet. While they propose to utilize two ELM-AE variants, here we name ELM-AE^{No} and ELM-AE^{Or} for H-ELMNet and ELMNet variants, respectively. Details have been shown in Chapter 2.3.4.

The main drawback of nonlinear ELM-AE is the uncertainty of the value scale in the transformed feature space. As illustrated in Figure 4.1, experiment on Iris¹ dataset shows that the value range along the x-axis is approximately $[-10, 8]$ while the range along the y-axis is $[-0.3, 0.4]$. In the NG-CNN's structure, the x and y axes are corresponding to two resulted feature maps. Thus, the y map is close to 'dead' neurons compared to the x map. After running channel-separable convolution 2.8 by the same kernel on y map, the outputs contain less meaningful information. The binarization function $S(\cdot)$ could further eliminate information within the feature map if feature values distribute away from zero. Shi et al. [104] also show the margin unlikely holds the linearly separable chance with negatively expanded dimensions. This drawback is not noteworthy with a proper nonlinear classifier. In contrast, the binarization function $S(\cdot)$ within the NG-CNN pipeline might be the simplest linear mapping. To the best of our knowledge, the simple activation methods, such as tanh and sigmoid, fail to bring improvement. The related H-ELMNet and ELMNet take different solutions as below:

¹<https://www.openml.org/d/61>

FIGURE 4.1: The top figure presents the results of nonlinear ELM-AE for feature reduction. It was performed on the Iris dataset. Note that feature along the x-axis shows a much bigger value scale and variance compared with feature along the y-axis. The bottom figure shows the result of orthogonal ELM-AE. Although it achieves secondary linear-separability, values of each dimension keep comparable scale and variance.

The ELM-AE^{No} utilizes scale hyper-parameters, LCN, and whitening to improve the performance of nonlinear ELM-AE in NG-CNN. In contrast, relevant drawbacks include larger hyper-parameter space and a time-consuming pre-processing stage.

The ELM-AE^{Or} sacrifices the nonlinear learning capability and highlights the importance of orthogonal characteristic $\mathbf{A}^T \mathbf{A} = \mathbf{I}$. As shown in Figure 4.1, results show a better value range.

Beyond their limitations, the regularized ELM-AE is proposed, which introduces a geometry regularization to achieve nonlinear learning capability together with the approximately orthogonal property. Details are described in the following chapters.

4.2 Regularized Extreme Learning Machine Network

This section first introduces the geometry regularization term to achieve nonlinear ELM-AE learning with approximately orthogonal property, which is called R-ELM-AE in this thesis. Moreover, the overall pipeline is summarized, which is called Regularized ELM Network (R-ELMNet).

4.2.1 Regularized ELM Auto-Encoder

This work aims to improve the nonlinear ELM-AE performance in the NG-CNN pipeline without sacrificing ELM's BP-free advantage or introducing complicated re-scaling methods. Let $\mathbf{X} \in \mathbb{R}^{n \times d}$ be the training set, where each row vector $\mathbf{x}_j \in \mathbb{R}^d$ is the j -th sample and n is the number of samples. Then the output of ELM-AE is also $\mathbf{X}$.

The objective of nonlinear ELM-AE is to minimize the reconstruction error. Hyper-parameter L denotes the number of hidden neurons. Then resulting hidden weights β follow the shape of matrix $\mathbb{R}^{L \times d}$. The reconstruction error is shown as following:

$$\sum_{i=1}^n \| \mathbf{h}_i \beta - \mathbf{x}_i \|_2^2, \quad (4.1)$$

where $\mathbf{h}_i$ represents the i -th row vector of activation result $\mathbf{H}$ with respect to $\mathbf{x}_i$. And the activation output $\mathbf{H}$ can be computed with:

$$\mathbf{H} = g(\mathbf{XW}), \quad (4.2)$$

where $\mathbf{W} \in \mathbb{R}^{d \times L}$ is the random matrix and $g(\cdot)$ denotes activation function.

The geometry regularization term is defined as the Euclidean distance between orthogonally projected representation $\mathbf{x}_j \mathbf{A}$ and transformed feature $\mathbf{x}_j \beta^T$, shown in Equation 4.3.

$$\sum_{j=1}^n \| \mathbf{x}_j \beta^T - \mathbf{x}_j \mathbf{A} \|_2^2, \quad (4.3)$$

where $\mathbf{A} \in \mathbb{R}^{d \times L}$ is orthogonal random matrix with $\mathbf{A}^T \mathbf{A} = \mathbf{I}$.

Geometry regularization term restricts the transpose of β to keep valid value scale and approximately orthogonal property. Also with Theorem 4.2.1, it shows paired Euclidean distance can be preserved without distorting.

Theorem 4.2.1. The transformed representations $\mathbf{X} \beta^T$ have the following property with $L = \Omega(\epsilon^{-2} \lg(\epsilon^2 n))$ and $L \leq d$:

$$(1 - \epsilon) \| \mathbf{x}_i - \mathbf{x}_j \|_2^2 \leq \| \mathbf{x}_i \beta^T - \mathbf{x}_j \beta^T \|_2^2 \leq \| \mathbf{x}_i - \mathbf{x}_j \|_2^2,$$

$$\forall i, j = 1, \dots, n \quad s.t. \quad i \neq j, \quad c_1 < \epsilon < 1,$$

where $\mathbf{x}_i \beta^T$ is the transformed feature by ELM-AE of i -th sample, β is learned by minimizing $\sum_{j=1}^n \| \mathbf{x}_j \beta^T - \mathbf{x}_j \mathbf{A} \|^2$, $\mathbf{A}$ is orthogonal matrix with $\mathbf{A}^T \mathbf{A} = \mathbf{I}$ and $c_1 = \lg^{0.5001} n / \sqrt{\min\{n, d\}}$.

Proof. The Johnson-Lindenstrauss lemma [78, 105] proves that a linear map function $f : \mathbb{R}^d \rightarrow \mathbb{R}^L$ for dataset with n samples can satisfy property as below with $L = \mathcal{O}(\epsilon^{-2} \lg n)$:

$$(1 - \epsilon) \|u - v\|_2^2 \leq \|f(u) - f(v)\|_2^2 \leq (1 + \epsilon) \|u - v\|_2^2, \quad (4.4)$$

$$\forall u, v \quad s.t. \quad u \neq v, \quad 0 < \epsilon < 1/2,$$

where u and v denote paired data samples.

Substituting paired samples u and v by $\mathbf{x}_i$ and $\mathbf{x}_j$, then we obtain:

$$(1 - \epsilon) \|\mathbf{x}_i - \mathbf{x}_j\|_2^2 \leq \|f(\mathbf{x}_i) - f(\mathbf{x}_j)\|_2^2 \leq (1 + \epsilon) \|\mathbf{x}_i - \mathbf{x}_j\|_2^2, \quad (4.5)$$

$$\forall i, j = 1, \dots, n \quad s.t. \quad i \neq j, \quad 0 < \epsilon < 1/2.$$

Furthermore, the linear function f in Equation 4.4 and 4.5 can be orthogonal random projection [105]. Therefore the orthogonal projection $\mathbf{X}\mathbf{A}$ has the property as following:

$$(1 - \epsilon) \|\mathbf{x}_i - \mathbf{x}_j\|_2^2 \leq \|\mathbf{x}_i\mathbf{A} - \mathbf{x}_j\mathbf{A}\|_2^2 \leq (1 + \epsilon) \|\mathbf{x}_i - \mathbf{x}_j\|_2^2, \quad (4.6)$$

$$\forall i, j = 1, \dots, n \quad s.t. \quad i \neq j, \quad 0 < \epsilon < 1/2.$$

Obviously, the optimum β of geometry regularization term is $\mathbf{A}^T$. Thus, by replacing $\mathbf{A}$ with β^T , we obtain:

$$(1 - \epsilon) \|\mathbf{x}_i - \mathbf{x}_j\|_2^2 \leq \|\mathbf{x}_i\beta^T - \mathbf{x}_j\beta^T\|_2^2 \leq (1 + \epsilon) \|\mathbf{x}_i - \mathbf{x}_j\|_2^2, \quad (4.7)$$

$$\forall i, j = 1, \dots, n \quad s.t. \quad i \neq j, \quad 0 < \epsilon < 1/2.$$

Kasper et al. [106] show a lower bound $L = \Omega(\epsilon^{-2} \lg(\epsilon^2 n))$ providing the guarantee 4.4 with a wider range of $\epsilon \in (\lg^{0.5001} n / \sqrt{\min\{n, d\}}, 1)$. Meanwhile, as the orthogonal matrix $\mathbf{A}$ performs dimension reduction, that upper inequality can be shrunk with follows:

Let W denotes the $\text{span}(\mathbf{A})$, then the orthogonal complement $W^\perp$ of W is also a subspace of $V \in \mathbb{R}^d$. Equivalently, we have $V = W \oplus W^\perp$. Each vector $\mathbf{x} \in V$ can be represented by the summation $\mathbf{x}_W + \mathbf{x}_{W^\perp}$ and $\mathbf{x}_W = \mathbf{x} \mathbf{A} \mathbf{A}^T$, where $\mathbf{A} \mathbf{A}^T$ denotes the corresponding orthogonal projection operator based on $\mathbf{A}$. According to the triangle inequality, it can be shown:

$$\| \mathbf{x} \mathbf{A} \|_2^2 = \| \mathbf{x} \mathbf{A} \mathbf{A}^T \|_2^2 = \| \mathbf{x}_W \|_2^2 \leq \| \mathbf{x}_W \|_2^2 + \| \mathbf{x}_{W^\perp} \|_2^2 = \| \mathbf{x}_W + \mathbf{x}_{W^\perp} \|_2^2 = \| \mathbf{x} \|_2^2.$$

Substituting $\mathbf{x}$ with $(\mathbf{x}_i - \mathbf{x}_j)$, we have the following inequality:

$$\| \mathbf{x}_i \mathbf{A} - \mathbf{x}_j \mathbf{A} \|_2^2 \leq \| \mathbf{x}_i - \mathbf{x}_j \|_2^2. \quad (4.8)$$

Complete the proof.

Let Equation 4.3 be the final objective function, then filter learning step is comparable to LRF-ELM. Combining objectives 4.1 and 4.3, the loss function of R-ELM-AE is defined as following:

$$Loss = \alpha \sum_{i=1}^n \| \mathbf{h}_i \boldsymbol{\beta} - \mathbf{x}_i \|_2^2 + (1 - \alpha) \sum_{j=1}^n \| \mathbf{x}_j \boldsymbol{\beta}^T - \mathbf{x}_j \mathbf{A} \|_2^2, \quad (4.9)$$

where α denotes a balance hyper-parameter.

In this chapter, only $\alpha \in [0, 1]$ is considered for simplification. Let α be close to 1, R-ELM-AE emphasizes the reconstruction target to preserve information. If α approaches 0 it highlights the importance of predictable feature scale and orthogonal projection.

4.2.2 Learning Details

Solution of Objective: Based on the geometry regularized ELM-AE objective, the solution is shown below, which can preserve ELM's BP-free advantage. The partial derivatives with respect to β can be derived as:

$$\begin{aligned} \frac{\partial Loss}{\partial \beta} &= 2\alpha(\mathbf{H}^T \mathbf{H} \beta - \mathbf{H}^T \mathbf{X}) \\ &\quad + 2(1 - \alpha)(\beta \mathbf{X}^T \mathbf{X} - \mathbf{A}^T \mathbf{X}^T \mathbf{X}). \end{aligned} \quad (4.10)$$

Let derivatives be zero, then it follows a Sylvester equation $\mathbf{B}\beta + \beta\mathbf{C} = \mathbf{D}$ as below:

$$\alpha \mathbf{H}^T \mathbf{H} \beta + (1 - \alpha) \beta \mathbf{X}^T \mathbf{X} = \alpha \mathbf{H}^T \mathbf{X} + (1 - \alpha) \mathbf{A}^T \mathbf{X}^T \mathbf{X}, \quad (4.11)$$

where $\mathbf{B} = \alpha \mathbf{H}^T \mathbf{H}$, $\mathbf{C} = (1 - \alpha) \mathbf{X}^T \mathbf{X}$ and $\mathbf{D} = \alpha \mathbf{H}^T \mathbf{X} + (1 - \alpha) \mathbf{A}^T \mathbf{X}^T \mathbf{X}$.

The linear equation has a unique solution for all $\mathbf{D}$ as long as $\mathbf{B}$ and $-\mathbf{C}$ have no common eigenvalues [107]. We may assume covariance matrices $\mathbf{H}^T \mathbf{H}$ in ELM feature space, and $\mathbf{X}^T \mathbf{X}$ in data space could satisfy that condition with high probability. Because quantitative experiments on various scenarios have shown ELM feature space presents a better generalization capability compared to original data space. Nonetheless, assuming the situations where $\alpha \mathbf{H}^T \mathbf{H}$ and $(1 - \alpha) \mathbf{X}^T \mathbf{X}$ have the same eigenvalues, we can make another trial of generating random matrix $\mathbf{W}$ to avoid failure. Actually, in our experiments, we found that all $\mathbf{W}$ trials can satisfy the condition of a unique solution in the Sylvester equation, which supports our assumption empirically.

Note that the unknown weights β follow the shape of the matrix $\mathbb{R}^{L \times d}$. In the implemented NG-CNNs [35–37], L is set to 8 and d is 7². Thus, the additional time-cost of solving the Sylvester equation can be ignored.

Practical Implementation: To fulfill the condition of $L = \Omega(\epsilon^{-2} \lg(\epsilon^2 n))$, the shape of $\mathbf{X} \in \mathbb{R}^{n \times d}$ should be noted, where n represents the number of patches.

Recall that the $\mathbf{X}$ here is actually $\mathbf{M}$ in Equation 2.45. To reduce the volume of image patches into the legal range for L , the image patches matrix $\mathbf{M}$ is further processed with Algorithm 4.

Algorithm 4 Image patches processing.

Input:

The image patches $\mathbf{M}$;

Output:

The compressed image patches $\mathbf{M}_c$.

1. Let image patches follow the details: $\mathbf{M} = [\mathbf{M}_1, \dots, \mathbf{M}_N]$, where $\mathbf{M}_i \in \mathbb{R}^{k^2 \times (p-k+1)^2}$, p denotes the height or width of image and N is the number of images.
 2. Compute the mean patch with $\mathbf{M}_c = \frac{1}{N} \sum_{i=1}^N \mathbf{M}_i$.
 3. Transpose $\mathbf{M}_c$ and return.
-

Then the shape of $\mathbf{X}$ matches $(p-k+1)^2 \times k^2$. Typically for MNIST dataset, p , k , and n can be 28, 7, and 484, respectively, providing the feasible $L = 8$ to hit the lower bound. Furthermore, to remove the effect of randomness, the orthogonal matrix $\mathbf{A}$ is chosen from the orthonormal basis of the range of β and β is the general solution of nonlinear ELM-AE [24].

4.2.3 Orthogonality Analysis

Let α be zero; the solution of Equation 4.11 is exactly the orthogonal basis $\mathbf{A}$. While it is necessary to illustrate the effect on the final solution from increased α . Using the notation $\mathbf{P} \otimes \mathbf{Q}$ to represent the Kronecker product [108], the Sylvester equation [107] can be transformed into:

$$[\mathbf{I}_n \otimes \mathbf{B} + \mathbf{C}^T \otimes \mathbf{I}_L] \text{vec}(\beta) = \text{vec}(\mathbf{D}), \quad (4.12)$$

where $\mathbf{I}_n$ and $\mathbf{I}_L$ are identity matrices, $\mathbf{B}$ is $\alpha \mathbf{H}^T \mathbf{H}$, $\mathbf{C}$ is $(1-\alpha) \mathbf{X}^T \mathbf{X}$, $\mathbf{D}$ denotes $\alpha \mathbf{H}^T \mathbf{X} + (1-\alpha) \mathbf{A}^T \mathbf{X}^T \mathbf{X}$ and $\text{vec}(\beta) = \beta_{11}, \beta_{21} \dots \beta_{12} \dots$ represents vectorization to β .

As $\mathbf{B}$ and $\mathbf{C}$ are symmetric matrices, they can be reformed to a diagonal formula by similarity transformations:

$$\begin{aligned} \mathbf{U}^{-1}\mathbf{B}\mathbf{U} &= \boldsymbol{\lambda} = \text{diag}(\lambda_1, \dots, \lambda_L), \\ \mathbf{V}^{-1}\mathbf{C}\mathbf{V} &= \boldsymbol{\mu} = \text{diag}(\mu_1, \dots, \mu_d). \end{aligned} \quad (4.13)$$

Thus, the solution can be calculated with following as shown in [107]:

$$\begin{aligned} \boldsymbol{\beta} &= \mathbf{U}\hat{\boldsymbol{\beta}}\mathbf{V}^{-1}, \\ \hat{\beta}_{ij} &= \frac{\hat{D}_{ij}}{\lambda_i + \mu_j}, \\ \hat{\mathbf{D}} &= \mathbf{U}^{-1}\mathbf{D}\mathbf{V}. \end{aligned} \quad (4.14)$$

As $\hat{\mathbf{D}} = \mathbf{U}^{-1}\mathbf{D}\mathbf{V} = \mathbf{U}^{-1}(\alpha\mathbf{H}^T\mathbf{X} + (1 - \alpha)\mathbf{A}^T\mathbf{X}^T\mathbf{X})\mathbf{V}$, we have the approximation $\hat{\mathbf{D}} \approx \mathbf{U}^{-1}\mathbf{A}^T\mathbf{X}^T\mathbf{X}\mathbf{V} = \mathbf{U}^{-1}\mathbf{A}^T\mathbf{V}\boldsymbol{\mu}$ while α is close to zero. Accordingly, it also concludes $\hat{\beta}_{ij} \approx \hat{D}_{ij}/\mu_j$ as eigenvalues of $\alpha\mathbf{H}^T\mathbf{X}$ approximate to zero. Combining the above, we have $\hat{\boldsymbol{\beta}} \approx \mathbf{U}^{-1}\mathbf{A}^T\mathbf{V}$ and $\boldsymbol{\beta} \approx \mathbf{A}^T$. Based on that, the mathematical effect of hyper-parameter α on the final analytical solution is presented. Experimental illustrations in the following section verify our derivation and assumption.

4.2.4 Overall Pipeline

The proposed R-ELM-AE is integrated into the NG-CNN pipeline to learn convolutional filters. R-ELMNet is then composed of pre-processing, filter learning with R-ELM-AE, and post-processing. The pre-processing step only includes patch-mean removal, without any normalization or whitening operations, as shown in H-ELMNet. Therefore, the overall network structure achieves the minimal implementation level. The pipeline is shown in Figure 4.2. Also, experiments show the effectiveness compared with related methods.

FIGURE 4.2: Illustration of the R-ELMNet's network structure.

TABLE 4.1: Datasets summary.

Dataset	Samples	Classes	Training split	Testing split
Orl	400	40	240	160
Jaffe	213	7	150	63
Coil20	1,440	20	420	1,020
Coil100	7,200	100	2,100	5,100
Fashion	70,000	10	60,000	10,000
Letters	145,600	26	124,800	20,800

4.3 Experiments

4.3.1 Datasets Preparation

The proposed unsupervised feature learning method, R-ELMNet, was tested on six image classification datasets to evaluate its effectiveness compared with related NG-CNN methods. These datasets can be grouped into three according to data volume. Small volume datasets include Jaffe [99] and Orl [100], both contain hundreds of samples. Middle volume datasets have Coil20 [109] and Coil100 [110], which contain several thousands of images. Also, the performances on big datasets, such as Fashion [101] and Letters [102], were evaluated. The details of the datasets are illustrated in Table 4.1.

1) Small volume datasets:

Jaffe dataset [111] is a small face emotion recognition dataset and contains seven types of face emotions that are angry, disgusting, fear, happy, neutral, sad, and surprised. The face area was extracted using a public face detector with 68 registered landmarks [103] and then scaled. The dataset was partitioned into the training set and testing set with samples 150 and 63, respectively. Also, the proportions are roughly the same for all seven classes.

Orl dataset [112] contains 40 subjects for face recognition. There are ten different images of each subject. The samples were taken via varying the lighting, facial expressions, and facial details. The officially cropped front faces were directly used by resizing images to 28×28 . Samples were randomly divided into the training subset and testing subset with samples 240 and 160, respectively. The actions were repeated five times.

2) Middle volume datasets:

Coil20 [109] and Coil100 [110] datasets present object recognition tasks, the former has 20 categories, and the latter has 100 categories. All pictures were captured with a camera to cover 360 degrees. To verify the generalization capability, training and testing subsets share no adjacent camera angle. We sorted the samples of each category according to the camera angle. A sliding window with a size of 21 was adapted to extract the training subset at a random starting index. The left images formed the testing subset. The actions were repeated five times. Samples were resized into a size of 28×28 .

3) Big volume datasets:

Fashion [101] dataset consists of a training set of 60,000 samples and a testing set of 10,000 samples. Each sample is 28×28 grayscale image and comes from 10 classes, such as trouser, shirt, and so on.

Letters [102] is a bigger image dataset on character classification, with 124,800 samples for training and 20,800 samples for testing. It contains 26 classes, and each class includes upper and lower cases. Data samples follow 28×28 grayscale shape.

4.3.2 Experimental Setup

The experiments were conducted to evaluate the performance of the proposed unsupervised feature learning method on image classification. It was compared with PCANet [35], ELMNet [36], H-ELMNet [37] and LRF-ELM [25, 26]. NG-CNN-related methods include PCANet, ELMNet, and H-ELMNet. LRF-ELM shares a similar channel-separable convolution step and highlights the orthogonal random projection.

LRF-ELM boosts the nonlinear learning capability by the ELM classifier. While NG-CNN utilizes binarization and block-wise histogram for this purpose, and a linear SVM is adopted for efficiency. For a fair comparison, two versions for LRF-ELM were set up. They are LRF-ELM_{ELM} and LRF-ELM_{SVM}, where *ELM* and *SVM* denote nonlinear the ELM classifier and linear SVM classifier, respectively.

ELM classifier was implemented as a baseline. For more challenging comparison, convolution neural networks (CNN) were developed due to the similar convolutional structure. They are named CNN-2, CNN-3, and CNN-4, respectively. They were trained via CPU. Details can be found in the following. Meanwhile, an evolutionary deep neural network (EDEN) [113] is compared.

4.3.3 Parameter Analysis and Selection

The influence of parameter α in the objective function 4.9 is addressed in this subsection. As discussed in this chapter, the importance of orthogonal projection is emphasized. LRF-ELM and ELMNet also verify it. Thus, α was chosen from $[0, 0.1, 0.2, 0.3, 0.4, 0.5]$. The best parameters were selected by random grid search and fine-tuning.

To evaluate the effect of varying α in the two-layer NG-CNN pipeline, one α was changed from 0 to 0.5 by fixing another. Accuracy sensitivity is illustrated

TABLE 4.2: Parameter selection.

Methods	Orl	Jaffe	Coil20	Coil100	Fashion	Letters
ELM ¹	1200	1000	9000	10000	10000	10000
PCANet ²				C[7,8]-C[7,8]		
ELMNet ²				C[7,8]-C[7,8]		
H-ELMNet ²				C[7,8]-C[7,8]		
LRF-ELM _{SVM} ^{2,3}	C[7,8]-P[3,3]- C[7,8]-P[2,2]	C[7,8]-P[3,3]- C[7,8]-P[3,3]	C[7,8]-P[3,3]- C[7,8]-P[3,3]	C[7,8]-P[3,3]- C[7,8]-P[3,3]	C[7,8]-P[3,3]- C[7,8]-P[3,3]	C[7,8]-P[3,3]- C[7,8]-P[3,3]
LRF-ELM _{ELM}	C[7,8]-P[3,3]- C[7,8]-P[2,2]	C[7,8]-P[3,3]- C[7,8]-P[3,3]	C[7,8]-P[3,3]- C[7,8]-P[3,3]	C[7,8]-P[3,3]- C[7,8]-P[3,3]	C[7,8]-P[3,3]- C[7,8]-P[3,3]	C[7,8]-P[3,3]- C[7,8]-P[3,3]
CNN-2 ⁴	-	-		[16-32]-[1024-512]		
CNN-3 ⁴	-	-		[16-32-64]-[1024-512]		
CNN-4 ⁴	-	-		[16-32-64-128]-[1024-512]		
R-ELMNet ⁵	0.3-0.3	0.0-0.4	0.1-0.2	0.1-0.0	0.1-0.0	0.0-0.1

¹ The parameter of ELM method denotes the number of hidden nodes.² C[7,8] represents the kernel size $k = 7$ and the output dimension $L = 8$.³ P[i,j] shows pooling size i and stride j .⁴ For CNN-2, corresponding parameters denote a two-layer CNN structure with 16 and 32 convolutional kernels, respectively, followed by two fully connected layers. The maxpooling operation is added to each convolutional layer.⁵ The network structure of R-ELMNet follows C[7,8]-C[7,8], same to ELMNet, H-ELMNet, and PCANet. The parameter a - b means that the α of first R-ELM-AE is a , and b stands for the α of second R-ELM-AE.

FIGURE 4.3: Accuracy sensitivity to parameter α is illustrated. The blue line presents the effect of varying α of the first convolution stage by fixing the best α in the second convolution stage. The red line denotes the influence of changing α in the second convolution stage.

in Figure 4.3. The blue line presents the effect of varying α of the first convolution stage by fixing the best α in the second convolution stage. The red line denotes the influence of changing α in the second convolution stage.

We can learn from figures that bigger α performs better on small datasets, such as Orl, Jaffe, and Coil20. While on bigger datasets, the best choice of α in the first or second convolutional layer can be zero.

The details of parameter selection in this chapter are presented in Table 4.2. All NG-CNNs used the same two-layer structure, which is C[7,8]-C[7,8]. C[7,8] represents the kernel size $k=7$, and the output dimension $L=8$. LRF-ELM has an

additional pooling operation, which is denoted by $P[i,j]$ where i is pooling size, and j is the step size. CNN-2 contains the CNN structure [16, 32]-[1024,512]. The parameters denote a two-layer CNN with 16 and 32 convolutional kernels, followed by two fully connected layers with output dimensions 1024 and 512. Details of CNN-3 and CNN-4 can be found in Table 4.2.

4.3.4 Performance Comparison

The experimental results are illustrated in Table 4.3. We divide all methods into four groups. The first group includes ELM classifier as a baseline. The second group contains PCANet, ELMNet, H-ELMNet, LRF-ELM_{SVM}, and LRF-ELM_{ELM}. The third consists of three convolutional neural networks trained with the back-propagation method. All CNNs were only conducted on middle and big volume datasets. Thus, only the results on Coil20, Coil100, Fashion, and Letters are involved. The performance of an evolutionary deep neural network (EDEN) [113] is cited from the original paper as a reference to the Fashion dataset. The last group is the proposed R-ELMNet.

The bold font is applied to the results of R-ELMNet, as it outperforms all compared methods. Among the compared methods, their results may be discussed from several views.

- 1) All CNN models perform worse than any NG-CNN on Fashion dataset. Within CNNs, CNN-2 performs best on Fashion dataset. CNN-3 presents the best accuracy on Letters dataset. Obviously, the CNNs shows less competitive performance on middle volume datasets.
- 2) LRF-ELM_{SVM} performs worse than ELM classifier on Orl, Jaffe, and Coil20 datasets. Because nonlinear learning capability is not involved in LRF-ELM_{SVM}. While it surpassed ELM on Coil100, Fashion and Letters datasets. That means models benefit from the local-receptive-field design on bigger datasets.

TABLE 4.3: Mean accuracy comparison on scalable classification datasets.

Methods	Orl	Jaffe	Coil20	Coil100	Fashion	Letters	map^7
ELM	94.88	82.95	77.37	63.67	87.43	85.96	82.04
PCANet	98.38	88.25	80.72	71.08	91.08	93.34	87.14
ELMNet	98.13	86.67	81.17	71.72	91.17	93.25	87.02
H-ELMNet	98.75	88.89	81.09	71.65	91.21	93.46	87.51
LRF-ELM _{SVM} ¹	94.38	82.21	75.48	65.68	88.31	89.76	82.64
LRF-ELM _{ELM} ²	97.63	83.81	79.71	67.42	88.45	90.27	84.55
CNN-2 ³	-	-	75.46	65.85	90.99	93.26	-
CNN-3 ⁴	-	-	74.45	59.67	90.11	93.49	-
CNN-4 ⁵	-	-	74.92	62.17	90.02	93.31	-
EDEN[113] ⁶	-	-	-	-	90.60	-	-
R-ELMNet	99.38	89.84	84.28	73.31	91.32	93.59	88.62

¹ The classifier is linear SVM, same to PCANet, ELMNet, H-ELMNet and R-ELMNet.² The classifier is nonlinear ELM, as defined in the original paper.³ A 2-Conv CNN model, well-trained of 50 epochs.⁴ A 3-Conv CNN model, well-trained of 50 epochs.⁵ A 4-Conv CNN model, well-trained of 50 epochs.⁶ Directly cited from the original paper.⁷ Mean average performance across all datasets.

- 3) LRF-ELM_{ELM} shows comparable accuracies to ELM on small datasets and presents better performance than LRF-ELM_{SVM}.
- 4) H-ELMNet achieves better results than PCANet and ELMNet on big datasets. As H-ELMNet applied a complex and effective pre-processing step. The according drawback is time-inefficiency.
- 5) Result of EDEN is directly cited from the source. EDEN proposed an optimized shallow CNN structure. It also verifies our CNN baseline designs.
- 6) The performance of the proposed R-ELMNet, which is based on R-ELM-AE, is highlighted in the last row. We can conclude it outperforms all related methods.
- 7) Results also verify the effectiveness of NG-CNNs on scalable datasets. The volume of samples can be from tens to tens of thousands. Note that it is unnecessary to adjust the structure of NG-CNNs.

Compared with LRF-ELM, we may conclude that NG-CNNs make the improvement jump mainly by the post-processing step. The performance of the NG-CNN could be further improved by applying a kernel SVM or nonlinear ELM. Nevertheless, the final feature dimension of NG-CNNs restricts its extension to a more generalized classifier, as feature vector after post-processing can achieve the length of tens of thousands.

4.3.5 Comparison with Deeper CNN

Although NG-CNNs were proposed as the unsupervised feature extraction methods, they show competitive performance with the supervised trained CNNs. Obviously, deeper CNN, such as VGG [114], would perform better on bigger datasets, such as Fashion and Letters. R-ELMNet still shows its superiority from two aspects: 1) it performs better with a small fraction of training samples. 2) it uses

significantly fewer parameters and requires very limited FLOPs (floating-point operations).

Experiments for VGG² and R-ELMNet were conducted on Fashion and Letters with training samples from 1000 to 20000 and complete testing split. The corresponding plotting is shown in Figure 4.4 that R-ELMNet is more robust to the volume of training size.

FIGURE 4.4: Robustness comparison on the various training volume. R-ELMNet shows better accuracy while training size is less than 20000.

4.3.6 Learning Efficiency Discussion

The time-cost on Fashion and Letters datasets was evaluated for related methods. The training and inference time are shown in Table 4.4. Obviously, the LRF-ELM presented the best training speed, as the convolutional kernels are learning-free. Compared to other NG-CNN methods, H-ELMNet spent significant time. The main reason is due to the LCN post-processing procedure. The CNN-related methods, especially VGG, demonstrated less efficiency.

As shown in the previous section, the R-ELMNet is more robust to the training size. One main reason comes from its minimal model complexity. R-ELMNet only needs 784 parameters. Compared to CNNs with millions of parameters, R-ELMNet could present state-of-the-art performance for datasets from hundreds to hundreds of thousands of samples. Details are shown in Table 4ftab:mcc. Note

²The last convolutional block is removed due to the small image size.

that R-ELMNet has similar FLOPs with shallow CNNs because it still could be improved further via pooling. We put that in the future work.

TABLE 4.4: Learning Efficiency (minutes) Comparison on Big Datasets.

Methods	Fashion	Letters
LRF-ELM	$F(3)+C(0.9)+I(0.4)^2$	$F(6)+C(1.7)+I(0.8)$
NG-CNN* ¹	$F(25)+C(6)+I(3)$	$F(47)+C(26)+I(5)$
H-ELMNet	$F(220)+C(6)+I(21)$	$F(379)+C(27)+I(39)$
CNN-2	$F(39)+I(0.1)$	$F(115)+I(0.2)$
CNN-3	$F(47)+I(0.1)$	$F(162)+I(0.2)$
CNN-4	$F(55)+I(0.1)$	$F(227)+I(0.2)$
VGG	$F(521)+I(0.4)$	$F(1107)+I(0.7)$

¹ NG-CNN* denotes the summary for PCANet, ELMNet, and R-ELMNet, as they share same time-efficiency. H-ELMNet takes much longer training time due to its complexity.

² The $F(\cdot)$ represents feature learning time-cost. The $C(\cdot)$ stands for the classifier training procedure, as CNN-related models are end-to-end supervised methods that the $C(\cdot)$ is therefore merged into $F(\cdot)$. The $I(\cdot)$ denotes the inference time-cost. The time precision is 0.1 minute if time-cost is less than 2 minutes.

TABLE 4.5: Model Complexity Comparison.

Methods	Number of Parameters	FLOPs ¹
R-ELMNet	784 ²	6.4 <i>m</i>
CNN-2	2.1 <i>m</i>	6.3 <i>m</i>
CNN-3	1.6 <i>m</i>	7.0 <i>m</i>
CNN-4	2.7 <i>m</i>	16.3 <i>m</i>
VGG	30.0 <i>m</i>	197.9 <i>m</i>

¹ Floating-point operations per image from Fashion or Letters datasets. The *m* denotes million.

² It excludes the parameters in linear SVM.

4.3.7 Orthogonality Visualization

To show the effect of increasing α on the orthogonality of β , experimental illustrations were accomplished by calculating the covariance matrix $\beta\beta^T \in R^{8 \times 8}$. The resulting matrices with various α selections from the same input were resized and rescaled between 0 and 1. As presented in Figure 4.5, pictures from left to right denote matrices with $\alpha=0$, $\alpha=0.1$, $\alpha=0.2$, $\alpha=0.5$, and $\alpha=1$, respectively. The color

block within each picture denotes a value close to 1 while it shows white. Covariance matrices with smaller α present more significant orthogonality of β , verifying the mathematical derivation and discussion.

FIGURE 4.5: Orthogonality visualization of $Mat = \beta\beta^T$. The upper row demonstrates Mat with directly learned β , while we normalize the row vector of β first before plotting lower figures. The color block within each picture denotes a value close to 1 while it shows white. The difference within rightmost column also shows that the magnitude of corresponding β_1 is huge.

4.3.8 Feature Map Visualization

The visualization of feature maps, generated by PCA, nonlinear ELM-AE, ELM-AE^{Or}, and R-ELM-AE, respectively, are shown in Figure 4.6. Feature maps came from one same sample. Feature map values were clipped by the range $[-2, 2]$ for an equal plotting scheme. The color approaches yellow, while the pixel value is close to 2. There are 8 and 64 feature maps in layer one and two, respectively. Only the first 8 of 64 feature maps in the layer two are illustrated.

The drawback of nonlinear ELM-AE can be directly learned from Figure 4.6 in row *b*. The values were stretched to a big scale in layer two. The discriminative capability is then accordingly suppressed. Meanwhile, figures from the second layer contain analogous feature patterns, such as the similarity between the 4-th and 8-th map. From row *d*, we can conclude that feature maps from left to right are

(a) Layer one with $ELM-AE^{No}$ (b) Layer two with $ELM-AE^{No}$ (c) Layer one with PCA (d) Layer two with PCA (e) Layer one with $ELM-AE^{Or}$ (f) Layer two with $ELM-AE^{Or}$ (g) Layer one with $R-ELM-AE$ (h) Layer two with $R-ELM-AE$

FIGURE 4.6: Feature maps from two cs-convolutional layers are shown for filtering learning methods. They came from one same sample of Fashion dataset. Feature map values were clipped with range $[-2,2]$ for equal plotting scheme. The color approaches yellow, while the pixel value is close to 2. There are 8 and 64 feature maps in layer one and two, respectively. Only the first 8 of 64 feature maps in layer two are illustrated.

computed by principal components corresponding covariances from big to small. In contrast, the rightmost feature maps contain more noisy information.

Compared with nonlinear ELM-AE and ELM-AE^{Or} , the feature maps generated by R-ELM-AE have proper feature patterns and value scale, which is important for the post-processing stage.

4.4 Conclusions and Future Work

In this chapter, deep insight into the application of ELM-AEs in the NG-CNN pipeline is analyzed. Despite the superiority of ELM-AEs over PCA within the past contributions, the merit of orthogonal projection from PCA, linear ELM-AE, and LRF-ELM is still highlighted within the NG-CNN pipeline. Accordingly, a regularized Extreme Learning Machine Auto-Encoder (R-ELM-AE) designed for NG-CNN is proposed. The R-ELM-AE fuses the feature of nonlinear learning capability and approximately orthogonal projection. A theorem also verifies the geometry restriction of R-ELM-AE. Without integrating complex pre-processing methods, such as LCN and whitening, a more efficient NG-CNN is achieved by simply replacing the filter learning method with R-ELM-AE. The overall structure is called R-ELMNet. Experiments on scalable datasets demonstrate the effectiveness of R-ELMNet, compared with PCANet, ELMNet, H-ELMNet, and related supervised CNNs.

Although the R-ELMNet is a light and powerful unsupervised representation learning method, it still shares network similarity with popular CNNs. Thus, the improvement of R-ELMNet could be inspired by the recent success of CNNs. The first concern is to develop an adapted channel pruning method, as He et al. [115] proposed for supervised CNN. The convolutional kernels of the R-ELMNet are directly related to the hidden nodes. Therefore, the pruning methods for fully connected ELM, such as SB-ELM-AE, Optimally Pruned ELM (OP-ELM) [116]

and its extension TROP-ELM [117], hold potential applications. The second concern is to form a valid three-dimensional or cross-channel convolution structure, which could apparently extend the receptive field and provide a flexible design.

Chapter 5

Unified ELM-AE for Dimension Reduction and Extensive Applications

Chapter 5 summarizes the drawbacks of applying ELM-AE variants to extensive scenarios as the plug-and-play role for representation learning or dimension reduction. In this chapter, the Unified ELM-AE (U-ELM-AE) is proposed for dimension reduction, which involves no additional hyper-parameters. Experiments show the competitive effectiveness and efficiency with popular ELM-AE variants and PCA. Thus, it can be conveniently verified the flexibility and adaptability to various scenarios. The experiment also presents the improvement for NG-CNN and LRF-ELM as the role of dimension reduction.

5.1 Motivation

PCA [14, 15] aims to linearly project data by an orthogonal matrix that the first dimension in the reduced space can describe the most variance of data while the second dimension can present the second most variance of data, and so on. PCA is very popular for dimension reduction or data visualization [14, 15]. Following ELM-AE [24], which shows consistent improvement compared to PCA on various datasets, researchers focus on the application or extension of basic ELM-AE [36, 37, 81, 81, 83, 84, 118]. Despite the success of ELM-AEs, it is not broadly used to the traditional scenarios where PCA is commonly integrated, such as the dimension reduction role in machine learning. There are two main reasons which restrict the propagation of ELM-AE for dimension reduction.

Firstly, the value scale after data transformation is not bounded, illustrated in the previous chapter. While PCA could avoid additional post-normalization methods that ELM-AEs commonly take, as PCA's transformation matrix holds the orthonormal property.

Secondly, PCA has only one hyper-parameter, the reduced dimension, while ELM-AE generally adds the ℓ_2 -regularization term. For example in [24], the range of ℓ_2 -norm is $[1e-8, \dots, 1e7, 1e8]$. This situation may be worse due to involving the hyper-parameters from data normalization or value scaling tricks.

Based on the former paragraph, using ELM-AE may take a long time for hyper-parameter tuning. Considering PCA often acts as the plug-and-play role for dimension reduction, a simple variant of ELM-AE is desired, verified the adaptability to any model with minimal trials. In this chapter, the proposed Unified ELM-AE (U-ELM-AE) presents competitive performance with other variants of ELM-AEs and PCA, and importantly only involves one hyper-parameter, the reduced dimension. The rest contents of this chapter are organized: the proposed Unified ELM-AE (referred to as U-ELM-AE for simplification), the extensive applications based on the proposed ELM-AE variant, the experiments, and the conclusion.

5.2 Proposed Method

Firstly, the definitions of mathematical notations are introduced. Let the matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ represent the input data, where n denotes the number of samples and d indicates the flatten data vector dimension. After random projection with matrix $\mathbf{W} \in \mathbb{R}^{d \times L}$ and activation function $g(\cdot)$, the ELM embedding $\mathbf{H} \in \mathbb{R}^{n \times L}$ is calculated by a standard operation in ELM's scenario: $\mathbf{H} = g(\mathbf{XW})$. The unknown transformation matrix $\beta \in \mathbb{R}^{L \times d}$ comes from ELM-AE's objective:

$$\beta = (C\mathbf{I} + \mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T \mathbf{X}, \quad (5.1)$$

where C denotes a ℓ_2 -regularization term.

As discussed in the motivation, involving C imposes additional tuning time and model uncertainty in the scenarios where PCA can act as the plug-and-play role for dimension reduction. That restricts the applications of ELM-AE.

In the following sections, a simple ELM-AE variant is proposed, which is called Unified ELM-AE (U-ELM-AE). Secondly, the direct applications based on U-ELM-AE for NG-CNN and LRF-ELM shows its broad effectiveness.

5.2.1 Unified ELM-AE for Dimension Reduction

Inspired by previous chapter, where a geometry regularization term is introduced to restrict the value scale of transformed data, here we discuss the analytical solution under the condition $\beta\beta^T = \mathbf{I}$, where $\beta \in \mathbb{R}^{L \times d}$ and $L \leq d$. The objective follows:

$$\begin{aligned} \text{Minimize : } & \|\mathbf{H}\beta - \mathbf{X}\|_F^2, \\ \text{Subject to : } & \beta\beta^T = \mathbf{I}. \end{aligned} \quad (5.2)$$

Equal dimension projection: Considering the situation of equal dimension projection where L is equal to d . As β is the square matrix, the condition is then

extended to $\beta^T \beta = \beta \beta^T = \mathbf{I}$. Formally, it is a standard orthogonal Procrustes problem [119, 120], which can be stated as below:

$$\begin{aligned} \text{Minimize : } & \|\mathbf{A}\mathbf{Q} - \mathbf{B}\|_F^2, \\ \text{Subject to : } & \mathbf{Q}^T \mathbf{Q} = \mathbf{Q} \mathbf{Q}^T = \mathbf{I}, \end{aligned} \quad (5.3)$$

where $\mathbf{A} \in \mathbb{R}^{n \times d}$, $\mathbf{B} \in \mathbb{R}^{n \times d}$ and $\mathbf{Q} \in \mathbb{R}^{d \times d}$.

The orthogonal Procrustes problem can be understood from a matrix approximation view. Let $\mathbf{A}$ be the coordinates of n points, and $\mathbf{B}$ be the corresponding target coordinates. Equation 5.3 aims to minimize the least squares error under the condition $\mathbf{Q}^T \mathbf{Q} = \mathbf{Q} \mathbf{Q}^T = \mathbf{I}$. Actually, $\mathbf{A}$ is $\mathbf{H}$, $\mathbf{B}$ is $\mathbf{X}$ and $\mathbf{Q}$ is β . The learned β is used to minimize the least squares distance between $\mathbf{H}$ and $\mathbf{X}$. The original solution [121] of Equation 5.3 is as below:

$$\begin{aligned} \mathbf{M} &= \mathbf{X}^T \mathbf{H}, \\ [\mathbf{U}, \mathbf{\Sigma}, \mathbf{V}] &= \text{svd}(\mathbf{M}), \\ \beta &= \mathbf{V} \mathbf{U}^T, \end{aligned} \quad (5.4)$$

where *svd* denotes singular value decomposition (SVD).

To understand objective 5.3 better, it is re-written as Equation 5.5, which is important to present the solution of the case $L < d$ in the following.

$$\begin{aligned} \|\mathbf{H}\beta - \mathbf{X}\|_F^2 &= \text{trace}((\mathbf{H}\beta - \mathbf{X})^T (\mathbf{H}\beta - \mathbf{X})) \\ &= \text{trace}(\beta^T \mathbf{H}^T \mathbf{H} \beta - \beta^T \mathbf{H}^T \mathbf{X} - \mathbf{X}^T \mathbf{H} \beta + \mathbf{X}^T \mathbf{X}) \\ &= \text{trace}(\beta^T \mathbf{H}^T \mathbf{H} \beta) - \text{trace}(2\mathbf{X}^T \mathbf{H} \beta) + \text{trace}(\mathbf{X}^T \mathbf{X}) \\ &= \text{trace}(\mathbf{H}^T \mathbf{H} \beta \beta^T) - \text{trace}(2\mathbf{X}^T \mathbf{H} \beta) + \text{trace}(\mathbf{X}^T \mathbf{X}) \\ &= \text{trace}(\mathbf{H}^T \mathbf{H} + \mathbf{X}^T \mathbf{X}) - 2\text{trace}(\mathbf{X}^T \mathbf{H} \beta) \\ &= \text{const} - 2\text{trace}(\mathbf{X}^T \mathbf{H} \beta). \end{aligned} \quad (5.5)$$

Thus, this problem is to maximize $\text{trace}(\mathbf{X}^T \mathbf{H} \boldsymbol{\beta})$ under the condition $\boldsymbol{\beta}^T \boldsymbol{\beta} = \boldsymbol{\beta} \boldsymbol{\beta}^T = \mathbf{I}$. A simple solution [122] has been derived as below:

$$\begin{aligned} \text{trace}(\mathbf{X}^T \mathbf{H} \boldsymbol{\beta}) &= \text{trace}(\mathbf{U} \boldsymbol{\Sigma} \mathbf{V}^T \boldsymbol{\beta}) \\ &= \text{trace}(\boldsymbol{\Sigma} \mathbf{V}^T \boldsymbol{\beta} \mathbf{U}) \\ &= \text{trace}(\boldsymbol{\Sigma} \mathbf{P}), \end{aligned} \quad (5.6)$$

where $[\mathbf{U}, \boldsymbol{\Sigma}, \mathbf{V}] = \text{svd}(\mathbf{X}^T \mathbf{H})$. Apparently, we know $\mathbf{P}^T \mathbf{P} = \mathbf{P} \mathbf{P}^T = \mathbf{I}$ and $\text{trace}(\boldsymbol{\Sigma} \mathbf{P}) = \sum_i^d \boldsymbol{\Sigma}_{i,i} \mathbf{P}_{i,i} \leq \sum_i^d \boldsymbol{\Sigma}_{i,i}$. The equality holds only if $\mathbf{P} = \mathbf{I}$. Thus, objective 5.6 can achieve maximum if $\boldsymbol{\beta} = \mathbf{V} \mathbf{U}^T$.

Dimension reduction: For case $L < d$, objective is similar to 5.3 while the difference is $\boldsymbol{\beta}^T \boldsymbol{\beta} \neq \mathbf{I}$ and $\boldsymbol{\beta} \boldsymbol{\beta}^T = \mathbf{I}$. Nevertheless the key derivation in Equation 5.5 still holds, which is shown as Equation 5.7.

$$\text{trace}(\boldsymbol{\beta}^T \mathbf{H}^T \mathbf{H} \boldsymbol{\beta}) = \text{trace}(\mathbf{H}^T \mathbf{H} \boldsymbol{\beta} \boldsymbol{\beta}^T) = \text{trace}(\mathbf{H}^T \mathbf{H}) = \text{const.} \quad (5.7)$$

For the case $L < d$, the objective of ELM-AE under condition $\boldsymbol{\beta} \boldsymbol{\beta}^T = \mathbf{I}$ is also equal to maximize $\text{trace}(\mathbf{X}^T \mathbf{H} \boldsymbol{\beta})$. Firstly, singular value decomposition (SVD) is applied on $\mathbf{X}^T \mathbf{H}$ with full orthogonal matrices and then get $[\mathbf{U}_{d \times d}, \boldsymbol{\Sigma}_{d \times L}, \mathbf{V}_{L \times L}]$. Thus, as presented in Equation 5.6, the objective is to maximize $\text{trace}(\boldsymbol{\Sigma}_{d \times L} \mathbf{V}_{L \times L}^T \boldsymbol{\beta} \mathbf{U}_{d \times d})$. Because $\boldsymbol{\Sigma}_{d \times L}$ only has L diagonal values, the target achieves maximum when diagonal values of $\mathbf{V}_{L \times L}^T \boldsymbol{\beta} \mathbf{U}_{d \times d}$ are 1 (the rows of $\mathbf{V}_{L \times L}^T \boldsymbol{\beta} \mathbf{U}_{d \times d}$ are orthonormal). Apparently, $\boldsymbol{\beta} = \mathbf{V}_{L \times L} \mathbf{U}_{d \times L}^T$ satisfies the target under condition $\boldsymbol{\beta} \boldsymbol{\beta}^T = \mathbf{I}$ to enable L diagonal ones of $\mathbf{V}_{L \times L}^T \boldsymbol{\beta} \mathbf{U}_{d \times d}$, where $\mathbf{U}_{d \times L}$ denotes the first L columns of $\mathbf{U}_{d \times d}$.

Summary: According to the discussions about equal dimension projection $L = d$ and dimension reduction $L < d$, we present the closed-form solution of U-ELM-AE 5.2, which is $\boldsymbol{\beta} = \mathbf{V}_{L \times L} \mathbf{U}_{d \times L}^T$ and $[\mathbf{U}_{d \times L}, \boldsymbol{\Sigma}, \mathbf{V}_{L \times L}] = \text{svd}(\mathbf{X}^T \mathbf{H})$. Thus, the transformed data follows Equation 5.8. Meanwhile, the activation function

$g(\cdot)$, such as sigmoid or tanh, can be applied to $\mathbf{X}^{proj}$.

$$\mathbf{X}_{proj} = \mathbf{X}\mathbf{U}_{d \times L}\mathbf{V}_{L \times L}^T. \quad (5.8)$$

Theorem 5.2.1. The U-ELM-AE with $L \leq d$ aims to minimize the Frobenius norm of residual $\mathbf{H}\boldsymbol{\beta} - \mathbf{X}$ subjected to $\boldsymbol{\beta}\boldsymbol{\beta}^T = \mathbf{I}$. This problem has the closed-form solution $\boldsymbol{\beta} = \mathbf{V}_{L \times L}\mathbf{U}_{d \times L}^T$, where $\mathbf{U}_{d \times L}$ denotes the first L columns of $\mathbf{U}_{d \times d}$. $\mathbf{U}_{d \times d}$ and $\mathbf{V}_{L \times L}$ are calculated via the singular value decomposition of $\mathbf{X}^T\mathbf{H}$. This learning procedure is equivalent to solving the matrix $\mathbf{Q}$ of maximizing the RV coefficient between $\mathbf{H}\mathbf{Q}$ and $\mathbf{X}$ with constraint $\mathbf{Q}\mathbf{Q}^T = \mathbf{I}$, also equivalent to finding optimal matrix $\mathbf{Q}$ of maximizing the inner-product between $\mathbf{X}\mathbf{Q}^T$ and $\mathbf{H}$ under condition $\mathbf{Q}\mathbf{Q}^T = \mathbf{I}$.

Proof. Given two data sets $\mathbf{A}$ and $\mathbf{B}$ with the same matrix size, the RV coefficient [123, 124], as shown in 5.9, can be used as the measurement of the similarity between $\mathbf{A}$ and $\mathbf{B}$

$$r_v^2 = \frac{\{\text{trace}(\mathbf{A}^T\mathbf{B})\}^2}{\text{trace}\{\mathbf{A}^T\mathbf{A}\}\text{trace}\{\mathbf{B}^T\mathbf{B}\}}. \quad (5.9)$$

Let $\mathbf{A}$ be $\mathbf{H}\mathbf{Q}$ and $\mathbf{B}$ be $\mathbf{X}$, where $\mathbf{X} \in \mathbb{R}^{n \times d}$, $\mathbf{H} \in \mathbb{R}^{n \times L}$, and $\mathbf{Q} \in \mathbb{R}^{L \times d}$. The RV coefficient is transformed into the form 5.10. As $\text{trace}\{(\mathbf{H}\mathbf{Q})^T\mathbf{H}\mathbf{Q}\}$ is the same to $\text{trace}\{\mathbf{H}^T\mathbf{H}\mathbf{Q}\mathbf{Q}^T\}$ and $\text{trace}\{\mathbf{H}^T\mathbf{H}\}$ is constant. Thus, only the numerator is effective. Then this problem is equal to U-ELM-AE

$$r_v^2 = \frac{\{\text{trace}(\mathbf{X}^T\mathbf{H}\mathbf{Q})\}^2}{\text{trace}\{(\mathbf{H}\mathbf{Q})^T\mathbf{H}\mathbf{Q}\}\text{trace}\{\mathbf{X}^T\mathbf{X}\}}. \quad (5.10)$$

Inner-product of matrices is shown via Equation 5.11.

$$\phi = \text{trace}(\mathbf{A}^T\mathbf{B}). \quad (5.11)$$

Let $\mathbf{A}$ be $\mathbf{H}$ and $\mathbf{B}$ be $\mathbf{X}\mathbf{Q}^T$, the objective of inner-product is apparently equivalent to U-ELM-AE via following transformation.

$$\text{trace}(\mathbf{H}^T \mathbf{X} \mathbf{Q}^T) = \text{trace}(\mathbf{Q}^T \mathbf{H}^T \mathbf{X}) = \text{trace}((\mathbf{H} \mathbf{Q})^T \mathbf{X}) = \text{trace}(\mathbf{X}^T \mathbf{H} \mathbf{Q}). \quad (5.12)$$

Complete the proof.

Remark 5.2.2. Feature projection procedure $\mathbf{X}\boldsymbol{\beta}^T$ of U-ELM-AE is equivalent to a generalized rotation from $\mathbf{X}$ to ELM feature space via the transpose of $\boldsymbol{\beta}$.

From the Theorem 5.2.1, the connections of U-ELM-AE with the RV coefficient and inner-product are bridged. Also, a straightforward geometrical understanding of U-ELM-AE is shown in Remark 5.2.2.

Although the transformation is to fit input onto ELM feature space, the U-ELM-AE is not simply equal to minimizing least squares error between $\mathbf{X}\boldsymbol{\beta}^T$ and $\mathbf{H}$ as illustrated below:

$$\begin{aligned} \|\mathbf{X}\boldsymbol{\beta}^T - \mathbf{H}\|_F^2 &= \text{trace}((\mathbf{X}\boldsymbol{\beta}^T - \mathbf{H})^T (\mathbf{X}\boldsymbol{\beta}^T - \mathbf{H})) \\ &= \text{trace}(\boldsymbol{\beta} \mathbf{X}^T \mathbf{X} \boldsymbol{\beta}^T - 2\mathbf{H}^T \mathbf{X} \boldsymbol{\beta} + \mathbf{H}^T \mathbf{H}) \\ &= \text{trace}(\mathbf{X}^T \mathbf{X} \boldsymbol{\beta}^T \boldsymbol{\beta}) - 2\text{trace}(\mathbf{H}^T \mathbf{X} \boldsymbol{\beta}) + \text{const}, \end{aligned} \quad (5.13)$$

where $\text{trace}(\mathbf{X}^T \mathbf{X} \boldsymbol{\beta}^T \boldsymbol{\beta})$ is not constant, as $\boldsymbol{\beta}^T \boldsymbol{\beta}$ is not equal to identity matrix. The Equation 5.13 has no closed-form solution.

Compared to U-ELM-AE, the orthogonal ELM [125] was proposed to minimize the least squares error for classification with orthogonal constraint $\boldsymbol{\beta}^T \boldsymbol{\beta} = \mathbf{I}$ rather than $\boldsymbol{\beta} \boldsymbol{\beta}^T = \mathbf{I}$. Because the classification problem generally requires the number of hidden neurons to be large enough, orthogonal ELM utilizes an iterative optimization method to solve $\boldsymbol{\beta}$. In contrast, in the scenario of U-ELM-AE, a simple and efficient solution to find optimal $\boldsymbol{\beta}$ is introduced, also the extensive applications of U-ELM-AE as plug-and-play role are shown in following Section 5.3.

5.2.2 Comparison with PCA, linear ELM-AE, nonlinear ELM-AE, and SB-ELM-AE

Compared with PCA, linear ELM-AE, nonlinear ELM-AE, and SB-ELM-AE, the advantages of proposed U-ELM-AE can be discussed from several aspects.

ELM Embedding: In this chapter, one method has the property of ELM embedding, as long as the random projection and activation are applied, because such designs have demonstrated comprehensive applications. Thus, the property of ELM embedding is highlighted. Obviously, linear ELM-AE and PCA stand out of this scope.

Learning Efficiency: PCA, linear ELM-AE, nonlinear ELM-AE, and U-ELM-AE are all training-efficient. These methods take a similar magnitude of time-cost for training. Only the SB-ELM-AE spends evidently longer time for learning. Time-efficiency matters if we expect a plug-and-play role of U-ELM-AE in various machine learning scenarios. Meanwhile, the SB-ELM-AE, the same as other sparsity-regularized methods, requires significantly more hidden neurons, which might limit its extensive applications.

Geometrical Interpretation: Geometrical interpretation to feature learning is necessary. Without that, it would bring difficulty to understand the motivation of ELM-AE. And we present a geometrical interpretation of U-ELM-AE, details are illustrated in Theorem 5.2.1 and Remark 5.2.2.

Bounded Output: The methods, including PCA, linear ELM-AE, nonlinear ELM-AE, SB-ELM-AE, and U-ELM-AE, are regarded to have the property of bounded output if the norm of transformed feature is certain and bounded. Square orthonormal matrices preserve the norm as illustrated below, and hence the norm of the vector keeps unchanged even with multiple multiplications by orthogonal matrix. The merits of orthogonal projection are also verified in the area of deep learning [126, 127].

$$\|\mathbf{A}\mathbf{x}\|^2 = \|\mathbf{x}\|^2. \quad (5.14)$$

We may conclude that PCA, linear ELM-AE, and U-ELM-AE have this property. Because the transformation matrix β is orthogonal (PCA and U-ELM-AE) or approximately orthogonal (linear ELM-AE).

Free of Additional Parameters: Besides the reduced dimension, generally, we expect no additional parameters for convenience. PCA, linear ELM-AE, and U-ELM-AE involve no additional parameters. While nonlinear ELM-AE requires a ℓ_2 -regularization term, SB-ELM-AE takes prior variances, convergence factor, and so on. More hyper-parameters may come from data processing methods.

The U-ELM-AE takes no additional hyper-parameters. It also avoids post-processing operations on the transformed feature. This property is essential for extensive applications.

Detailed comparisons are illustrated in Table 5.1. A checkmark means the corresponding property is involved in the method.

5.2.3 Comparison with SAE, VAE, and SOM

The learning procedure of SAE, VAE, and Self-Organizing Map (SOM) [128] all require iterative updating learning. As discussed in the previous section, it may bring implementation difficulty as the plug-and-play method, which is one of the main motivations of U-ELM-AE. The significance of U-ELM-AE could be illustrated from the following views: 1) the iterative learning spends longer training time while the U-ELM-AE is light; 2) the SOM generally maps the input into discrete space and aims for visualization. The extensive applications based on the powerful U-ELM-AE in the following chapter are introduced, verifying the motivation and innovation.

Methods	ELM Embedding	Learning Efficiency	Geometrical Interpreta- tion	Bounded Output	Free of Additional Parameters
U-ELM-AE	✓	✓	✓	✓	✓
PCA		✓	✓	✓	✓
ELM-AE (linear)		✓	✓	✓	✓
ELM-AE (nonlinear)	✓	✓			
SB-ELM-AE	✓				

TABLE 5.1: The property comparisons of U-ELM-AE, PCA, ELM-AE (linear), ELM-AE (nonlinear), and SB-ELM-AE. A check symbol indicates the method has the corresponding property.

5.3 Extensive Applications

The U-ELM-AE is initially proposed for dimension reduction. Meanwhile, its extensions into LRF-ELM and NG-CNN are presented. LRF-ELM [25, 26] utilizes the orthogonal random matrix as the convolutional kernels. NG-CNNs use PCA, linear ELM-AE, nonlinear ELM-AE, or R-ELM-AE in the stage of filter learning. In the following contents, these two pipelines can be conveniently integrated with U-ELM-AE. The advantages of U-ELM-AE are accordingly highlighted, such as orthogonal projection and involving no additional hyper-parameters.

5.3.1 Local Receptive Fields-based Extreme Learning Machine with U-ELM-AE

The LRF-ELM incorporates orthogonal convolution, pooling, and fully-connected layers.

- (1) Orthogonal convolution layer: the convolutional kernels are randomly generated and then orthogonalized.
- (2) Pooling layer: LRF-ELM uses square/square-root pooling, which has the properties of rectification nonlinearity and translation invariance [87, 88].
- (3) Fully-connected layer: the feature maps passed from the pooling layer construct the input for classification. We may denote the fully-connected layer as the feature learning layer of single-layer ELM, illustrated in Figure 2.1. Thus, the orthogonal convolution and pooling layers are analogous to single-layer ELM's ELM feature mapping procedure.

This chapter focuses on the improvement of orthogonal convolution layer. Given the window size $k \times k$ of local receptive field and output dimension L ($L < k^2$ is generally satisfied), LRF-ELM generates the convolutional kernels with three steps:

- (1) Generate initial random kernel $\hat{\mathbf{A}} \in \mathbb{R}^{k^2 \times k^2}$. Each element is sampled from Gaussian or uniform distribution.
- (2) Applying singular value decomposition on $\hat{\mathbf{A}}$, we get $\mathbf{U}$, $\mathbf{\Sigma}$, and $\mathbf{V}$. Next the first L columns of $\mathbf{U}$ are selected to form final $\mathbf{A} \in \mathbb{R}^{k^2 \times L}$.
- (3) Each column $\mathbf{a}_i$ of $\mathbf{A}$ represents a convolutional kernel. The $\mathbf{a}_i$ accepts a local receptive window $k \times k$ and produces the i -th feature map.

The random matrix $\mathbf{A}$ projects data from k^2 -dimensional data onto L -dimensional space, typically $k = 7$ and $L = 8$. Thus, it is not guaranteed that all the randomly generated filters impose a positive affect on the feature learning. Once the j -th filter $\mathbf{a}_j$ takes negative influence, then the j -th feature map may be regarded as noise.

Hence, U-ELM-AE can conveniently substitute random matrix $\mathbf{A}$ to generate the orthogonal matrix $\mathbf{A}$. Given the training samples $\mathbf{X} \in \mathbb{R}^{n \times H \times W}$, sliding window $k \times k$ is used to extract image patches and form the patch matrix $\mathbf{M}$. Then the U-ELM-AE is trained on $\mathbf{M}$, and $\mathbf{A}$ is therefore learned. The detailed learning steps are illustrated in Algorithm 5:

Note the mean patch $\bar{\mathbf{P}}$ is adapted rather than using complete patch matrix $\mathbf{P}$ due to following considerations:

- (1) Complete patch matrix $\mathbf{P}$ has excessive patches $n_p = n \times (H - k + 1) \times (W - k + 1)$ while the patch vector length k^2 is small. That, it is unnecessary to train on a complete patch matrix.
- (2) When n_p is large, training U-ELM-AE is inefficient.
- (3) The mean patch can promote the elimination of illumination change.

After learning the convolutional kernels according to Algorithm 5, the same network structure is built with the original LRF-ELM. The final output from the last pooling or convolution layer is regarded as $\mathbf{F}$, a four-dimensional feature map.

Algorithm 5 Learning convolutional kernels with U-ELM-AE.

Input:

- The image samples $\mathbf{X}$;
- The number of target feature maps L ;
- The size of the local-receptive-field k .

Output:

- The learned convolutional kernels $\mathbf{A}$.
 - 1. For the i -th (i starts from 1 and ends at n) sample $\mathbf{x}_i \in \mathbb{R}^{H \times W}$, we use sliding window with the size of $k \times k$ to extract image patches. Each patch $\mathbf{p}_i^j$ follows the size of $k \times k$. We reshape $\mathbf{p}_i^j$ into a vector and stack all patch vectors to form the matrix $\mathbf{p}_i$. As the number of patches of i -th sample is $[(H - k + 1) \times (W - k + 1)]$, that the size of the two-dimensional matrix $\mathbf{p}_i$ should be $[((H - k + 1) \times (W - k + 1))] \times k^2$.
 - 2. Compute the mean values of all row vectors of $\mathbf{p}_i$ and remove accordingly.
 - 3. Repeat step (1-2) until $i = n$. Formally, the final patch matrix of all samples is represented by $\mathbf{P} = [\mathbf{p}_1, \dots, \mathbf{p}_{n_p}]$. The n_p denotes the number of samples used for learning.
 - 4. Calculate the mean patch matrix of $\mathbf{P}$, we get $\bar{\mathbf{P}} \in \mathbb{R}^{[(H - k + 1) \times (W - k + 1)] \times k^2}$.
 - 5. The $\bar{\mathbf{P}}$ is finally used as the U-ELM-AE input with L hidden neurons to learn orthogonal transformation matrix β .
 - 6. The $\mathbf{A}$ is the transpose of β .
-

$\mathbf{F}$ is re-organized by reshaping it to the two-dimensional matrix $\mathbf{H}$ for the classifier. The first dimension of $\mathbf{H}$ corresponds to n samples, and the second dimension represents all the features of one image. The resulted LRF-ELM variant with U-ELM-AE is denoted as LRF-ELM_U.

5.3.2 Non-Gradient Convolutional Neural Network with U-ELM-AE

A detailed discussion of the NG-CNN-related methods is presented in the previous chapter. The NG-CNNs utilize PCA, linear ELM-AE, nonlinear ELM-AE, or R-ELM-AE as the filter learning method. Although the R-ELMNet (NG-CNN variant with R-ELM-AE) achieved the best performance on various image classification datasets, it still requires the additional hyper-parameter α in Equation

4.9. As presented in the previous chapter, it is recommended to find the best hyper-parameter combinations of R-ELMNet as follows:

- (1) Run the random grid search on all hyper-parameter combinations and select limited candidates.
- (2) Fine-tune candidates on training dataset or a smaller subset, which depends on the time efficiency and volume of the training dataset.

Hence the U-ELM-AE is introduced to NG-CNN (referred to as NG-CNN_U) and expected competitive performance with all mentioned NG-CNN-related methods, PCANet, ELMNet, H-ELMNet, and R-ELMNet. Based on that, the effectiveness of U-ELM-AE as the plug-and-play dimension reduction method is verified.

The overall pipeline of NG-CNN_U follows: 1) pre-processing, 2) filter learning with U-ELM-AE, 3) post-processing. Detailed algorithm is illustrated in Chapter 2.3.4 and Chapter 4.

5.4 Experiments

5.4.1 Experimental Setup

The effectiveness and efficiency of the U-ELM-AE for dimension reduction were evaluated on image classification datasets, including Coil20 [109], Coil100 [110], and Fashion [101]. The experiments compare PCA, linear ELM-AE, nonlinear ELM-AE, and SB-ELM-AE. Note that SB-ELM-AE in Chapter 3 was not originally proposed for the dimension reduction task. The range of hyper-parameter L is [100, 200, 300, 400] and [100, 200, 300, 400, 500, 600, 700]. The former list is only considered on Coil20 dataset, as the training split contains 420 samples. The range of ℓ_2 -regularization factor for nonlinear ELM-AE is $[1e-8, \dots, 1e8]$. For SB-ELM-AE, the δ of SB-ELM-AE came from [0.1, 1, 10]. The pruning scheme was not

applied in the compared SB-ELM-AE. The best hyper-parameters were selected with strategy: 1) running random grid search with three-folder cross-validation on training split, 2) fine-tuning hyperparameters from the first step.

Based on the feature learning/dimension reduction methods, the reduced features were evaluated with linear SVM and ELM classifiers. The number of hidden neurons of the ELM classifier was chosen from [6000, 8000, 10000, 12000]. The corresponding ℓ_2 -regularization factor fell in [1e-3, 1e-2, 1e-1, 1, 1e1, 1e2, 1e3].

The results of plug-and-play role of U-ELM-AE in LRF-ELM and NG-CNN were tested on Coil20 [109], Coil100 [110], Fashion [101] and Letters [102]. The two latter datasets bring the challenge to big data classification.

Experiments were tested on the Ubuntu platform configured with a 28-core CPU. Note that the SB-ELM-AE in a parallel method was implemented, as proposed in Chapter 3. Therefore, SB-ELM-AE utilized full usage of CPU, while other methods mainly occupied one CPU core.

5.4.2 Datasets Preparation

The images classification datasets include Coil20 [109], Coil100 [110] Fashion [101] and Letters [102]. The data partition scheme and introduction are enumerated as below:

- (1) Coil20 [109] dataset brings object recognition task, which has 20 categories. All pictures were captured with a camera to cover 360 degrees. The whole dataset was split into training and testing partitions. The former has 420 samples, and the latter retains the remaining 1020 samples. To verify the generalization capability, training and testing subsets share no adjacent camera angle. The samples of each category were sorted first according to the camera angle. Then a sliding window with a size of 21 was adapted to extract the training subset at a random starting index. The left images formed the

testing subset. The actions were repeated five times. Samples were resized into a size of 28×28 .

- (2) Coil100 [110] is a bigger dataset than Coil20. It contains 100 categories. All the pictures were captured with the same method of Coil20. Hence the dataset partition and pre-processing followed the same scheme with Coil20. The final training subset has 2100 samples, and the testing subset holds the remaining 5100.
- (3) Fashion [101] dataset consists of a training set of 60,000 samples and a testing set of 10,000 samples. Each sample is 28×28 grayscale image and comes from 10 classes, such as trouser, shirt, and so on.
- (4) Letters [102] is a bigger image dataset on character classification, which has 124,800 samples for training and 20,800 samples for testing. It contains 26 classes, and each class includes upper and lower cases. Data samples follow 28×28 grayscale shape.

The performance comparison of dimension reduction was tested on Coil20, Coil100, Fashion, and a subset of Fashion denoted as Fashion(1), which contains 2000 training samples and 2000 testing samples.

The classification accuracies of extensive applications, including LRF-ELM_U and NG-CNN_U, were conducted on Coil20, Coil100, Fashion, and Letters. Meanwhile, the data pre-processing strategy varies in different scenarios.

- (1) *Dimension Reduction*: Each sample was normalized to have zero mean and unit variance.
- (2) *Extensive Applications*: Values were directly re-scaled between 0 and 1.

5.4.3 Sensitivity to Hyper-Parameter L

The experiment of dimension reduction was conducted with all L selections. The Figures 5.1, 5.2 plot the curves with the linear SVM classifier. Results with ELM classifier are reflected in Figures 5.3, 5.4. The linear ELM-AE is generally better than PCA for each L choice. Apparently, increasing the number of features can improve the performance of SB-ELM-AE. With the ELM classifier, U-ELM-AE shows the best accuracies.

For better access, the mean accuracy histogram and standard deviation are presented in Figures 5.5, 5.6, 5.7, and 5.8. The mean accuracy was computed by averaging the classification accuracies of all L choices for each method individually. The standard deviation was calculated accordingly. These figures can illustrate the performance sensitivity to L , which matters if integrating into other models, as we expect a method with stable and, meanwhile, high performance. The overall performance of U-ELM-AE can be highlighted. Nonlinear ELM-AE and SB-ELM-AE only exceed the mean accuracy of U-ELM-AE on Coil100 with linear SVM. Nevertheless, the standard deviation of U-ELM-AE is lower. SB-ELM-AE illustrates the largest standard deviation that is most sensitive to L . Increasing L of SB-ELM-AE may improve performance further, as illustrated in tables, while this chapter mainly focuses on dimension reduction.

5.4.4 Performance Comparison for Dimension Reduction

Firstly, the effectiveness comparison of U-ELM-AE, PCA, linear ELM-AE, nonlinear ELM-AE, and SB-ELM-AE was conducted with a linear SVM classifier. The reported mean accuracy, best accuracy, the number of features, and training time are illustrated in Table 5.2. It mainly demonstrates the linear separability after dimension reduction. U-ELM-AE performs best on Coil10 and Fashion(1). SB-ELM-AE shows the best accuracy on Coil100, and nonlinear ELM-AE illustrates the highest accuracy on Fashion. Although U-ELM-AE is overwhelmed by

FIGURE 5.1: Illustration of the influence of feature dimension on Coil20 and Coil100 with linear SVM classifier. Note that the upper figure's maximum features are 400 as training split on Coil20 only contains 420 samples. Considering PCA and linear ELM-AE requirement, the maximum feature-length was set to 400.

FIGURE 5.2: Illustration of the influence of the feature dimension on Fashion(1) and Fashion with linear SVM classifier.

SB-ELM-AE or nonlinear ELM-AE on Coil100 or Fashion(1), it still shows better performance than PCA and linear ELM-AE. Note that three methods are free of additional hyper-parameters.

Secondly, the performance comparison was evaluated for U-ELM-AE, PCA, linear ELM-AE, nonlinear ELM-AE, and SB-ELM-AE with ELM classifier. The configuration of the ELM classifier contains sigmoid activation and ℓ_2 regularization. Table 5.3 presents corresponding mean accuracy, best accuracy, the number

FIGURE 5.3: Illustration of the influence of the feature dimension on Coil20 and Coil100 with ELM classifier. The maximum feature dimension was set to 400 on Coil20.

FIGURE 5.4: Illustration of the influence of the feature dimension on Fashion(1) and Fashion with an ELM classifier.

of features, and training time. We may conclude that U-ELM-AE demonstrates the best results on Coil20, Coil100, Fashion(1), and Fashion.

More detailed comparisons and discussions can be drawn from several views:

- (1) *PCA vs. linear ELM-AE*: Both the Table 5.2 and Table 5.3 reflect that linear ELM-AE is better than PCA.

FIGURE 5.5: Mean accuracy and standard deviation over all feature dimension choices for each method (U-ELM-AE, PCA, linear ELM-AE, nonlinear ELM-AE and SB-ELM-AE) on Coil20 and Coil100 with the linear SVM classifier. The mean accuracy was calculated over the performance of each L and shown by the histogram. The standard deviation is shown by the error bar.

FIGURE 5.6: Mean accuracy and standard deviation over all feature dimension choices for each method (U-ELM-AE, PCA, linear ELM-AE, nonlinear ELM-AE, and SB-ELM-AE) on Fashion(1) and Fashion with the linear SVM classifier. The mean accuracy is shown by the histogram. The standard deviation is illustrated by the error bar.

FIGURE 5.7: Mean accuracy and standard deviation over all feature dimension choices for each method (U-ELM-AE, PCA, linear ELM-AE, nonlinear ELM-AE, and SB-ELM-AE) on Coil20 and Coil100 with the ELM classifier. The mean accuracy is shown by the histogram. The standard deviation is illustrated by the error bar.

FIGURE 5.8: Mean accuracy and standard deviation over all feature dimension choices for each method (U-ELM-AE, PCA, linear ELM-AE, nonlinear ELM-AE, and SB-ELM-AE) on Fashion(1) and Fashion with the ELM classifier. The mean accuracy is shown by the histogram. The standard deviation is illustrated by the error bar.

- (2) *SB-ELM-AE*: Note SB-ELM-AE was not initially proposed for dimension reduction. Generally, SB-ELM-AE can benefit from a larger number of hidden neurons. We may observe that the optimum number of features of SB-ELM-AE in tables are commonly larger than other methods, which justifies above the statement.
- (3) *ELM vs. linear SVM*: Apparently, ELM performs better than linear SVM not only on the accuracy performance also on the training efficiency. Linear SVM requires a much longer time for learning.
- (4) *U-ELM-AE vs. PCA vs. linear ELM-AE*: These methods are clustered due to free of additional hyper-parameters. Only the number of features is considered, unlike nonlinear ELM-AE and SB-ELM-AE. Meanwhile, the performances can be sorted from best to worst with the sequence: U-ELM-AE, linear ELM-AE, and PCA, across all the experiments.
- (5) *Learning efficiency*: ELM classifier spends less time than linear SVM. With a linear SVM classifier, PCA requires the longest training time on Fashion. While with the ELM classifier, SB-ELM-AE is the most time-consuming method on Fashion. Note the SB-ELM-AE was implemented with batch-size training, which utilizes the full usage of CPU cores and memory.

5.4.5 Performance Comparison as Plug-and-Play Role in LRF-ELM and NG-CNN

The effectiveness of U-ELM-AE as the integrated feature learning method was evaluated within two frameworks, including LRF-ELM and NG-CNN. The resulted methods are referred to as LRF-ELM_U and NG-CNN_U. The classification accuracies are listed in Table 5.4. The best performances were emphasized with bold font. Reported accuracies of LRF-ELM_U are better than LRF-ELM. Generally, the LRF-ELM framework shows less competitive results compared to NG-CNNs. NG-CNN_U presents consistent improvement compared to PCANet, ELMNet, and

Datasets	Methods	Testing Accuracy (%)	Best Acc. (%)	Features	Training Time (s)
Coil20	U-ELM-AE	75.88 (± 0.18)	76.36	400	0.39(± 0.01)
	PCA	74.31(± 0.12)	74.58	300	0.33(± 0.01)
	ELM-AE (linear)	74.36(± 0.52)	75.28	400	0.41(± 0.02)
	ELM-AE (nonlinear)	74.51(± 2.18)	75.39	300	0.44(± 0.03)
	SB-ELM-AE	75.19(± 1.23)	76.24	400	0.45(± 0.02)
Coil100	U-ELM-AE	61.77(± 0.31)	62.04	200	4.87(± 0.04)
	PCA	53.24(± 0.18)	53.53	700	5.15(± 0.02)
	ELM-AE (linear)	54.22(± 0.47)	54.71	100	4.41(± 0.03)
	ELM-AE (nonlinear)	63.87(± 0.46)	64.38	500	4.79(± 0.10)
	SB-ELM-AE	65.37 (± 0.36)	65.45	700	6.04(± 0.63)
Fashion(1)	U-ELM-AE	81.35 (± 0.39)	81.60	300	1.50(± 0.04)
	PCA	78.95(± 0)	78.95	100	0.95(± 0.03)
	ELM-AE (linear)	81.22(± 0.32)	81.38	100	0.80(± 0.03)
	ELM-AE (nonlinear)	80.61(± 0.94)	81.45	700	3.31(± 0.04)
	SB-ELM-AE	81.33(± 0.54)	81.85	700	9.93(± 0.04)
Fashion	U-ELM-AE	85.01(± 0.25)	85.22	400	171.17(± 2.35)
	PCA	83.09(± 0)	83.09	400	299.81(± 1.28)
	ELM-AE (linear)	84.02(± 0.09)	84.15	400	278.27(± 2.06)
	ELM-AE (nonlinear)	86.35 (± 0.36)	86.78	600	162.24(± 2.04)
	SB-ELM-AE	84.83(± 0.23)	85.11	700	210.74(± 1.32)

TABLE 5.2: The testing accuracy and training time on Coil20, Coil100, and Fashion datasets of U-ELM-AE, PCA, ELM-AE (linear), ELM-AE (nonlinear), and SB-ELM-AE using the linear SVM classifier.

Datasets	Methods	Testing Accuracy (%)	Best Acc. (%)	Features	Training Time (s)
Coil20	U-ELM-AE	80.46 (± 0.34)	80.88	200	0.08(± 0.01)
	PCA	78.78(± 0.78)	79.95	300	0.08(± 0.01)
	ELM-AE (linear)	79.96(± 1.40)	80.17	400	0.08(± 0.01)
	ELM-AE (nonlinear)	79.74(± 0.92)	81.56	200	0.07(± 0.01)
	SB-ELM-AE	80.17(± 0.54)	80.98	400	0.98(± 0.02)
Coil100	U-ELM-AE	68.38 (± 0.46)	68.79	200	2.02(± 0.01)
	PCA	67.87(± 0.26)	68.06	100	1.18(± 0.01)
	ELM-AE (linear)	67.93(± 0.22)	68.13	100	2.69(± 0.01)
	ELM-AE (nonlinear)	67.33(± 0.53)	67.92	300	2.23(± 0.01)
	SB-ELM-AE	68.14(± 0.35)	68.58	700	6.65(± 0.02)
Fashion(1)	U-ELM-AE	84.11 (± 0.34)	84.44	100	1.47(± 0.03)
	PCA	82.32(± 0.18)	82.45	100	1.54(± 0.02)
	ELM-AE (linear)	82.95(± 0.25)	83.67	500	1.53(± 0.01)
	ELM-AE (nonlinear)	82.29(± 0.38)	83.11	500	1.85(± 0.03)
	SB-ELM-AE	82.71(± 0.49)	83.13	600	8.56(± 0.03)
Fashion	U-ELM-AE	88.31 (± 0.16)	88.45	200	59.01(± 0.26)
	PCA	87.51(± 0.16)	87.76	200	59.15(± 0.23)
	ELM-AE (linear)	87.65(± 0.09)	87.68	100	61.34(± 0.51)
	ELM-AE (nonlinear)	87.53(± 0.19)	87.69	700	57.86(± 0.27)
	SB-ELM-AE	86.53(± 0.24)	86.84	700	72.61(± 0.61)

TABLE 5.3: The testing accuracy and training time on Coil20, Coil100, and Fashion datasets of U-ELM-AE, PCA, ELM-AE (linear), ELM-AE (nonlinear), and SB-ELM-AE using the ELM classifier.

TABLE 5.4: Mean accuracy comparison as a plug-and-play role in LRF-ELM and NG-CNN. Methods are divided into four groups: single-layer ELM classifier, LRF-ELM related methods, NG-CNNs, and CNN models.

Methods	Coil20	Coil100	Fashion	Letters	map^4
ELM	77.37	63.67	87.43	85.96	78.61
LRF-ELM	79.71	67.42	88.45	90.27	81.46
LRF-ELM _U	81.43	69.86	88.56	90.75	82.65
PCANet[35]	80.72	71.08	91.08	93.34	84.06
ELMNet[36]	81.17	71.72	91.17	93.25	84.33
H-ELMNet[37]	81.09	71.65	91.21	93.46	84.35
R-ELMNet	84.28	73.31	91.32	93.59	85.63
NG-CNN _U	81.76	72.24	91.89	93.67	84.89
CNN-2 ¹	75.46	65.85	90.99	93.26	81.39
CNN-3 ²	74.45	59.67	90.11	93.49	79.43
CNN-4 ³	74.92	62.17	90.02	93.31	80.11
EDEN[113]	-	-	90.60 [*]	-	-

¹ A 2-Conv CNN model, same to Chapter 4.

² A 3-Conv CNN model, same to Chapter 4.

³ A 4-Conv CNN model, same to Chapter 4.

⁴ Mean average performance across all datasets.

^{*} Directly cited from the original paper.

H-ELMNet. R-ELMNet outperforms NG-CNN_U on Coil20 and Coil100. Nevertheless, NG-CNN_U exceeds R-ELMNet on big datasets.

Compared to orthogonal random projection, PCA, linear ELM-AE in LRF-ELM or NG-CNN, which all are free of additional hyper-parameters, U-ELM-AE presents the most competitive performance.

5.5 Conclusion

In this chapter, the drawbacks of current ELM-AE variants are analyzed first. The ELM-AE, as a dimension reduction method, is not as popular as PCA or other related methods. The most frequently used ELM-AE variant is nonlinear ELM-AE. While nonlinear ELM-AE commonly requires additional normalization or rescaling operations to overcome its uncertain output scale problem. Also, one needs to adjust the ℓ_2 -regularization hyper-parameter. Due to all these shortages, nonlinear ELM-AE is not as popular as the ELM classifier. Inspired by the success of

LRF-ELM, the importance of orthogonal projection is highlighted. By introducing the orthogonal condition to unknown weights, an analytical solution of the novel ELM-AE variant is presented, referred to as Unified ELM-AE, which presents the effectiveness of its orthonormal basis. The U-ELM-AE achieves the minimum reconstruction error under the condition of orthogonal projection from output to hidden activations. U-ELM-AE is also bridged with popular problems, such as RV coefficient and matrix inner-product. Experiments on dimension reduction on image datasets verify the effectiveness and efficiency.

Furthermore, two scenarios where U-ELM-AE may act as the plug-and-play dimension reduction/feature learning methods are illustrated: 1)LRF-ELM, 2)NG-CNN. The U-ELM-AE can be conveniently integrated into the two frameworks to present better performance or implementation efficiency. Experiments on image classification datasets show the competitive result with a single-layer ELM classifier, LRF-ELM, NG-CNNs, and CNNs.

Chapter 6

Stacking Projection Regularized ELM-AE with U-ELM-AE for ML-ELM

U-ELM-AE is proposed for dimension reduction-related feature learning. It requires a dimension/feature expansion method to develop a stacked multi-layer fully connected network. Nevertheless, the experiments show that the existing SB-ELM-AE and ELM-AE fail to bring effectiveness. Hence, the Projection Regularized ELM-AE (PR-ELM-AE) is proposed as the first ELM-AE for dimension expansion with a regularization term to restrict the output scale. U-ELM-AE then performs representation learning based on the first ELM-AE. The overall structure achieves better performance compared to ML-ELM, H-ELM, and SBAE-ELM.

6.1 Proposed Method

Given the input $\mathbf{X} \in \mathbb{R}^{n \times d}$ and the number of hidden neurons L , the activation output of single ELM is denoted with $\mathbf{H} \in \mathbb{R}^{n \times L}$. The nonlinear ELM-AE learns output weights β with Equation 2.25 and transforms data with $\mathbf{X}\beta^T$. As illustrated in Figure 4.1, one drawback of nonlinear ELM-AE is the uncertain and unmatched dimension scale of $\mathbf{X}\beta^T$. The U-ELM-AE with the orthogonal β ($\beta\beta^T = \mathbf{I}$) can fulfill the requirement of bounded output while it can only handle the case $L \leq d$. A straightforward design is to stack nonlinear ELM-AE first for dimension expansion and utilize U-ELM-AE for dimension reduction. While U-ELM-AE requires the input value $\mathbf{X}$ should be comparable with hidden activations $\mathbf{H}$ (usually falls into $[-1, 1]$ or $[0, 1]$), as β_U comes from the covariance matrix between $\mathbf{X}$ and $\mathbf{H}$. That U-ELM-AE can not follow a nonlinear ELM-AE or SB-ELM-AE directly and simply act as the second ELM-AE. Although the output feature of the first ELM-AE can be normalized, there is no observed consistent improvement by such skills. Hence a Projection Regularized ELM-AE (PR-ELM-AE) is proposed for dimension expansion.

To constrain the value scale of $\mathbf{X}^{proj}$, the trace regularization is introduced as below:

$$\begin{aligned} \text{Minimize : } & \text{trace}((\mathbf{X}^{proj})^T \mathbf{X}^{proj}) \\ & = \text{trace}(\beta \mathbf{X}^T \mathbf{X} \beta^T). \end{aligned} \tag{6.1}$$

The objective 6.1 forces β to approximate zero. Thus, it can only act as regularization to avoid degeneration. The motivation of introducing 6.1 comes from the disadvantage of linear least squares regression. That the according objective $\text{trace}\{[f(\mathbf{X}, \beta) - \mathbf{Y}]^T [f(\mathbf{X}, \beta) - \mathbf{Y}]\}$ is sensitive to outliers, where $f(\cdot, \cdot)$ denotes a linear function. Thus, it can effectively incorporate with reconstruction error to

avoid over-expected feature scale. The overall target function follows:

$$\begin{aligned} \text{Minimize : } & \text{trace}((\mathbf{H}\boldsymbol{\beta} - \mathbf{X})^T(\mathbf{H}\boldsymbol{\beta} - \mathbf{X})) + \\ & \gamma \text{trace}(\boldsymbol{\beta}\mathbf{X}^T\mathbf{X}\boldsymbol{\beta}^T), \end{aligned} \quad (6.2)$$

where γ is the factor on the importance of regularization, and only a small set $[0, 0.1, 0.2, \dots, 1]$ is considered for implementation efficiency.

The derivatives with respect to $\boldsymbol{\beta}$ can be derived as follows:

$$\frac{\partial \text{Loss}}{\partial \boldsymbol{\beta}} = \mathbf{H}^T \mathbf{H} \boldsymbol{\beta} + \gamma \boldsymbol{\beta} \mathbf{X}^T \mathbf{X} - \mathbf{H}^T \mathbf{X}. \quad (6.3)$$

Let derivatives be zero, and then the target objective is a Sylvester equation. The transformation of PR-ELM-AE follows the same to other ELM-AE variants. The second ELM-AE for stacking fully connected multi-layer ELM is simply the standard U-ELM-AE. Thus, the overall structure of combined multi-layer ELM can be represented with d - $L1$ - $L2$ - $L3$ - t , where d is the input dimension, $L1$ is the number of hidden neurons of PR-ELM-AE, $L2$ denotes the number of hidden neurons of U-ELM-AE, $L3$ represents the hidden nodes of ELM classifier and t is the target dimension. ML-ELM_U represents the abbreviation of that network structure. The stacking procedure is shown in Figure 6.1.

FIGURE 6.1: Illustration of the ML-ELM_U 's network structure.

6.2 Experiments

The experiments were conducted on Coil20, Coil100, and five datasets from Openml¹. The dataset partition scheme is the same as previous chapters for consistent comparison. Each Openml dataset was divided into training and testing split with a proportion of 0.7 and 0.3, respectively. While on Coil20 and Coil100, 21 samples with adjacent camera angles of the same category were selected for training from random starting camera angle indexes, and the remaining data fell into the testing split. The action was repeated five repeats. The big datasets, Fashion and Letters, are the same as the previous chapters. All the experiments were coded by Python 2.7 and tested on the platform with a 28-core CPU.

To verify the effectiveness of PR-ELM-AE for regularized output. The compared methods ML-ELM₁ and ML-ELM₂ were developed. ML-ELM₁ denotes that the first ELM-AE is nonlinear ELM-AE without normalized output, followed by a U-ELM-AE. In contrast, ML-ELM₂ normalizes the output of the first nonlinear ELM-AE.

The experimental results are illustrated in Table 6.1. ML-ELM_U outperforms ML-ELM variants. It verifies the necessity of developing proper feature expansion methods beyond past ELM-AE.

The sensitivity analysis on γ is illustrated in Figure 6.2, the hyper-parameter γ was chosen from $[0, 0.1, \dots, 1]$. Consistent improvements are shown for $\gamma > 1$ than cases $\gamma = 0$ on all benchmark datasets. Simply stacking nonlinear ELM-AE with and without normalization fails to present elevation, verifying the necessity of PR-ELM-AE for feeding feature into U-ELM-AE.

The selected network structure is shown in Table 6.2. For example, the structure 784-1200(0.2)-500-8000-100 on Coil100 matches d -L1-L2-L3- t , where the input dimension is 784, the first hidden layer has 1200 neurons with $\gamma=0.2$, the

¹www.openml.org

second hidden nodes are 500, the ELM classifier is with 8000 hidden neurons, and the output dimension is 100.

To illustrate the overall comparison with the previous methods, Table 6.3 presents results on image datasets and non-image datasets. Obviously, the ML-ELM_U performs best on the non-image datasets. R-ELMNet is the best method for Coil20 and Coil100. The VGG, the very deep CNN, apparently has the state-of-the-art accuracy on big datasets. Nevertheless, considering the learning efficiency in Table 6.4, the NG-CNN_U requires much less training time. Meanwhile, the fully connected ELMs have evidently better training efficiency.

Datasets	Methods	Testing Accuracy (%)	Std.	Training Time (s)
Cmc	ML-ELM_U	57.19	1.37	0.27
	ML-ELM ₁	53.61	2.24	0.24
	ML-ELM ₂	48.14	3.01	0.25
Abalone	ML-ELM_U	66.71	0.52	0.56
	ML-ELM ₁	63.89	1.07	0.47
	ML-ELM ₂	65.65	0.67	0.48
Digits	ML-ELM_U	98.87	0.61	0.93
	ML-ELM ₁	97.41	0.56	0.91
	ML-ELM ₂	97.92	0.38	0.91
Diabetes	ML-ELM_U	79.12	1.42	0.07
	ML-ELM ₁	75.39	2.26	0.03
	ML-ELM ₂	76.52	2.51	0.04
Isolet	ML-ELM_U	95.15	0.34	7.62
	ML-ELM ₁	92.09	0.28	1.41
	ML-ELM ₂	92.95	0.49	1.42
Coil20	ML-ELM_U	82.53	1.24	10.75
	ML-ELM ₁	80.52	1.11	0.55
	ML-ELM ₂	80.53	1.81	0.56
Coil100	ML-ELM_U	68.44	0.41	6.42
	ML-ELM ₁	65.20	0.48	2.98
	ML-ELM ₂	68.28	0.58	2.99

TABLE 6.1: Table shows the testing accuracy and training time comparison on several datasets. ML-ELM₁ and ML-ELM₂ represent applying a nonlinear ELM-AE without and with normalization, respectively, as the first ELM-AE followed by a U-ELM-AE.

FIGURE 6.2: Illustration of the effect of γ on the classification accuracy.

Datasets	Network Structure
Cmc	9-20(0.2)-10-3000-3
Abalone	8-50(0.2)-10-3000-3
Digits	64-100(0.3)-50-3000-2
Diabetes	8-50(0.3)-10-3000-2
Isolet	617-1200(1.0)-500-8000-26
Coil20	784-2000(0.7)-400-8000-20
Coil100	784-1200(0.2)-500-8000-100
Fashion	784-1200(0.2)-500-8000-10
Letters	784-1200(0.9)-500-8000-26

TABLE 6.2: Table shows the network structure of ML-ELM_U. For example, the structure 784-1200(0.2)-500-8000-100 on Coil100 illustrates the input dimension, the first hidden layer (γ), the second hidden layer, the third layer, and the output dimension, respectively.

6.3 Conclusion

To explore the possibility of transferring U-ELM-AE into more generalized scenarios besides dimension reduction, the PR-ELM-AE is proposed to expand dimension first. Compared to feeding the transformed feature from nonlinear ELM-AE directly, incorporating PR-ELM-AE into fully connected multi-layer ELM enables a consistent improvement based on U-ELM-AE. Experiments on several datasets verify the effectiveness and efficiency compared to ML-ELM, H-ELM, and SBAE-ELM.

TABLE 6.3: Mean accuracy comparison on scalable classification datasets.

Methods	Cmc	Abalone	Digits	Diabetes	Isolet	Coil20	Coil100	Fashion	Letters	map ¹
NMF	54.65	64.65	95.14	74.28	94.75	73.04	58.47	86.04	81.87	75.88
PCA	48.33	64.41	96.67	71.29	92.06	78.78	67.87	87.51	86.77	77.08
ML-ELM	55.42	64.93	98.14	76.12	92.18	80.49	67.33	87.32	88.25	78.91
H-ELM	54.71	65.17	98.29	77.04	90.78	82.09	68.01	87.53	87.97	79.07
SBAE-ELM	55.48	65.65	99.07	78.01	93.20	82.38	68.14	87.57	88.32	79.76
ML-ELM _U	57.19 ²	66.71	98.87	79.12	95.15	82.53	68.44	89.72	88.49	80.69
SAE	49.17	63.27	97.95	76.34	92.41	73.77	67.74	87.42	86.81	77.21
VAE	45.88	64.37	96.06	72.52	91.23	75.27	65.22	86.83	90.14	76.39
PCANet	-	-	-	-	-	80.72	71.08	91.08	93.34	84.06
ELMNet	-	-	-	-	-	81.17	71.72	91.17	93.25	84.33
H-ELMNet	-	-	-	-	-	81.09	71.65	91.21	93.46	84.35
CNN-2	-	-	-	-	-	75.46	65.85	90.99	93.26	81.39
CNN-3	-	-	-	-	-	74.45	59.67	90.11	93.49	79.43
CNN-4	-	-	-	-	-	74.92	62.17	90.02	93.31	80.11
EDEN[113]	-	-	-	-	-	-	-	90.60	-	-
VGG	-	-	-	-	-	-	-	92.21	94.13	-
R-ELMNet	-	-	-	-	-	84.28	73.31	91.32	93.59	85.63
NG-CNN _U	-	-	-	-	-	81.76	72.24	91.89	93.67	84.89

¹ Mean average performance across all available datasets.² The best performance uses bold-face.

TABLE 6.4: Training Time (minutes) Comparison on Big Datasets.

Methods	Fashion	Letters
NG-CNN* ¹	31	73
H-ELMNet	247	445
CNN-2	39	115
CNN-3	47	162
CNN-4	55	227
VGG	521	1108
ML-ELM	1.0	1.6
H-ELM	3.2	7.7
SBAE-ELM	1.1	1.9
ML-ELM _U	1.0	1.8
SAE	492	1017
VAE	19	30

¹ NG-CNN* denotes the summary for PCANet, ELMNet, and R-ELMNet, as they share the same time-efficiency. H-ELMNet takes much longer training time due to its complexity.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

This thesis builds Extreme Learning Machine Auto-Encoder (ELM-AE) variants based on the original ELM-AE structure for feature learning, dimension reduction, and extensive applications. Along with the timeline, the proposed ELM-AEs and ELMs follow the principal from complex to concise.

Chapter 3 investigates the sparse Bayesian inference-based ELM-AE, namely, sparse Bayesian ELM-AE (SB-ELM-AE). It develops a proper probability framework and presents improved performance compared to ℓ_2 -regularized ELM-AE and ℓ_1 -regularized ELM-AE. Moreover, to overcome the training inefficiency of SB-ELM-AE with multiple outputs dimensions, a parallel learning pipeline is addressed for efficiency. Multi-Layer ELM (ML-ELM) and Hierarchical ELM (H-ELM) are formed via stacking ℓ_2 -ELM-AE and ℓ_1 -ELM-AE, respectively. Based on a similar multi-layer structure, the sparse Bayesian Auto-Encoding-based ELM (SBAE-ELM) is proposed via SB-ELM-AE.

Chapter 4 explores the research focus on the ELM-AE-based convolutional kernel learning framework for unsupervised feature learning. Non-Gradient Convolutional Neural Network (NG-CNN) is the abbreviation for the two-layer unsupervised feature learning pipeline, including three stages: pre-processing, filter learning, and post-processing. NG-CNN summarizes the overall structure of Principal Component Analysis Network (PCANet), and the following ELM Network (ELMNet) and Hierarchical ELMNet (H-ELMNet). The detailed discussion about previous applications of ELM-AE variants in NG-CNN is addressed. Accordingly, the Regularized ELM-AE (R-ELM-AE) is proposed, specified for NG-CNN, to form the R-ELMNet. Compared to H-ELMNet, R-ELMNet removes Local Contrast Normalization (LCN) and time-consuming whitening pre-processing and achieves a minimum implementation level the same as PCANet and ELMNet. Experiments show the performance improvement of unsupervised feature learning on various volumes of image datasets. Moreover, R-ELMNet outperforms related CNN models, which might be highlighted as NG-CNN only utilizes a linear SVM classifier.

Chapter 5 summarizes the most desired properties of ELM-AE variant, including nonlinear ELM random mapping, learning efficiency, free of additional hyper-parameters or normalization methods, and orthogonal projection. Based on the orthogonal Procrustes solution, the Unified ELM-AE (U-ELM-AE) for dimension reduction is proposed with an analytical solution. Experiments illustrate the improvement in dimension reduction tasks compared with PCA, nonlinear ELM-AE, linear ELM-AE, and SB-ELM-AE. Considering the scenarios where dimension reduction may act as feature learning, Chapter 5 also extends U-ELM-AE into Local Receptive Fields-based ELM (LRF-ELM) and NG-CNN. LRF-ELM directly uses orthogonal random convolutional kernels. It is shown that U-ELM-AE can replace the role of randomly generated kernels efficiently and effectively. Meanwhile, as shown in Chapter 4, the U-ELM-AE can be directly integrated and achieve improved performance compared to related methods.

As shown in Chapter 5, although U-ELM-AE has achieved the most competitive performance for dimension reduction, one may notice that SB-ELM-AE or

other methods could improve performance further with expanded dimension. The U-ELM-AE only holds solutions, while the number of hidden neurons is smaller than output neurons. Thus, Chapter 6 presents the simple multi-layer pipeline for U-ELM-AE to handle with dimension expansion problem: 1) a Projection Regularized ELM-AE (PR-ELM-AE) is proposed, as the first layer, for dimension expansion. 2) the U-ELM-AE is then applied for feature learning. With that framework, the U-ELM-AE can be extended to the fully connected multi-layer ELM efficiently and effectively.

7.2 Future Work

The future focus should cover the followings:

- (1) Optimize the network design of NG-CNN_U. Some trials, such as depth-wise/spatial pooling and channel pruning, should be useful for improving effectiveness and efficiency.
- (2) Other ELM-AE applications, such as clustering and sparse coding, should be the next attractive scenarios.
- (3) The U-ELM-AE may be valuable to combine with deep neural networks. For example, it may work as an additional objective that applies orthogonal and reconstruction regularization.

List of Author's Publications

Journal Papers

1. Tu E, **Zhang G**, et al. Exploiting AIS Data for Intelligent Maritime Navigation: A Comprehensive Survey From Data to Methodology. *IEEE Transactions on Intelligent Transportation Systems*, 2017.
2. **Zhang G**, et al. Unsupervised Feature Learning with Sparse Bayesian Auto-Encoding based Extreme Learning Machine. *International Journal of Machine Learning and Cybernetics*, 2020(3).
3. **Zhang G**, et al. R-ELMNet: Regularized Extreme Learning Machine Network. *Neural Networks*, 2020.
4. **Zhang G**, et al. Unified Extreme Learning Machine Auto-Encoder for Dimension Reduction and Extensive Applications. Under Review by *Cognitive Computation*.

Conference Papers

1. **Zhang G**, et al. Stable and improved generative adversarial nets (GANS): A constructive survey. *IEEE International Conference on Image Processing (ICIP)*. IEEE, 2017: 1871-1875.
2. **Zhang G**, et al. Sparse Bayesian Learning for Extreme Learning Machine Auto-encoder. *International Conference on Extreme Learning Machine*. Springer, Cham, 2018: 319-327.

3. Tu E, **Zhang G**, et al. A theoretical study of the relationship between an ELM network and its subnetworks. 2017 International Joint Conference on Neural Networks (IJCNN). IEEE, 2017: 1794-1801.
4. Cui D, **Zhang G**, et al. Compact Feature Representation for Image Classification Using ELMs. Proceedings of the IEEE International Conference on Computer Vision. 2017: 1015-1022.

Bibliography

- [1] Michael I Jordan. Serial order: A parallel distributed processing approach. In *Advances in psychology*, volume 121, pages 471–495. Elsevier, 1997. [1](#)
- [2] David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988. [1](#), [8](#)
- [3] Guang Bin Huang, Chen Lei, and Chee Kheong Siew. *Universal approximation using incremental constructive feedforward networks with random hidden nodes*. 2006. [2](#), [8](#), [11](#)
- [4] R. Zhang, Y. Lan, G. B. Huang, and Z. B. Xu. Universal approximation of extreme learning machine with adaptive growth of hidden nodes. *IEEE Transactions on Neural Networks and Learning Systems*, 23(2):365, 2012. [8](#)
- [5] Guang Bin Huang and Lei Chen. Enhanced random search based incremental extreme learning machine. *Neurocomputing*, 71(16):3460–3468, 2008. [11](#)
- [6] Guang Bin Huang, Qin Yu Zhu, and Chee Kheong Siew. Extreme learning machine: Theory and applications. *Neurocomputing*, 70(1):489–501, 2006.
- [7] G. B. 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*, 42(2):513–529, 2012.
- [8] Guang Bin Huang. An insight into extreme learning machines: Random neurons, random features and kernels. *Cognitive Computation*, 6(3):376–390, 2014. [2](#), [8](#)

- [9] Zhu Hong You, Ying Ke Lei, Lin Zhu, Junfeng Xia, and Bing Wang. Prediction of protein-protein interactions from amino acid sequences with ensemble extreme learning machines and principal component analysis. *Bmc Bioinformatics*, 14(S8):S10, 2013. [2](#)
- [10] A.H. Nizar, Z.Y. Dong, and Y. Wang. Power utility nontechnical loss analysis with extreme learning machine method. *IEEE Transactions on Power Systems*, 23(3):946–955, 2008. [2](#)
- [11] Yuedong Song, Jon Crowcroft, and Jiaxiang Zhang. Automatic epileptic seizure detection in eegs based on optimized sample entropy and extreme learning machine. *Journal of neuroscience methods*, 210(2):132–146, 2012. [2](#)
- [12] Mahesh Pal, Aaron E Maxwell, and Timothy A Warner. Kernel-based extreme learning machine for remote-sensing image classification. *Remote Sensing Letters*, 4(9):853–862, 2013. [2](#)
- [13] Daniel D. Lee and H. Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999. [2](#), [14](#), [15](#), [51](#)
- [14] Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901. [2](#), [15](#), [16](#), [51](#), [88](#)
- [15] Harold Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6):417, 1933. [2](#), [15](#), [16](#), [51](#), [88](#)
- [16] Daniel D. Lee and H. Sebastian Seung. Algorithms for non-negative matrix factorization. In *Proceedings of the 13th International Conference on Neural Information Processing Systems*, 2000. [2](#), [15](#), [16](#)
- [17] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In *Advances in neural information processing systems*, pages 153–160, 2007. [2](#), [15](#), [18](#)

- [18] Qing He, Xin Jin, Changying Du, Fuzhen Zhuang, and Zhongzhi Shi. Clustering in extreme learning machine feature space. *Neurocomputing*, 128:88–95, 2014. [2](#), [14](#)
- [19] Gao Huang, Shiji Song, Jatinder ND Gupta, and Cheng Wu. Semi-supervised and unsupervised extreme learning machines. *IEEE transactions on cybernetics*, 44(12):2405–2417, 2014. [14](#)
- [20] Yong Peng, Wei-Long Zheng, and Bao-Liang Lu. An unsupervised discriminative extreme learning machine and its applications to data clustering. *Neurocomputing*, 174:250–264, 2016. [14](#)
- [21] Chenping Hou, Feiping Nie, Dongyun Yi, and Dacheng Tao. Discriminative embedded clustering: A framework for grouping high-dimensional data. *IEEE transactions on neural networks and learning systems*, 26(6):1287–1299, 2014. [14](#)
- [22] Tianchi Liu, Chamara Kasun Liyanaarachchi Lekamalage, Guang-Bin Huang, and Zhiping Lin. Extreme learning machine for joint embedding and clustering. *Neurocomputing*, 277:78–88, 2018. [2](#), [14](#)
- [23] Liyanaarachchi Lekamalage Chamara Kasun, Hongming Zhou, Guang-Bin Huang, and Chi Man Vong. Representational learning with extreme learning machine for big data. *IEEE intelligent systems*, 28(6):31–34, 2013. [2](#), [3](#), [15](#), [18](#), [24](#), [26](#), [28](#), [29](#), [40](#), [64](#)
- [24] Liyanaarachchi Lekamalage Chamara Kasun, Yan Yang, Guang-Bin Huang, and Zhengyou Zhang. Dimension reduction with extreme learning machine. *IEEE Transactions on Image Processing*, 25(8):3906–3918, 2016. [2](#), [15](#), [18](#), [19](#), [29](#), [40](#), [64](#), [71](#), [88](#)
- [25] Guang-Bin Huang, Zuo Bai, Liyanaarachchi Lekamalage Chamara Kasun, and Chi Man Vong. Local receptive fields based extreme learning machine. *IEEE Computational Intelligence Magazine*, 10(2):18–29, 2015. [3](#), [28](#), [75](#), [97](#)

- [26] Zuo Bai and Guang Bin Huang. Generic object recognition with local receptive fields based extreme learning machine. *Procedia Computer Science*, 53(1):391–399, 2015. [2](#), [3](#), [24](#), [28](#), [75](#), [97](#)
- [27] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006. [2](#)
- [28] Geoffrey E Hinton. Deep belief networks. *Scholarpedia*, 4(5):5947, 2009. [3](#)
- [29] Ruslan Salakhutdinov and Geoffrey Hinton. Deep boltzmann machines. In *Artificial intelligence and statistics*, pages 448–455, 2009. [3](#)
- [30] David H Hubel and Torsten N Wiesel. Receptive fields and functional architecture of monkey striate cortex. *The Journal of physiology*, 195(1):215–243, 1968. [3](#)
- [31] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. [3](#)
- [32] Kevin Jarrett, Koray Kavukcuoglu, Yann LeCun, et al. What is the best multi-stage architecture for object recognition? In *2009 IEEE 12th international conference on computer vision*, pages 2146–2153. IEEE, 2009. [30](#), [35](#)
- [33] Yann LeCun, Fu Jie Huang, Leon Bottou, et al. Learning methods for generic object recognition with invariance to pose and lighting. In *CVPR (2)*, pages 97–104. Citeseer, 2004.
- [34] Jiquan Ngiam, Zhenghao Chen, Daniel Chia, Pang W Koh, Quoc V Le, and Andrew Y Ng. Tiled convolutional neural networks. In *Advances in neural information processing systems*, pages 1279–1287, 2010. [3](#), [28](#)
- [35] Tsung-Han Chan, Kui Jia, Shenghua Gao, Jiwen Lu, Zinan Zeng, and Yi Ma. Pcanet: A simple deep learning baseline for image classification? *IEEE*

- transactions on image processing*, 24(12):5017–5032, 2015. [3](#), [29](#), [64](#), [70](#), [75](#), [115](#)
- [36] Dongshun Cui, Guang-Bin Huang, LL Chamara Kasun, Guanghao Zhang, and Wei Han. Elmnet: feature learning using extreme learning machines. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 1857–1861. IEEE, 2017. [20](#), [24](#), [29](#), [33](#), [64](#), [75](#), [88](#), [115](#)
- [37] Wentao Zhu, Jun Miao, Laiyun Qing, and Guang-Bin Huang. Hierarchical extreme learning machine for unsupervised representation learning. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2015. [3](#), [20](#), [24](#), [29](#), [35](#), [64](#), [70](#), [75](#), [88](#), [115](#)
- [38] Emilio Soria-Olivas, Juan Gomez-Sanchis, José D Martin, Joan Vila-Frances, Marcelino Martinez, José R Magdalena, and Antonio J Serrano. Belm: Bayesian extreme learning machine. *IEEE Transactions on Neural Networks*, 22(3):505–509, 2011. [8](#), [12](#), [40](#)
- [39] Jiahua Luo, Chi-Man Vong, and Pak-Kin Wong. Sparse bayesian extreme learning machine for multi-classification. *IEEE Transactions on Neural Networks and Learning Systems*, 25(4):836–843, 2013. [8](#), [13](#), [40](#)
- [40] Guang Bin Huang and Chen Lei. Convex incremental extreme learning machine. *Neurocomputing*, 70(16):3056–3062, 2007. [8](#), [11](#)
- [41] W. F. Schmidt, M. A. Kraaijveld, and R. P. W. Duin. Feedforward neural networks with random weights. In *Pattern Recognition, 1992. Vol.II. Conference B: Pattern Recognition Methodology and Systems, Proceedings., 11th IAPR International Conference on*, 1992. [10](#)
- [42] Halbert White. An additional hidden unit test for neglected nonlinearity in multilayer feedforward networks. In *Neural Networks, 1989. IJCNN., International Joint Conference on*, 1989. [10](#), [11](#)

- [43] Bartlett and P.L. The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. *IEEE Transactions on Information Theory*, 44(2):525–536. [10](#)
- [44] Arthur E Hoerl and Robert W Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970. [11](#)
- [45] Yoh Han Pao, Gwang Hoon Park, and Dejan J. Sobajic. Learning and generalization characteristics of the random vector functional-link net. *Neurocomputing*, 6(2):163–180, 1994. [11](#)
- [46] Chen and C.L.P. A rapid supervised learning neural network for function interpolation and approximation. *IEEE Transactions on Neural Networks*, 7(5):1220–1230.
- [47] C. L. Philip Chen, Steven R. LeClair, and Yoh-Han Pao. An incremental adaptive implementation of functional-link processing for function approximation, time-series prediction, and system identification. *Neurocomputing*, 18(1-3):11–31.
- [48] C. L. P. Chen and J. Z. Wan. A rapid learning and dynamic stepwise updating algorithm for flat neural networks and the application to time-series prediction. *IEEE Transactions on Systems Man and Cybernetics Part B Cybernetics A Publication of the IEEE Systems Man and Cybernetics Society*, 29(1):62–72, 2002.
- [49] B. IgelNIK and Yoh-Han Pao. Stochastic choice of basis functions in adaptive function approximation and the functional-link net. *IEEE Trans Neural Netw*, 6(6):1320–1329. [11](#)
- [50] T Lee, H White, and C W Granger. *Testing For Neglected Nonlinearity in Time Series Models: A Comparison of Neural Network Methods and Alternative Tests*. 1993. [11](#)

- [51] Maxwell B. Stinchcombe and Halbert White. Consistent specification testing with nuisance parameters present only under the alternative. *Econometric Theory*, 1998. [11](#)
- [52] Peter Congdon. *Bayesian statistical modelling*, volume 704. John Wiley & Sons, 2007. [11](#)
- [53] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006. [12](#)
- [54] Tao Chen and Elaine Martin. Bayesian linear regression and variable selection for spectroscopic calibration. *Analytica chimica acta*, 631(1):13–21, 2009. [12](#)
- [55] James O Berger. *Statistical decision theory and Bayesian analysis*. Springer Science & Business Media, 2013. [12](#)
- [56] David JC MacKay. Probable networks and plausible predictionsa review of practical bayesian methods for supervised neural networks. *Network: computation in neural systems*, 6(3):469–505, 1995. [12](#)
- [57] David JC MacKay. Bayesian methods for backpropagation networks. In *Models of neural networks III*, pages 211–254. Springer, 1996. [13](#)
- [58] Anil K Jain, M Narasimha Murty, and Patrick J Flynn. Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323, 1999. [14](#)
- [59] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396, 2003. [14](#)
- [60] Andrew Y Ng, Michael I Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in neural information processing systems*, pages 849–856, 2002. [14](#)
- [61] Barbara Hammer and Thomas Villmann. Generalized relevance learning vector quantization. *Neural Networks*, 15(8-9):1059–1068, 2002. [15](#)

- [62] Andrew Ng et al. Sparse autoencoder. *CS294A Lecture notes*, 72(2011):1–19, 2011. [15](#), [20](#), [51](#)
- [63] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. [15](#), [18](#), [22](#)
- [64] Zhijian Yuan and Erkki Oja. Projective nonnegative matrix factorization for image compression and feature extraction. In *Scandinavian Conference on Image Analysis*, pages 333–342. Springer, 2005. [15](#)
- [65] Zhirong Yang and Erkki Oja. Linear and nonlinear projective nonnegative matrix factorization. *IEEE Transactions on Neural Networks*, 21(5):734–749, 2010. [15](#)
- [66] Rainer Gemulla, Erik Nijkamp, Peter J Haas, and Yannis Sismanis. Large-scale matrix factorization with distributed stochastic gradient descent. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 69–77. ACM, 2011. [16](#)
- [67] Yang Bao, Hui Fang, and Jie Zhang. Topicmf: Simultaneously exploiting ratings and reviews for recommendation. In *Twenty-Eighth AAAI conference on artificial intelligence*, 2014. [16](#)
- [68] Suvrit Sra and Inderjit S Dhillon. Generalized nonnegative matrix approximations with bregman divergences. In *Advances in neural information processing systems*, pages 283–290, 2006. [16](#)
- [69] Ben Murrell, Thomas Weighill, Jan Buys, Robert Ketteringham, Sasha Moola, Gerdus Benade, Lise Du Buisson, Daniel Kaliski, Tristan Hands, and Konrad Scheffler. Non-negative matrix factorization for learning alignment-specific models of protein evolution. *PloS one*, 6(12):e28898, 2011. [16](#)
- [70] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001. [17](#)

- [71] Geoffrey E Hinton and Richard S Zemel. Autoencoders, minimum description length and helmholtz free energy. In *Advances in neural information processing systems*, pages 3–10, 1994. [18](#)
- [72] Hugo Larochelle and Yoshua Bengio. Classification using discriminative restricted boltzmann machines. In *Proceedings of the 25th international conference on Machine learning*, pages 536–543. ACM, 2008.
- [73] Yoshua Bengio et al. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.
- [74] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.
- [75] Minmin Chen, Zhixiang Xu, Kilian Weinberger, and Fei Sha. Marginalized denoising autoencoders for domain adaptation. *arXiv preprint arXiv:1206.4683*, 2012.
- [76] Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, pages 833–840. Omnipress, 2011. [18](#)
- [77] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11(12):3371–3408, 2010. [18](#)
- [78] William B Johnson and Joram Lindenstrauss. Extensions of lipschitz mappings into a hilbert space. *Contemporary mathematics*, 26(189-206):1, 1984. [19](#), [68](#)
- [79] Chris Ding and Xiaofeng He. K-means clustering via principal component analysis. In *International Conference on Machine Learning*, 2004. [19](#)

- [80] Bruno A Olshausen and David J Field. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision research*, 37(23):3311–3325, 1997. [20](#)
- [81] Jiexiong Tang, Chenwei Deng, and Guang-Bin Huang. Extreme learning machine for multilayer perceptron. *IEEE transactions on neural networks and learning systems*, 27(4):809–821, 2015. [20](#), [24](#), [27](#), [88](#)
- [82] Youngwoo Yoo and Se-Young Oh. Fast training of convolutional neural network classifiers through extreme learning machines. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 1702–1708. IEEE, 2016. [40](#)
- [83] Yueqing Wang, Zhige Xie, Kai Xu, Yong Dou, and Yuanwu Lei. An efficient and effective convolutional auto-encoder extreme learning machine network for 3d feature learning. *Neurocomputing*, 174:988–998, 2016. [40](#), [88](#)
- [84] Kai Sun, Jianshe Zhang, Chunxia Zhang, and Junying Hu. Generalized extreme learning machine autoencoder and a new deep neural network. *Neurocomputing*, 230:374–381, 2017. [20](#), [88](#)
- [85] Migel D Tissera and Mark D McDonnell. Deep extreme learning machines: supervised autoencoding architecture for classification. *Neurocomputing*, 174: 42–49, 2016. [24](#), [25](#)
- [86] Kevin Jarrett, Koray Kavukcuoglu, Marc’Aurelio Ranzato, and Yann LeCun. What is the best multi-stage architecture for object recognition? In *2009 IEEE 12th international conference on computer vision*, pages 2146–2153. IEEE, 2009. [28](#)
- [87] Andrew M Saxe, Pang Wei Koh, Zhenghao Chen, Maneesh Bhand, Bipin Suresh, and Andrew Y Ng. On random weights and unsupervised feature learning. In *ICML*, volume 2, page 6, 2011. [29](#), [97](#)
- [88] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth*

- international conference on artificial intelligence and statistics*, pages 215–223, 2011. [29](#), [97](#)
- [89] Jinghong Huang, Zhu Liang Yu, Zhaoquan Cai, Zhenghui Gu, Zhiyin Cai, Wei Gao, Shengfeng Yu, and Qianyun Du. Extreme learning machine with multi-scale local receptive fields for texture classification. *Multidimensional Systems and Signal Processing*, 28(3):995–1011, 2017. [29](#)
- [90] Huaping Liu, Fengxue Li, Xinying Xu, and Fuchun Sun. Multi-modal local receptive field extreme learning machine for object recognition. *Neurocomputing*, 277:4–11, 2018. [29](#)
- [91] Xinying Xu, Jing Fang, Qi Li, Gang Xie, Jun Xie, and Mifeng Ren. Multi-scale local receptive field based online sequential extreme learning machine for material classification. In *International Conference on Cognitive Systems and Signal Processing*, pages 37–53. Springer, 2018. [29](#)
- [92] Cheng-Yaw Low and Andrew Beng-Jin Teoh. Stacking-based deep neural network: Deep analytic network on convolutional spectral histogram features. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 1592–1596. IEEE, 2017. [40](#)
- [93] David JC MacKay. The evidence framework applied to classification networks. *Neural computation*, 4(5):720–736, 1992. [42](#)
- [94] Ian T Nabney. Efficient training of rbf networks for classification. 1999. [43](#)
- [95] Michael E Tipping. Sparse bayesian learning and the relevance vector machine. *Journal of machine learning research*, 1(Jun):211–244, 2001. [44](#), [45](#)
- [96] David JC MacKay. Bayesian interpolation. *Neural computation*, 4(3):415–447, 1992. [45](#)
- [97] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin,

- et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016. [47](#)
- [98] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. 2014. [51](#)
- [99] Michael Lyons, Shigeru Akamatsu, Miyuki Kamachi, and Jiro Gyoba. Coding facial expressions with gabor wavelets. In *Proceedings Third IEEE international conference on automatic face and gesture recognition*, pages 200–205. IEEE, 1998. [52](#), [73](#)
- [100] Ferdinando S Samaria and Andy C Harter. Parameterisation of a stochastic model for human face identification. In *Proceedings of 1994 IEEE Workshop on Applications of Computer Vision*, pages 138–142. IEEE, 1994. [52](#), [73](#)
- [101] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017. [52](#), [73](#), [74](#), [100](#), [101](#), [102](#)
- [102] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and André van Schaik. Emnist: an extension of mnist to handwritten letters. *arXiv preprint arXiv:1702.05373*, 2017. [52](#), [73](#), [74](#), [101](#), [102](#)
- [103] Vahid Kazemi and Josephine Sullivan. One millisecond face alignment with an ensemble of regression trees. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1867–1874, 2014. [53](#), [74](#)
- [104] Qinfeng Shi, Chunhua Shen, Rhys Hill, and Anton Van Den Hengel. Is margin preserved after random projection? In *Proceedings of the 29th International Conference on Machine Learning*, pages 643–650. Omnipress, 2012. [64](#)
- [105] Peter Frankl and Hiroshi Maehara. The johnson-lindenstrauss lemma and the sphericity of some graphs. *Journal of Combinatorial Theory, Series B*, 44(3):355–362, 1988. [68](#)

- [106] Kasper Green Larsen and Jelani Nelson. Optimality of the johnson-lindenstrauss lemma. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 633–638. IEEE, 2017. [69](#)
- [107] Antony Jameson. Solution of the equation $ax+xb=c$ by inversion of an $m*m$ or $n*n$ matrix. *SIAM Journal on Applied Mathematics*, 16(5):1020–1023, 1968. [70](#), [71](#), [72](#)
- [108] Richard Bellman. *Introduction to matrix analysis*, volume 19. Siam, 1997. [71](#)
- [109] Sameer A Nene, Shree K Nayar, Hiroshi Murase, et al. Columbia object image library (coil-20). 1996. [73](#), [74](#), [100](#), [101](#)
- [110] Sameer A Nene, Shree K Nayar, and Hiroshi Murase. object image library (coil-100). 1996. [73](#), [74](#), [100](#), [101](#), [102](#)
- [111] Michael Lyons, Shigeru Akamatsu, Miyuki Kamachi, and Jiro Gyoba. Coding facial expressions with gabor wavelets. In *Proceedings Third IEEE international conference on automatic face and gesture recognition*, pages 200–205. IEEE, 1998. [74](#)
- [112] Ferdinando S Samaria and Andy C Harter. Parameterisation of a stochastic model for human face identification. In *Proceedings of 1994 IEEE Workshop on Applications of Computer Vision*, pages 138–142. IEEE, 1994. [74](#)
- [113] Emmanuel Dufourq and Bruce A Bassett. Eden: Evolutionary deep networks for efficient machine learning. In *2017 Pattern Recognition Association of South Africa and Robotics and Mechatronics (PRASA-RobMech)*, pages 110–115. IEEE, 2017. [75](#), [78](#), [79](#), [115](#), [124](#)
- [114] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. [80](#)

- [115] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1389–1397, 2017. [85](#)
- [116] Yoan Miche, Antti Sorjamaa, Patrick Bas, Olli Simula, Christian Jutten, and Amaury Lendasse. Op-elm: optimally pruned extreme learning machine. *IEEE transactions on neural networks*, 21(1):158–162, 2009. [85](#)
- [117] Yoan Miche, Mark Van Heeswijk, Patrick Bas, Olli Simula, and Amaury Lendasse. Trop-elm: a double-regularized elm using lars and tikhonov regularization. *Neurocomputing*, 74(16):2413–2421, 2011. [86](#)
- [118] Xiong Luo, Yang Xu, Weiping Wang, Manman Yuan, Xiaojuan Ban, Yueqin Zhu, and Wenbing Zhao. Towards enhancing stacked extreme learning machine with sparse autoencoder by correntropy. *Journal of The Franklin Institute*, 355(4):1945–1966, 2018. [88](#)
- [119] John C Gower, Garmt B Dijksterhuis, et al. *Procrustes problems*, volume 30. Oxford University Press on Demand, 2004. [90](#)
- [120] John R Hurley and Raymond B Cattell. The procrustes program: Producing direct rotation to test a hypothesized factor structure. *Behavioral science*, 7(2):258–262, 1962. [90](#)
- [121] Peter H Schönemann. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31(1):1–10, 1966. [90](#)
- [122] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence*, 22, 2000. [91](#)
- [123] Paul Robert and Yves Escoufier. A unifying tool for linear multivariate statistical methods: the rv-coefficient. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 25(3):257–265, 1976. [92](#)
- [124] Hervé Abdi. Rv coefficient and congruence coefficient. *Encyclopedia of measurement and statistics*, 849:853, 2007. [92](#)

-
- [125] Yong Peng, Wanzeng Kong, and Bing Yang. Orthogonal extreme learning machine for image classification. *Neurocomputing*, 266:458–464, 2017. [93](#)
 - [126] Martin Arjovsky, Amar Shah, and Yoshua Bengio. Unitary evolution recurrent neural networks. In *International Conference on Machine Learning*, pages 1120–1128, 2016. [94](#)
 - [127] Mikael Henaff, Arthur Szlam, and Yann LeCun. Recurrent orthogonal networks and long-memory tasks. *arXiv preprint arXiv:1602.06662*, 2016. [94](#)
 - [128] Teuvo Kohonen. *Self-organizing maps*, volume 30. Springer Science & Business Media, 2012. [95](#)