2022

ADVANCED REPRESENTATION LEARNING STRATEGIES FOR BIG DATA ANALYSIS

Wandong Zhang
University of Windsor

ADVANCED REPRESENTATION LEARNING STRATEGIES
FOR BIG DATA ANALYSIS

by

Wandong Zhang

A Dissertation
Submitted to the Faculty of Graduate Studies
through the Department of Electrical and Computer Engineering
in Partial Fulfillment of the Requirements for
the Degree of Doctor of Philosophy
at the University of Windsor

Windsor, Ontario, Canada

2022

Advanced Representation Learning Strategies
for Big Data Analysis

by

Wandong Zhang

APPROVED BY:

R. Laganière, External Examiner
University of Ottawa

B. Boufama
School of Computer Science

R. Razavi-Far
Department of Electrical and Computer Engineering

M. Khalid
Department of Electrical and Computer Engineering

J. Wu, Advisor
Department of Electrical and Computer Engineering

April 21, 2022

Declaration of Co-Authorship / Previous Publication

I. Co-Authorship Declaration

I hereby declare that this dissertation incorporates material that is result of joint research, as follows: This dissertation also incorporates the outcome of a research under the supervision of professor QM Jonathan Wu and co-supervisor Dr. Yimin Yang, and collaboration with Dr. Hui Zhang (Chapter 4), Dr. Ming Li (Chapter 4), Haojin Deng (Chapter 6), Dr. WG Will Zhao (Chapter 6) and Dr. Thangarajah Akilan (Chapter 4, 5 and 9). The research under Prof. QM Jonathan Wu and Dr. Yimin Yang is covered in Chapter 4, 5, 6, 7, 8, and 9 of the dissertation. In all cases, the key ideas, primary contributions, experimental designs, data analysis, interpretation, and writing were performed by the author, and the contribution of the coauthors was primarily through the provision of proof reading and reviewing the research papers regarding the technical content.

I am aware of the University of Windsor Senate Policy on Authorship and I certify that I have properly acknowledged the contribution of other researchers to my dissertation, and have obtained written permission from each of the co-authors to include the above materials in my dissertation.

I certify that, with the above qualification, this dissertation, and the research to which it refers, is the product of my own work.

II. Declaration of Previous Publication

This thesis includes 8 original papers that have been previously published/submitted to journals for publication, as follows:

Thesis chapter	Publication title/full citation	Status
Chapter 4	W. Zhang, Q. M. J. Wu, Y. Yang, T. Akilan, H. Zhang. "A width-growth model with subnetwork nodes and refinement structure for representation learning and image classification." IEEE Transactions on Industrial Informatics 17, no. 3 (2020): 1562-1572.	Published
	W. Zhang, Q. M. J. Wu, Y. Yang, T. Akilan, M. Li. "HKPM: A Hierarchical Key-Area Perception Model for HFSWR Maritime Surveillance." IEEE Transactions on Geoscience and Remote Sensing (2021).	Published
Chapter 5	W. Zhang, Q. M. J. Wu, Y. Yang, T. Akilan. "Multimodel Feature Reinforcement Framework Using Moore-Penrose Inverse for Big Data Analysis." IEEE Transactions on Neural Networks and Learning Systems 32, no. 11 (2021): 5008-5021.	Published
	W. Zhang, Q. M. J. Wu, Y. Yang. "Wi-HSNN: A subnetwork-based encoding structure for dimension reduction and food classification via harnessing multi-CNN model high-level features." Neurocomputing 414 (2020): 57-66.	Published
Chapter 6	W. Zhang, Q. M. J. Wu, H. Deng, W. G. W. Zhao and Y. Yang. "Hierarchical One-Class Model with Subnetwork for Representation Learning and Outlier Detection." IEEE Transactions on Cybernetics, 2022.	In Press
Chapter 7	W. Zhang, Q. M. J. Wu, Y. Yang. "Semi-supervised Manifold Regularization via a Subnetwork-based Representation Learning Model." IEEE Transactions on Cybernetics.	Under Review
Chapter 8	W. Zhang, Q. M. J. Wu, Y. Yang. "Multi-Model MPI-based Recomputation Framework for Large Data Analysis." IEEE Transactions on Neural Network and Learning Systems.	Under Review

Chapter 9	W. Zhang, Q. M. J. Wu, Y. Yang, T. Akilan. “Fast Domain Transfer Learning for Application Towards Efficient Pattern Recognition.” IEEE Transactions on Systems, Man, and Cybernetics: Systems.	Under Review

I certify that I have obtained a written permission from the copyright owners to include the above published materials in my thesis. I certify that the above material describes work completed during my registration as a graduate student at the University of Windsor.

III. General

I declare that, to the best of my knowledge, my thesis does not infringe upon anyone’s copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owners to include such materials in my thesis.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

Abstract

With the fast technological advancement in data storage and machine learning, big data analytics has become a core component of various practical applications ranging from industrial automation to medical diagnosis and from cyber-security to space exploration. Recent studies show that every day, more than 1.8 billion photos/images are posted on social media, and 720 thousand hours of videos are uploaded to YouTube. Thus, to handle this large amount of visual data efficiently, image/video classification, object detection/recognition, and segmentation tasks have gathered a lot of attention since the decade. Consequently, the researchers in this domain has proposed various feature extraction, feature learning, and feature encoding algorithms for improving the generalization performance of the aforesaid tasks. For example, the generalization performance of the image classification models mainly depends on the choice of data representation. These models aim at building comprehensive representation learning (RL) strategies to encode the relationship among the input and output attributes from the raw big data.

Existing RL strategies can be divided into three general categories: statistic approaches (e.g. probabilistic-based analysis, and correlation-based measures), unsupervised learning (e.g., autoencoders), and supervised learning (e.g., deep convolutional neural network (DCNN)). Among these categories, the unsupervised and supervised learning strategies using artificial neural networks (ANNs) have been widely adopted. In this direction, several auxiliary ideas have been proposed over the past decade, to improve the learning capability of the ANNs. For instance, Moore-Penrose (MP) inverse is exploited to refine the parameters (weights and biases) of a trained network. However, the existing MP inverse-based RL methods have an important limitation. The representations learned through the MP inverse-based strategies suffer from loosely-connected feature coding, resulting into a poor representation of the objects having lack of discriminative power. To address this issue, this dissertation proposes a set of eight novel MP inverse-based RL algorithms.

The first part of this dissertation from Chapter 4 to Chapter 7 is dedicated proposing novel width-growth models based on subnet neural network (SNN) for representation learning and image classification. In this part, a novel feature learning algorithm, coined Wi-HSNN is proposed, followed by an improved batch-by-batch learning algorithm, called OS-HSNN. Then, two novel SNNs are introduced to detect extreme outliers for one-class classification (OCC). Finally, a semi-supervised SNN, named SS-HSNN is introduced to extend the strategy from the supervised learning domain to the semi-supervised learning domain.

The second part of this thesis, subsuming Chapter 8 and Chapter 9, focuses on improving the performance of the existing multilayer neural networks through harnessing the MP inverse. Here, a novel weight optimization strategy is proposed to improve the performance of multilayer extreme learning machines (ELMs), where the MP inverse is used to feedback the classification imprecision information from the output layer to the hidden layers. Then, a novel fast retraining framework is proposed to enhance the efficiency of transfer learning of DCNNs.

The effectiveness of the proposed subnet- and retraining-based algorithms have been evaluated on several widely used image classification datasets, such as ImageNet and Places-365. Furthermore, we validated the performance of the proposed strategies in some extended domains, such as ship-target detection, food image classification, camera model identification and misinformation identification. The experimental results illustrate the superiority of the proposed algorithms.

This thesis is dedicated to

my dear parents

Acknowledgements

Foremost, I would like to express my sincere appreciation to my supervisor Dr. Q. M. Jonathan Wu for giving me the opportunity to work under his supervision as well as for his guidance and support in my research. He is kind and patient, and he has provided me with more than enough resources for my research. I feel that I am very lucky to be his student. Also, I would like to convey my sincere thanks to my co-supervisor Dr. Yimin Yang who has raised highly constructive comments throughout my study. He has supported me greatly in my research of analytic learning in which we have published several papers in some highly esteemed journals.

I also would like to thank my committee members Dr. Mohammed Khalid, Dr. Roozbeh Razavi-Far, and Dr. Boubakeur Boufama for guiding me and helping me with their invaluable suggestions and comments. I heartily thank our department graduate secretary Ms. Andria Ballo, who has helped me a lot in many areas. I thank my friend Dr. Thangarajah Akilan and my colleagues in the CVSS laboratory who have supported and spent time in helping me revise my papers.

Finally yet most importantly, I would like to thank my family, my parents who have sacrificed nearly everything in supporting my Ph.D. study. They stand with me as strong pillars of support, and I cannot thank them enough for their sacrifice and patience.

Contents

List of Tables

List of Figures

List of Acronyms

ANN	Artificial neural network
AE	Autoencoder
BLS	Broad learning system
CFAR	Constant false alarm rate
DCNN	Deep convolutional neural network
ELM	Extreme learning machine
HFSWR	High frequency surface wave radar
HMP	Hierarchical matching pursuit
LS	Least-squares
M-ELM	Multilayer extreme learning machine
MCC	Maximum correntropy criterion
MCOC-HSNN	MCC-based hierarchical subnet neural network
MLNN	Multi-layer neural network
MSNN	Multi-layer subnet neural network
MP	Moore-Penrose
MPFR	MP inverse-based fast retraining
MR	Manifold regularization
MSE	Mean square error
OCC	One-class classification

OC-HSNN	One-class hierarchical subnet neural network
OS-HSNN	Online sequential hierarchical subnet neural network
PCA	Principal components analysis
RBM	Restricted Boltzmann Machine
RD	Range-Doppler
RL	Representation learning
RML-MP	Recomputation-based multilayer network using MP inverse
RS-node	Refinement layer subnetwork node
S-node	Entrance layer subnetwork node
SCN	Stochastic configuration network
SGD	Stochastic gradient descent
SLNN	Single layer neural network
SMN	Semantic multinomial
SNN	Subnet neural network
SPF	Spatial pyramidal feature
SRML-MP	Sparse recomputation-based network using MP inverse
SS-HSNN	Semi-supervised hierarchical SNN
SVM	Support vector machine
Wi-HSNN	Width-growth hierarchical SNN

Chapter 1

Introduction

In this chapter, the preliminary concepts of big-data, image classification, RL and MP inverse-based neural networks are discussed. Also, the motivation, objective, contributions and organization of this thesis are elaborated.

1.1 Overview

As the development of science and technology improves, the demands for high-dimensional big-data analysis are becoming increasingly higher. Nowadays, more than billions of images and photos are being generated by the use of smartphones and cameras [5]. According to a study [6], there were very few photos uploaded to the internet in 2008. However, in 2014, more than 1.8 billion images were uploaded to Facebook, Instagram and WhatsApp every day. This number is roughly equivalent to one photo for every person on the planet. In addition, a large volume of data has been generated by the installed surveillance cameras, on the fixed or moving platforms. For example, in 2014, there were more than 200 million surveillance cameras used world-wide. In 2016, it leaped by more than 100 million to an estimated 350 million cameras. According to one study [7], all the new vehicles will be equipped with more than 25 cameras by the end of the year 2022, and the total number of surveillance cameras will reach 45 billion. These cameras can generate a huge amount of images for remote monitoring and facility protection. The generation of an overwhelming number of images has raised technical questions of how to real-time monitor and analyze the information. It is almost impossible for humans to process such a volume of data because it incurs considerable expense in terms of skilled experts. Hence, intelligent algorithms which can automatically process large-scale information in applications are what we urgently need.

Image classification is one of the most basic tasks in machine learning. It is a fundamental computer vision problem, where a specific label is assigned to each testing image. The pivotal difficulty of big-data classification lies in the extraction of the optimal representations, which has been investigated many years [8]. It is widely known that the learnt representations can be characterized by different views of the encoded input data [9, 10]. For example, an image can be described both in color view with RGB components and in a texture view by texture representation. Similarly, a vessel target can be localized by using satellite data and radar signals. In most cases, unimodal feature representation will be biased and inadequate for a certain learning task, while the multi-model feature learning frameworks can overcome the aforesaid shortcomings through learning complementary clues [11, 12, 13, 14]. However, the combined feature vector of multi-model often lies in a high-dimensional space, posing a serious problem in the final pattern recognition as an enormous dimensional feature is hard to explain clearly by a mathematical modeling [15]. At the same time, there can be many redundant components and noise in such high dimensional feature vector. Thus, it is crucial to develop an RL technology to remove redundancy, noise, and reduce the dimension of the feature vector for improving the performance of the model. If the data are fairly low dimensional, evaluating the final statistical model from the training samples would be more efficient. In the past few decades, much efforts had been dedicated into exploiting RL and dimension reduction, such as the statistic approaches [16, 17, 18, 19, 20], supervised learning algorithms [21, 22, 23] and unsupervised learning strategies [24, 25].

1.2 Motivation

In general, most of the supervised and unsupervised learning methods utilize artificial neural networks (ANNs) as the foundation in their learning. Most of the present-day ANNs apply stochastic gradient descent (SGD) or MP inverse to train their frameworks. The MP inverse specifically, which was proposed by Schmidt [26] in 1992, is the earliest attempt of using the least-squares (LS) process in single layer neural networks (SLFNs). Compared to some iterative learning algorithms including SGD that suffer from slow convergence and get trapped in a local minimum, the MP inverse-based strategies have advantages of higher training speed [27] and quicker convergence. Authors in [26] mentioned that the output layer vector can be called Fishier vector if the weights are solved by the standard MP inverse solutions. Later, Huang *et al.* [28] proved that by having the MP inverse, the network trained with randomly generated hidden weights can be considered as a universal approximator, even without updating the hidden layer parameters. Following that, several MP inverse-based RL algorithms, which encode features in a specific real-world application have been investigated, such as ELM [24, 25], stochastic configuration network (SCN) [29], broad learning system (BLS) [30] and SNN [31, 23]. However, the existing MP inverse-based RL approaches have several limitations elaborated as follows.

First, they cannot obtain satisfactory results on high-dimensional datasets with a large number of training samples. Most MP inverse-based algorithms have focused on processing small-scale and medium-scale datasets, such as MNIST [32] and NORB [33] with no more than $100K$ samples, whereas utilizing MP inverse to handle big-datasets, such as ImageNet [34] containing more than 1.2 million patterns and Places365 [35] with a sample size of more than 1.8 million, has rarely been explored.

Second, most of the multilayer MP inverse-based algorithms suffer from being incapable of global representations. The existing multilayer MP inverse-based frameworks utilize a two-step learning mechanism to capture the optimal low-dimensional encoding and do the final classification separately. Such a learning strategy has apparent drawbacks. Each learning model is designed for solving one specific task. Users are hard to determine whether or not the obtained encoding is part of the global-level representations. A more advanced technique is to provide a one-step learning algorithm that merges two steps, aiming to generate a generalized representation that facilitates the final cognition [36, 37].

Third, the traditional multilayer MP inverse-based algorithms focus on a one-batch learning fashion that generates the best parameters by training on the entire data at once. Such a training strategy puts severe demand on the processing environment, especially when handling a large-scale dataset with huge input data volume. As for the SGD-based neural networks [38, 39], they can easily avoid the out-of-memory problem because the SGD can process the input data sequentially. Hence, it urges to have a multilayer out-of-core algorithm that is capable of handling the inputs in a batch-by-batch manner. Through a batch-by-batch strategy, users can handle any datasets without having to consider the main memory of the environment.

Fourth, the low-dimensional representations learned from MP inverse-based methods are weak-kneed to handle noise. In some real-world applications, such as OCC, the input data are usually polluted by non-Gaussian noise and extreme outliers. Existing multilayer MP inverse-based algorithms utilize the mean square error (MSE) to learn the encoding, which is less effective when handling the input data with noise and large outliers. The maximum correntropy criterion (MCC) is a second-order statistical measure, showing high robustness in handling non-Gaussian outliers. Therefore, it is intuitive to solve the non-Gaussian noise by OCC instead of the MSE.

1.3 Objective and Contributions

The major objective of this Ph.D. dissertation is to improve the performance of classification by proposing a number of MP inverse-based multilayer neural networks. To the best of our knowledge, the subnet-based RL algorithms [1, 23] have shown the benefits of discriminative feature learning, and recent work in [23] has already achieved superior performance over other state-of-the-art multilayer MP inverse-based algorithms in multi-class classification. Thus, we plan to implement the enhancements by introducing novel learning architectures and optimization strategies based on SNN and MP inverse. The major contributions of this dissertation are listed as follows.

1. **A novel SNN-based classification structure called Wi-HSNN**.A novel width-growth MP inverse-based RL structure is proposed. Different from the other multi-layer neural networks (MLNNs) that build architectures in a layer-wise manner, this model enriches the latent space representations by the growth of network width. Here, we call it the width-growth manner. Note that this method is not a simple ensemble network. Here, the latent space subnet is guided and optimized according to the error term. Further, a novel end-to-end feature transformation and classification algorithm is presented. Here, the feature space transformation and pattern recognition process are combined, where the potential encoding space at the global-level is searched iteratively. Here, the mathematical proof of Wi-HSNN is given.

2. **A batch-by-batch learning algorithm called OS-HSNN.** A batch-by-batch learning strategy for Wi-HSNN to process large-scale datasets is proposed. The learning mode of the proposed OS-HSNN can be switched from one-batch to batch-by-batch, only with the consideration of the input data size. By having the batch-by-batch strategy, the data volume of the input data batch is dramatically reduced, and the process of OS-HSNN does not have a severe computational requirement. In this dissertation, we mathematically prove the effectiveness of the proposed batch-by-batch strategy.
3. **Two SNN-based one-class classification algorithms**. First, a novel multilayer subnet-based structure called OC-HSNN for OCC is proposed in this study. To increase the representational power of the structure, two subnet structures, namely, the entrance layer subnet (En-Subnet) and exiting layer subnet (Ex-Subnet), are introduced in OC-HSNN to find the generalized feature subspaces. Second, the MCOC-HSNN is further proposed for outlier detection in the presence of non-Gaussian noise, where the MCC is designed in low-dimensional RL to improve the performance of classification.
4. **A semi-supervised subnet structure named SS-HSNN**. A subnet-based algorithm using manifold regularization and graph Laplacian is proposed for semi-supervised classification. The manifold regularization assumes that both the labeled and unlabeled samples should have the same distribution. Hence, the conditional probabilities of two nearby points, i.e., $P(\boldsymbol{y}|\boldsymbol{x}_1)$ and $P(\boldsymbol{y}|\boldsymbol{x}_2)$ should be similar.
5. **Two novel optimization strategies based on multilayer ELM (M-ELM)**. A recomputation-based multilayer network using MP inverse (RML-MP) is developed. The M-ELM can only generate loosely-connected representations in processing large-scale datasets because it does not contain parameters' fine-tuning. Thus, in the proposed RML-MP, a novel recomputation strategy based on MP inverse is proposed to feedback the error and fine-tune hidden layer parameters. Meanwhile, the effective $\ell_{1/2}$ penalty-based learning framework is adopted in the retraining stage of RML-MP, leading to a sparse algorithm SRML-MP. A comprehensive comparison has been conducted to validate the effectiveness of the proposed methods over other approaches on different datasets.

6. **A novel transfer learning strategy for DCNNs**. A novel fast domain transfer learning strategy based on MP inverse technique and random layer learning is proposed for DCNNs. Specifically, the batch-by-batch MP inverse is utilized to fine-tune the FC layers of one DCNN, while the random layer freezing strategy is proposed to decrease fine-tuning workloads and to avoid over-fitting.
7. **Two new datasets**. Two new datasets are gathered in this thesis. First, a one-class classification dataset named CO-Mask for misinformation detection is created with texts collected from the "big three" news agencies (Associated Press, Reuters, and Bloomberg) on wearing masks as a way to curb COVID-19. Second, a new ship-target detection dataset called HFSWR-RDE has been prepared. The HFSWR-RDE dataset is a challenging but comprehensive dataset for semi-supervised object detection.

1.4 Research Findings

During the doctorate research, the proposed algorithms and related researches have been published and submitted in some highly esteemed journals and transactions, which are elaborated as below.

1. Zhang W, Wu QMJ, et al., Hierarchical One-Class Model with Subnetwork for Representation Learning and Outlier Detection, *IEEE Trans. on Cybern.*, 2022.
2. Zhang W, Wu, QMJ, et al., Multimodel Feature Reinforcement Framework Using Moore-Penrose Inverse for big-data Analysis. *IEEE Trans. Neural Netw. Learn. Syst.*, 2021
3. Zhang W, Wu QMJ, et al., HKPM: A Hierarchical Key-Area Perception Model for Maritime Surveillance Using HFSWR, *IEEE Trans. Geosci. Remote Sens.*, 2021
4. Zhang W, Wu QMJ, et al., Detection of Vessel Object Using HFSWR with ANOVA-based Spatio-Frequency Analysis, *IEEE Geosci. Remote Sens. Lett.*, 2021
5. Zhang W, Wu QMJ, et al., A Width-growth Model with Subnetwork Nodes and Refinement Structure for Representation Learning and Image Classification. *IEEE Trans. Ind. Inform.*, 2020

6. Zhang W, Wu QMJ, et al., Wi-HSNN: A subnetwork-based encoding structure for dimension reduction and food classification via harnessing multi-CNN model high-level features. *Neurocomputing*, 57-66, 2020.
7. Zhang W, Wu QMJ, et al., Predicting COVID-19 trends in Canada: a tale of four models. *Cogn. Comput. Syst.*, 112-118, 2020.
8. Zhang W, Wu QMJ, et al., A Novel Ship Target Detection Algorithm Based on Error Self adjustment Extreme Learning Machine and Cascade Classifier. *Cogn. Comput.*, 110-124, 2019.
9. Zhang W, Wu QMJ, et al., Semi-supervised Manifold Regularization via Subnetwork based Representation Learning Model, *IEEE Trans. on Cybern.* (Major Revision)
10. Zhang W, Wu QMJ, et al., Multi-Model LS-based Recomputation Framework for Large Data Analysis, *IEEE Trans. Neural Netw. Learn. Syst.* (Under Review)
11. Zhang W, Wu QMJ, et al., Fast Domain Transfer Learning for Application Towards Efficient Pattern Recognition, *IEEE Trans. Syst. Man Cybern.-Syst.* (Under Review)

1.5 Organization of Thesis

This thesis consists of ten chapters, initially starting with this introductory section. Then, it elaborates the details of each proposed method and algorithm. Finally, it concludes the thesis and provides future directions. This thesis is structured as follows.

Chapter 1 introduces the basic concept of RL, the motivation, aims and objectives.

Chapter 2 discusses the background details of the analytic learning methods with definitions, equations derivations, and loss function.

Chapter 3 surveys the existing RL algorithms in different categories, including statistic, supervised and unsupervised learning algorithms. Also, the state-of-the-art MP inverse-based methods are reviewed. The comparison of these methods is discussed, and the strength and limitations of these algorithms are explained as well.

Chapter 4 proposes a novel MP inverse-based algorithm called Wi-HSNN for RL and image classification. The experimental results show that the Wi-HSNN is superior to existing MP inverse-based algorithms on most of the image classification datasets.

Chapter 5 introduces an MP inverse-based batch-by-batch learning strategy for big-data analysis. It is a recursive LS learning algorithm that is capable of processing any large-scale datasets, even the Places-365 dataset containing more than 1.8 million training samples.

Chapter 6 proposes two one-class classification algorithms with MSE or MCC to capture the robust high-level features. Experimental studies verify the effectiveness of the proposed methods on large-scale datasets.

Chapter 7 intends to present a novel subnet-based neural network for semi-supervised RL and classification. A new semi-supervised learning dataset for ship-target object detection is gathered and compared.

Chapter 8 harnesses the ability of multilayer ELM modules in handling large-scale datasets. The output layer error is pulled back by the proposed retraining strategy, and the latent space representations can be updated with the pulled back data.

Chapter 9 proposes an MP inverse-based retraining scheme for DCNN transfer learning. A novel network optimization algorithm instead of a new network structure is proposed for 2D image classification. Thus, the proposed retraining scheme can be utilized in any DCNNs.

Chapter 10 concludes the research findings and summarizes the future directions.

Chapter 2

Background

2.1 Overview

This chapter introduces the foundation of this dissertation by explaining definitions, mathematical equations, derivations, and the background information on the key points related to our proposed MP inverse-based algorithms.

2.2 Image Classification

In computer vision, image classification is a task to decide if a visual image belongs to one image or not. Classifying an image according to a fixed set of labels is an effortless task for humans. However, such a process can incur high costs in terms of skilled human experts, especially when considering the large volume of images uploaded every day through social media, such as Facebook, Instagram and WhatsApp.

In most cases, solving an image classification task can be divided into four stages: image preprocessing, representation learning (feature extraction), post-processing (feature selection) and the final classifier, as shown in Fig. 2.1. The representation learning stage plays a crucial role in the whole process of image classification, where an effective algorithm is designed to remove the redundancy and noise from the input feature vector to boost the classification performance. Among the existing representation learning algorithms [16, 17, 24, 25], the analytic learning strategy [28, 40] is a vital branch.

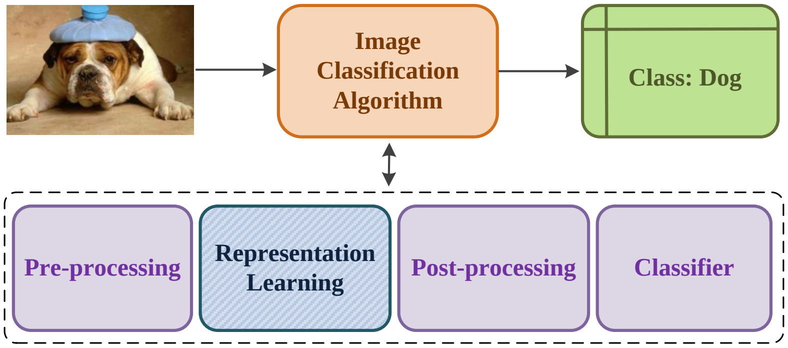


Figure 2.1 – General image classification strategy

2.3 Analytic Learning with Moore-Penrose Inverse

Traditional analytic learning methods convert a nonlinear network optimization problem into several linear segments, where a closed-form least-square solution is utilized to solve the optimal parameters and weights. The commonly used analytic learning algorithms include random vector functional link (RVFL) [41], ELM [28], SCN [29], BLS [30] and SNN [1]. Here, the foundation of ELM and SNN is discussed.

2.3.1 Extreme Learning Machine

Given a training dataset with N training samples $\{\mathbf{X},\ \mathbf{T}\}$, $\mathbf{x}_i \in \mathcal{R}^n$ is the input vector, $\mathbf{t}_i \in \mathcal{R}^d$ is its associated target. The input weights and the bias of latent space neurons are randomly assigned, and the training optimization equation becomes minimizing the error between target $\mathbf{T}$ and output.

$$\min J = ||\mathbf{H}\mathbf{W}_f - \mathbf{T}||_F^2, \tag{2.1}$$

where $|| \cdot ||_F$ is the Frobenius norm, $\mathbf{W}_f$ refers to the output layer weight, $\mathbf{H} = \{\mathbf{h}(\mathbf{x}_i)\} = g(\mathbf{w}_j \cdot \mathbf{x}_i + b_j)$ stands for hidden layer features, $\mathbf{w}_j$ and b_j are the input weights and bias of j-th hidden neuron, respectively, and $g(\cdot)$ is the activation function. The output weight $\mathbf{W}_f$ is calculated via the LS strategy:

$$\mathbf{W}_f = \mathbf{H}^{\dagger}\mathbf{T}, \tag{2.2}$$

where $\mathbf{H}^\dagger$ is the MP inverse of $\mathbf{H}$.

Based on ELM, the researchers proposed ELM autoencoders (ELM-AEs). Similar to the other autoencoder (AE) algorithms, the ELM-AE tries to encode the input data by setting the input as the target output. In particular, the optimal output layer parameters of ELM-AE are calculated with the MP inverse technique. The earliest ELM-AE [42], which is an ℓ_2 norm-based structure, is trained by minimizing the following problem:

$$\begin{aligned} \min\ J &= \frac{1}{2}||\mathbf{H}\mathbf{W}_o - \mathbf{X}||_F^2 + \frac{C}{2}||\mathbf{W}_o||_F^2, \\ \text{s.t. } \mathbf{H} &= g(\mathbf{X}, \mathbf{W}_h, b_h),\ \text{and} \\ \mathbf{W}_h^T\mathbf{W}_h &= I,\ b_h^T b_h = 1, \end{aligned} \tag{2.3}$$

where C is the regularization term, $\mathbf{X}$ refers to the input, and $\{\mathbf{W}_h, b_h\}$ are the hidden layer parameters, which are randomly assigned. The optimal weight $\mathbf{W}_o$ is calculated by the MP inverse strategy, and the encoding $\mathbf{H}_e$ of ELM-AE is described by

$$\mathbf{H}_e = g(\mathbf{X}\mathbf{W}_o^T). \tag{2.4}$$

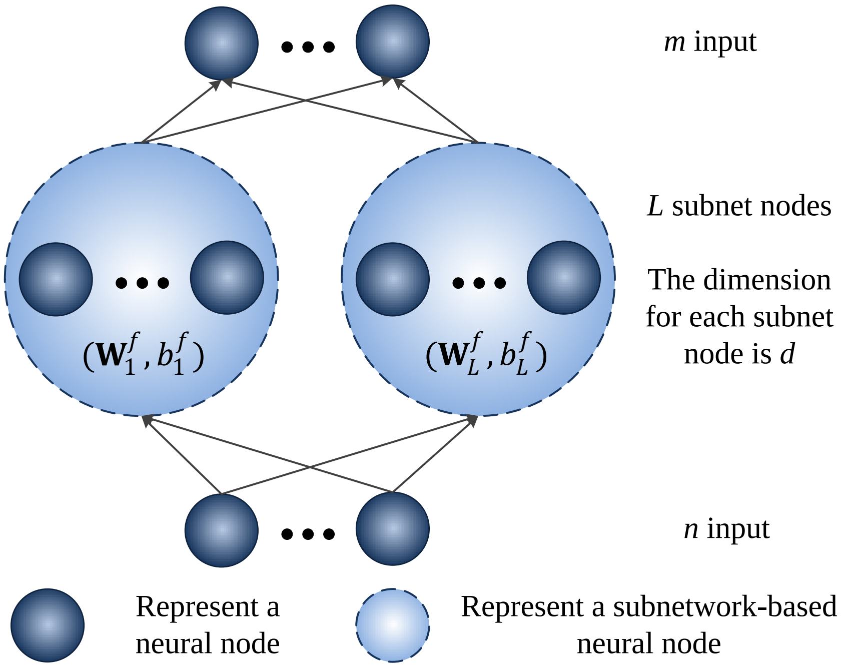
Tang et al. [24] proposed M-ELM algorithm by stacking multiple ELM-AEs. Suppose a M-ELM has K hidden layers, g is the activation function which can be chosen as linear or nonlinear. Mathematically, the objective function J to learn M-ELM can be described as

$$\begin{aligned} \min\ J &= \frac{1}{2}||\mathbf{H}_e^K\mathbf{W}_f - \mathbf{T}||_F^2 + \frac{C}{2}||\mathbf{W}_f||_F^2, \\ \text{s.t. } \mathbf{H}_e^k &= g\left(\mathbf{H}_e^{k-1}(\mathbf{W}_o^k)^T\right),\ 0 \le k \le K, \end{aligned} \tag{2.5}$$

where $\mathbf{H}_e^k$ is the k-th layer output matrix, $\mathbf{X}$ can be considered as the 0-th layer feature matrix where k equal to zero, i.e., $\mathbf{H}_e^0 = \mathbf{X}$, $\mathbf{W}_f$ refers to the weights of the final classification layer, and $\mathbf{W}_o^k$ is the k-th layer weights that will be fixed once determined. The weights between output layer and the k-th hidden layer are generated through MP inverse, which is Eq. (2.2).

2.3.2 Subnet Neural Network

In SNN [1], a hidden neuron node itself can be replaced with a subnetwork formed by several neural nodes, and each hidden layer can be considered as a separate subspace. Hence, the SNN is still a single-layer neural network (SLNN). The structure of the SNN is depicted as Fig. 2.2.



m input
L subnet nodes
The dimension for each subnet node is d
$(\mathbf{W}_1^f, b_1^f)$
$(\mathbf{W}_L^f, b_L^f)$
n input
Represent a neural node
Represent a subnetwork-based neural node

Figure 2.2 – The structure of single layer subnet structure proposed in [1].

The SNN is structured iteratively, i.e., the network is first initialized with one subnet, then, the latent space is gradually enriched by adding new subnets in the structure. Here, we use index i to denote the number of subnetwork. Suppose a dataset has N arbitrary samples, $\{\mathbf{X}, \mathbf{T}\}$, which is sampled from a continues system, the number of subnet in SNN is L, and the dimensionality of each subnet is d. The representations generated from the first subnet, i.e., $\mathbf{H}_i^f$ (i=1) can be expressed as:

$$\mathbf{H}_i^f = g(\mathbf{X} \cdot \mathbf{W}_i^f + b_i^f),\ (\mathbf{W}_i^f)^T \mathbf{W}_i^f = I,\ (b_i^f)^T b_i^f = 1, \tag{2.6}$$

where $\mathbf{W}_i^f \in \mathcal{R}^{n \times d}$, b_i^f are the random weights and bias, respectively.

Given an invertible activation function g and the desired output $\mathbf{T}$, the parameters between the i-th subnet and the output layer is calculated by

$$\begin{aligned}
\mathbf{W}_i^h &= \left(\frac{I}{C} + (\mathbf{H}_i^f)^T \mathbf{H}_i^f \right)^{-1} (\mathbf{H}_i^f)^T \cdot \mathbf{T}, \\
b_i^h &= \sqrt{\mathrm{MSE}(\mathbf{H}_i^f \mathbf{W}_i^h - \mathbf{T})},
\end{aligned} \tag{2.7}$$

where $(\mathbf{H}_i^f)^T \mathbf{H}_i^f$ is the MP inverse of the $\mathbf{H}_i^f$, C is the regularization term.

Then, the current output layer error $\mathbf{E}_i$ and feedback data $\mathbf{P}_i$ are described as

$$\begin{aligned}
\mathbf{E}_i &= \mathbf{T} - (\mathbf{W}_i^h \mathbf{H}_i^f + b_i^h), \\
\mathbf{P}_i &= \left(\frac{I}{C} + (\mathbf{W}_i^h)^T \mathbf{W}_i^h \right)^{-1} (\mathbf{W}_i^h)^T \mathbf{E}_i.
\end{aligned} \tag{2.8}$$

Set $i = i + 1$, add a new subnet in the structure. The parameters and the representations for the newly added subnet is calculated by

$$\begin{aligned}
\mathbf{W}_i^f &= (\frac{I}{C} + \mathbf{X}^T \mathbf{X}) \mathbf{X}^T \cdot T g^{-1}(\mathbf{P}^{i-1}), \\
b_i^f &= \sqrt{\mathrm{MSE}\left(\mathbf{X} \mathbf{W}_j^f - g^{-1}(\mathbf{P}_{i-1}) \right)}, \\
\mathbf{H}_i^f &= g(\mathbf{X} \cdot \mathbf{W}_j^f + b_i^f).
\end{aligned} \tag{2.9}$$

The last step of SNN is to repeat Eq. (2.7) to Eq. (2.9) $L - 1$ times.

2.4 Loss Function

In machine learning and mathematical optimization, the loss function is an equation that measures the distances between the actual network's output and the desired output (label or target). It uses a mathematical equation to describe how the algorithm models the input data. The training of one neural network can be considered as an optimization problem minimizing the loss function. The MSE is the most commonly used loss function for classification and regression. However, the network trained by

MSE is less effective when handling data with non-Gaussian noise and extreme outliers. By contrast, the MCC-based loss function is more effective in handling noise and outliers.

2.4.1 Mean Square Error Criterion

Given N number of training samples $\{\mathbf{X}, \mathbf{T}\}$, let the dimension of each sample is n. Then, $\mathbf{X} \in \mathcal{R}^{N\times n}$ and $\mathbf{T} \in \mathcal{R}^{N\times m}$ are the input data and the expected output, respectively. Suppose $\mathbf{H} = g(\mathbf{X} \cdot \mathbf{W}_h + b)$ is the hidden layer features, where $\mathbf{W}_h$ and b are hidden layer weights and bias, respectively. For a single-layer network, the MSE-based objective function is:

$$\min J = \frac{C}{2}||\mathbf{H}\mathbf{W}_f - \mathbf{T}||_2^2 + \frac{1}{2}||\mathbf{W}_f||_2^2, \tag{2.10}$$

where C is the regularization term, $\mathbf{H}$ is the hidden layer features, $\mathbf{W}_f$ refers to the output weight matrix, and the term $||\mathbf{W}_f||_2^2$ is the ℓ_2 regularization. The $\mathbf{W}_f$ optimized by MSE criterion is described as

$$\mathbf{W}_f = \mathbf{H}^{\dagger}\mathbf{T}, \tag{2.11}$$

where $\mathbf{H}^{\dagger}$ is the MP inverse of $\mathbf{H}$. In fact, there are different ways to calculate $\mathbf{H}^{\dagger}$, such as the orthogonal projection, and the singular value decomposition (SVD). In this paper, with the identity matrix I, the orthogonal projection is utilized because of its efficiency:

$$\begin{aligned} &\mathbf{W}_f = \mathbf{H}^{\dagger}\mathbf{T} = (\frac{I}{C} + \mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{T}, \quad \text{or} \\ &\mathbf{W}_f = \mathbf{H}^{\dagger}\mathbf{T} = \mathbf{H}^T(\frac{I}{C} + \mathbf{H}\mathbf{H}^T)^{-1}\mathbf{T}. \end{aligned} \tag{2.12}$$

2.4.2 Maximum Correntropy Criterion

The correntropy $V(\cdot)$ is the second order statistical measure of two vectors $\mathbf{X}$ and $\mathbf{Y}$ in the Hilbert space [43] defined as in Eq. (2.13).

$$V(X,Y) = E\left[\langle \psi(\mathbf{X}), \psi(\mathbf{Y})\rangle_{\mathcal{H}}\right], \tag{2.13}$$

where E is the expectation, and $\psi(\cdot)$ is the projection of a vector into the Hilbert space. The projection of the vector in the Hilbert space can be efficiently computed by exploiting a kernel function as defined in Eq. (2.14).

$$E\left[\langle \psi(\mathbf{X}), \psi(\mathbf{Y})\rangle_{\mathcal{H}}\right] = E\left[k(\mathbf{X},\mathbf{Y})_{\mathcal{H}}\right]. \tag{2.14}$$

In this dissertation, a Gaussian kernel $G_\sigma(\cdot)$ with variable center is utilized to calculate the expectation in the Hilbert space as defined in Eq. (2.15) [44].

$$k(\mathbf{X},\mathbf{Y})_{\mathcal{H}} = G_\sigma(\mathbf{X}-\mathbf{Y}) = \exp\left(-\frac{||\mathbf{X}-\mathbf{Y}||^2}{2\sigma^2}\right), \tag{2.15}$$

where σ is the bandwidth of Gaussian kernel.

2.5 Convolutional Neural Network

The DCNN is a deep learning algorithm that takes images as the input, using trainable parameters to differentiate one from the other. The pre-processing required in a convolutional model is much less than the traditional classification algorithms. The DCNN is composed of convolutional layers, activation layers and FC layers. The activation layer can be the rectified linear unit (ReLU) layer, sigmoid layer, softmax layer and batch normalization layer. For example, Fig. 2.3 shows a DCNN for handwritten digit recognition in which the feature extraction section and classification section are included. For the feature extraction part, the input image goes through convolutional and activation layers to extract the high-level abstractive features. As for the final classification part, this stage uses the fully-connected (FC) layers and objective functions to classify the latent space representations. Hence, the feature extractors first transform the data from the original RGB space into a latent space. Then, a loss function, such as the cross-entropy loss, evaluates the relationship between the output and the ground truth numerically. The DCNN considers the network learning as an optimization task, where the loss function will be minimized through a back-propagation strategy like SGD or Adam with respect to the parameters of the loss function.

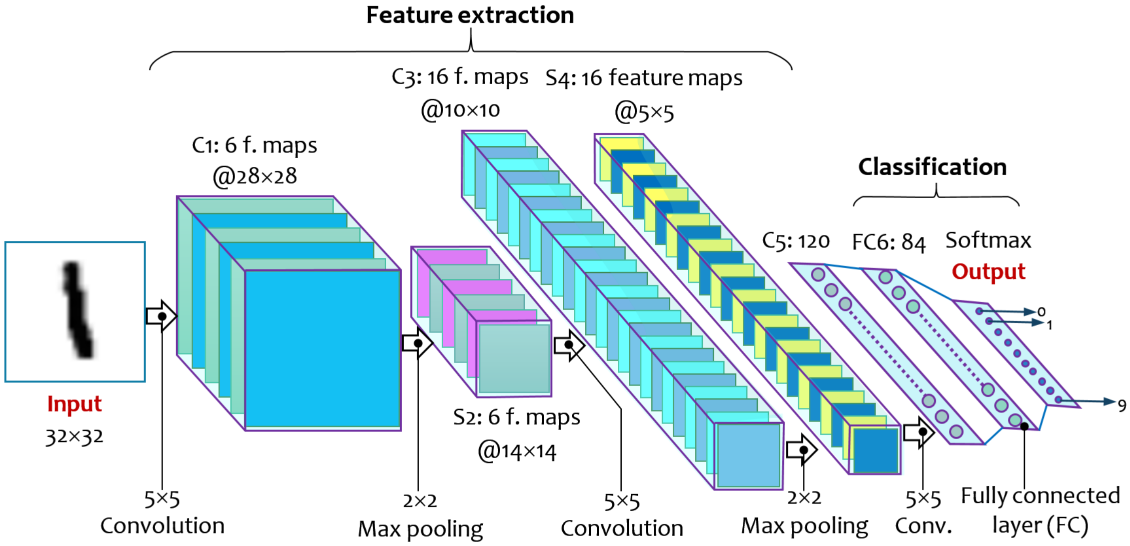


Figure 2.3 – An example of CNN architecture: A handwritten digit classifier [2].

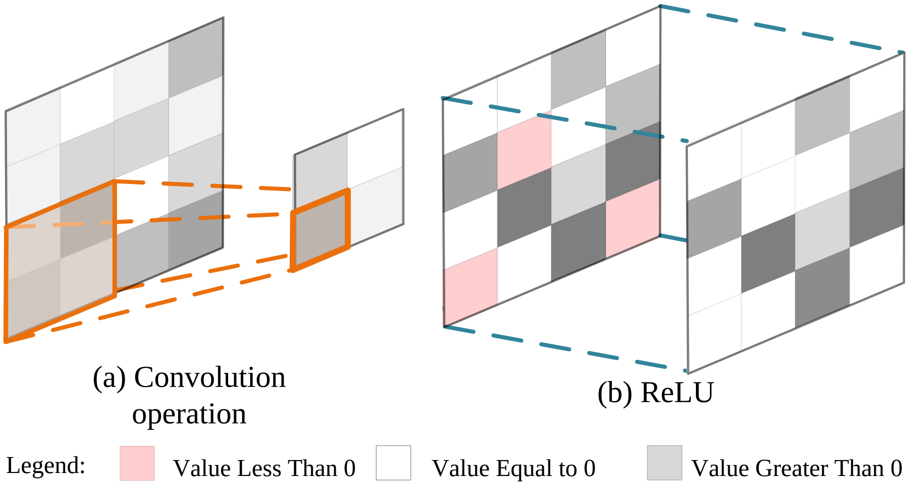
The studies in [45, 46] verified that the depth of one DCNN plays a vital role in pattern recognition, and the DCNN is capable of achieving state-of-the-art performance compared to the traditional methods on various real-world applications. By the use of local receptive fields, shared weights, and spatial sub-sampling, the feature extraction layers of a DCNN have distortion and shift-invariant principles. In the following subsections, we elaborate on the background of the convolutional, fully-connected and activation layers.

2.5.1 Convolutional Layers

The convolutional layers are the core part of the DCNN. Essentially, they apply convolution operation for representation learning and feature extraction, where the weights of each filter operate like a dictionary of feature patterns. The weights of the convolution operation within a feature map are shared. Hence, the position of the local feature is not an important factor in feature extraction, leading to shift-invariant. For the input patch $\mathcal{X}$ and the filter kernel $\mathcal{K}$, the convolution operation can be described as the following

$$C(m, n) = \sum_{p=0}^{P} \sum_{q=0}^{Q} \mathcal{K}(p, q) * \mathcal{X}(m + p, n + q) + b, \tag{2.16}$$

where $*$ is the convolution operation, $\{m, n\}$, $\{P, Q\}$, $\{p, q\}$ refer to the location of 2D patch, the size of convolution kernel, and the index of convolution kernel, respectively. A sample of convolution operation is elaborated as Fig. 2.4a.


(a) Convolution operation

(b) ReLU

Figure 2.4 – Convolution and ReLU operation.

2.5.2 Activation Layers

The purpose of adding activation layers in a DCNN is to bring non-linearity into networks and to learn complex patterns in the data. When comparing with the neurons in our brain, the activation function is similar to the area at the end deciding which neuron is going to be activated. In a DCNN, there are several activation layers in practice, such as ReLU, softmax and batch normalization layers. Among them, the most commonly used activation function is ReLU, which is shown in Fig. 2.4b.

ReLU. The ReLU layer is a piecewise linear function that directly outputs the input when the input is greater than zero, otherwise, it outputs zero. The DCNN rectifies the output of convolution operation through ReLU function defined as $f(x) = max(x, 0)$, where x is the output of convolutional layer.

Softmax. The softmax function is defined as

$$y_i(z) = \frac{e^{z_i}}{\sum_{k=1}^{K} e_k^z}, \tag{2.17}$$

where z is a K-dimensional vector extracted from one layer, $y_i(z)$ is the normalized output with a probability distribution consisting of K probabilities. Each element in $y_i(z)$ ranges from 0 to 1, with the summation of these elements equal to 1. Therefore, if the last layer of a DCNN is the softmax function, then the output of this DCNN of the actual probability scores for each class label. Further, if the cross-entropy is used to build the loss function, the softmax function is modified as

$$L_i = -\ln(\frac{e^{z_i}}{\sum_{k=1}^{K} e_k^z}), \tag{2.18}$$

Batch Normalization. There are several advantages of using batch normalization in a DCNN, such as speeding up the training and reducing the internal covariate shift. Mathematically, batch normalization is defined as Eq. (2.19).

$$\mathbf{X}_n = \frac{\mathbf{X} - \mu_{\mathbf{X}}}{\sqrt{\sigma_{\mathbf{X}}^2 + \epsilon}}, \tag{2.19}$$

where $\mathbf{X}$ is the input of the batch normalization layer, $\mu_{\mathbf{X}}$, $\sigma_{\mathbf{X}}$ and ϵ are the mean, variance of the input batch, and a small positive value, respectively.

2.5.3 Fully-connected Layers

The output from the convolutional layer is the high-level features extracted from the data. The FC layer takes the one-dimensional signal of the output of the last convolutional layer, and it connects every neuron in it to every neuron in the following layer. Essentially, the convolutional layers aim at extracting the invariant features from the input images, while the FC layers learn a prediction/classification function in that space. In place of FC layers, we can also use a conventional classifier, such as ELM or support vector machine (SVM). But the researchers usually use the FC layers at the end to make the DCNN end-to-end trainable. A sample connection by the FC layer is shown in Fig. 2.5.

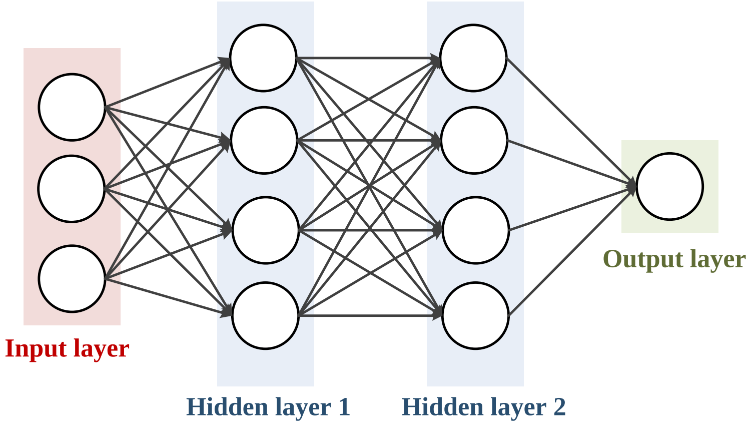


Figure 2.5 – A regular three-layer neural network.

2.6 Transfer Learning

Nonetheless, while DCNNs have achieved the state-of-the-art in various machine learning tasks, such as image classification [14], nature language processing [47], image captioning [48], and object detection [49], they still face several pitfalls. First, they need a large number of labelled data to train the model. For example, AlexNet [50] contains more than 60 million trainable weights, and the researchers need to use a large-scale dataset like ImageNet with more than 1.4 million training samples to optimize the parameters. Also, there is a stringent demand for the initialization of weights for efficient training. Severyn *et al.* [51] has pointed out that the performance of the DCNN depends on the initialization of the parameters. Second, the training of the DCNN from the scratch is time-consuming. If the DCNN has hundreds of convolutional layers, then the training of this DCNN can take more than months if the computer does not have a good graphics processing unit (GPU).

Transfer learning technique overcomes the afore-said limitations. Transfer learning unlocks two major benefits: 1) it reduces the network's training time, and 2) it reduces the amount of data required. These benefits encourage the researchers to investigate the transfer learning strategy in many applications, such as scene image classification and video action recognition. Generally, a base neural network, such as AlexNet, is first trained on a source domain. Then, the weights of this model are utilized as the initial weights to a new network which will be fine-tuned and retrained on the target domain.

One core application of transfer learning is that of multi-modal fusion. In this area, transfer learning exploits the commonalities among different learning tasks, aiming to share statistical strength and to transfer the knowledge across tasks. The RL algorithms have an advantage for this task as they learn the latent space encoding that captures underlying clues, as shown in Fig. 2.6.

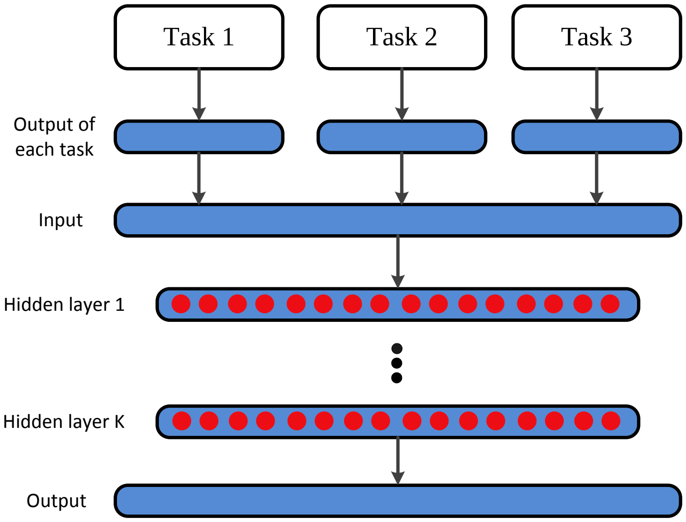


Figure 2.6 – Illustration of RL algorithms discovering underling factors.

Chapter 3

Literature Review

3.1 Overview

In this chapter, a detailed literature review of this dissertation is given. The review mainly includes two sections: the related works on RL and the works regarding the MP inverse-based algorithms. The existing RL methods elaborated in Section 3.2 can be categorized into three categories, namely, statistic approaches, supervised learning methods and unsupervised learning strategies. The review of MP inverse-based algorithms mainly consists of single-layer and multi-layer analytic learning structures.

3.2 Representation Learning Methods

The topic of representation learning has been investigated for many years [52]. The following subsections lay the background and elaborate on the existing RL methods.

3.2.1 Statistic Approaches

The current statistic approaches contains two types of algorithms, the probabilistic methods and the correlation-based frameworks.

3.2.1.1 Probabilistic Methods

From the probabilistic modelling perspective, the RL tasks are based on the statistical model $p(x,\, h)$ describing the joint space of the latent space h and the observed data x. In particular, $p(x,\, h)$ can be considered as the result of the inference process to determine the statistical model $p(h|x)$, which is the posterior probability explaining the probability distribution of the latent space variable h given the raw data x. The probabilistic methods give us two different manners in which we can consider the question of latent space learning, directed or undirected graphical algorithms. In the following, we review the existing methods in these two directions.

Directed Graphical Methods These methods apply the conditional likelihood $p(h|x)$ and the prior probability $p(x)$ to compute the joint distribution probability.

$$p(x,\, h) = p(h|x)p(x). \tag{3.1}$$

The directed graphical methods include principal components analysis (PCA) [53], sigmoid belief networks [54], and spike-and-slab sparse coding model [55]. The PCA is a technique that achieves dimension reduction and minimizes the information loss [56]. The earliest work regarding PCA can be traced back to 1901 proposed by Pearson *et al.* [57], but it was not until the computers became widely used that it was therefore computationally feasible to be processed. Going beyond the basic PCA, there are many adapt and advanced PCAs that achieve modified goals to analyze data, such as functional PCA [58] and robust PCA [59]. As for the spike-and-slab sparse coding model [55], it defines two different variables: the *spike* and *slab* variables. The spike variables are latent binary values, while the slab variables are a set of latent real values. The performance of spike-and-slab sparse coding has been verified in many datasets, such as CIFAR-10 and CIFAR-100. Experiments validate that this method outperforms most of the sparse coding representations.

Undirected Graphical Methods Undirected graphical methods, also called Markov random fields (MRFs), construct the joint distribution probability $p(x,\, h)$ through a product of unnormalized non-negative clique potentials [60]:

$$p(x, y) = \frac{1}{Z_\theta} \prod_i \psi_i(x) \prod_j \eta_j(h) \prod_k v_k(x, h), \tag{3.2}$$

where $\psi_i(x)$, $\eta_j(h)$ and $v_k(x, h)$ refer to the clique potentials explaining the interactions between the visible elements, between the hidden variables, and between the visible and hidden variables, respectively, and the function $1/Z_\theta$ normalizes the distribution. The most commonly used undirected graphical method is the restricted

Boltzmann machine (RBM). The RBM [61] essentially is a random network with a two-layer symmetric connection. In their basic form, the RBM assumes the input of their structure is the Bernoulli variables with two possible states, 0 or 1. It models the most prominent interactions that occur in the processing dataset. Depending on the objective of the task, the RBM can be utilized in either the supervised or unsupervised domain. Going beyond the original RBM, important progress has been made in exploring the advanced and improved RBMs for better representation purposes. Ranzato *et al.* [62] proposed the Gaussian RBM that adds a bias term which is quadratic in the visible units x to the visible units biases. However, the Gaussian RBM is sometimes incapable of modelling natural images. Recently, some deep structures of RBM, such as deep belief network [63] and deep Boltzmann machine [64] have been formed. Because of its excellent feature representation abilities, the original and improved RBMs are widely used in feature extraction [65] and image processing [66].

3.2.1.2 Correlation-based Methods

Another important perspective of statistic learning approach is that of the correlation-based methods. Its premise is the manifold hypothesis based on manifold learning, which assumes that the observed data lie on a low-dimensional manifold embedded in a higher-dimensional space. The current correlation-based algorithms can be categorized into two families, the non-parametric and parametric approaches.

Non-parametric Correlation-based Algorithms In the non-parametric algorithms, each high dimensional data point has its own set of free low dimensional embedding coordinates. The data points optimized by non-parametric correlation-based methods can preserve certain properties of the neighbourhood graph from the original high dimensional space. The neighbourhood graph is also called the geometric perspective. Most of the geometric perspective-based methods are implemented based on a training set nearest neighbour graph [67, 68]. However, these methods cannot learn a mathematical expression or feature learning function from the training dataset, which is inefficient when handling the newly arrived testing patterns. Hence, several non-linear manifold learning algorithms have been proposed, which are discussed in the following subsection.

Parametric Correlation-based Algorithms One main advantage of the parametric correlation-based methods is that, when the new testing data arrives, these methods can use a parametric map f_θ to directly construct encoding for new points. Some of the non-parametric algorithms can be modified as the parametric algorithms, such as the t-SNE [69]. Recently, parametric correlation-based algorithms have been employed in the area of semi-supervised learning. For example, Weston *et al.* [70] utilizes the manifold hypothesis to learn the representation of the input data by a neighbourhood graph, wherein a hierarchical neural network that can simultaneously learn the latent embedding and the classifier is developed. The training criterion of [70] encourages the neighbours of training samples to maintain a similar representation. Compared to the non-parametric correlation-based algorithms, the parametric methods forces the model to generate the manifold shape non-locally [71], which generates better representations and final performance.

3.2.2 Supervised Learning Methods

The supervised learning algorithms have been investigated for decades. Two vital branches of supervised learning methods are supervised coding methods and DCNNs.

3.2.2.1 Supervised Coding Methods

The supervised coding methods evaluate the importance of a specific feature through the correlation between features and categories. The supervised coding methods can be split into three families: sparse representation methods [72, 73], semantic multinomial methods [74, 3] and the intermediate representation methods [75].

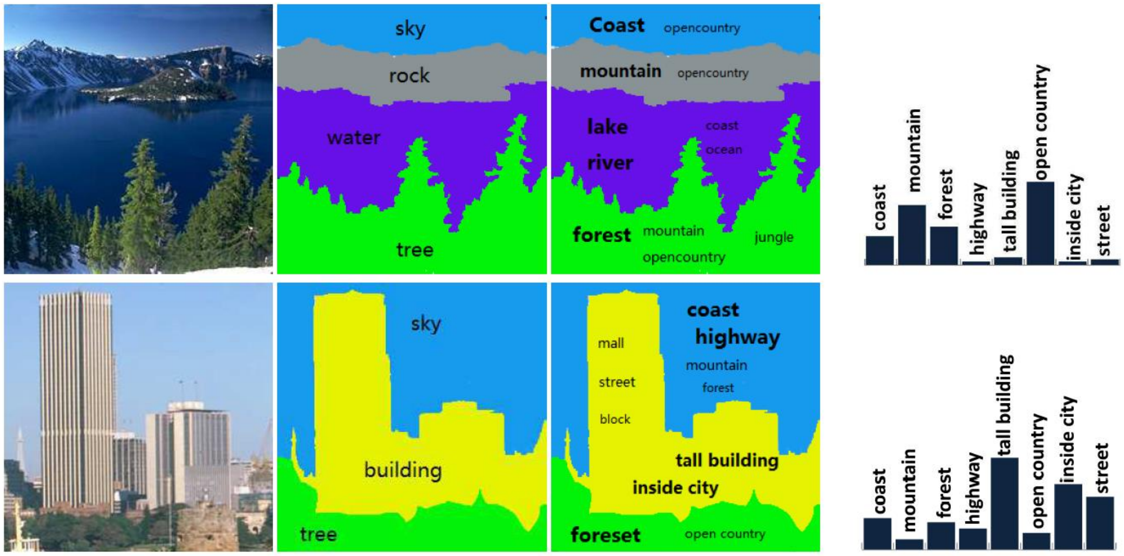
Sparse Representation The sparse representation methods learn a discriminative codebook for feature representation with separate learning stages or in an end-to-end fashion. Jiang *et al.* [22] introduced a supervised coding method, label consistent K-means-based singular value decomposition (K-SVD), for sparse representation learning. This algorithm learns the discriminative dictionary, coding parameters, and classifier parameters simultaneously, whereby the optimal solution is derived. On the other hand, the neighbourhood preserving embedding (NPE) proposed by He *et* *al.* [76], is a linear dimensional reduction algorithm that preserves the local neighbourhood structure on the data manifold. In this way, the NPE is more robust in handling outlier data points compared with its counterparts. In [77], Cai *et al.* provide a novel data-analytic method named locality sensitive discriminant analysis (LSDA). In contrast with the basic linear discriminant analysis (LDA), the LSDA-based data processing tool works on the local manifold structure so that it can have complete preservation of both discriminant and local geometrical structures in the data. Bo [72] introduced a descriptor called multipath hierarchical Matching Pursuit (M-HMP), that combines a collection of hierarchical sparse features for image classification so that multiple discriminative feature vectors can be obtained. The sparse representation-based methods have achieved good performance in many scenarios [78].

An important branch of sparse representation is the bag-of-word (BoW) method. Originating in the previous year's literature, BoW algorithms have become popular for image processing and pattern recognition because of their effectiveness and flexibility. BoW methods have four steps: feature extraction, codebook generation, feature coding, and visual histogram representation. It is already shown that given the codebook, how to generate the feature coding to find the proper representation has a vital impact on final classification and recognition. The simplest way is to generate a specific feature to the closest code, and assign one or zero as its coefficients, which is termed as "hard-assignment" [79]. This assignment strategy does not consider the feature ambiguity and may introduce large quantization errors. In [80, 81], the improved coding strategies, *soft-assignment* and *visual word ambiguity* (VWA) were proposed to alleviate the drawback by assigning a local feature to all codewords. Precisely, the generated coding coefficient shows the membership of a certain feature to the generated codewords, and the strong relationship with the codeword always generates a high coefficient. In recent years, local coding schemes have been widely investigated. They optimize a linear combination of a few codewords to approximate a local feature and code it with the optimized coefficients. For example, in [82], Yang *et al.* proposed a local coding algorithm named sparse coding SPM (ScSPM). This method uses sparse coding instead of traditional vector quantization to obtain feature coefficients, and the local maximum pooling is used to generate the optimal representation. In [83], Wang *et al.* introduce a simple coding algorithm called locality constrained linear coding (LLC). It uses local constraints to project the feature descriptors to multiple inner subspaces, and these local subspaces are integrated by a merging operator to further obtain the final representation.

Semantic Multinomial Representation The semantic multinomial (SMN) method aims to extract the local-level representation. It shows the probability distribution of a given image patch belonging to each scene category. Figure 3.1 shows two examples of the categorization results of Image-SMN. It is observed that both regions and images share the same semantic space in the scene level [3]. In most cases, researchers use weak-supervision algorithms trained with scene category labels to learn patch models. However, image patches representing the same local-level concept may also exist in another scene category, which leads to the category co-occurrences [84]. For example, in Fig. 3.1b and c, the local-level concept "tree" is common in the categories "tall building" and "open country". Kwitt *et al.* [74] utilizes a second classifier to separate category co-occurrences from purely accidental co-occurrences, i.e., noise. It shows that category co-occurrences are consistent across the images in the same category. This second classifier is capable of correcting the ambiguity resulting from the weak supervision. However, the work [74] only considers category co-occurrences using only image-level information and supervised modelling. Hence, Song *et al.* [3] proposed an advanced algorithm that can solve the category co-occurrences in both patch and image levels, which shows superior performance than Kwitt's work.

3.2.2.2 Deep Convolutional Neural Networks

In the past few years, the deep learning structures have arguably become the state-of-the-art methods for many computer vision tasks, such as video action recognition and image classification. In a deep learning network, the input data goes through multiple hidden layers to extract high-level and abstractive representations. During training, each hidden layer of a deep structure tries to transform its inputs from the original space into a different encoding, as shown in Fig. 3.2. Furthermore, this encoding itself can be considered as an input to the subsequent layer, which leans to transform it into another representation. Eventually, the final layer of this deep network learns to represent the last encoding into the final prediction.



(a) Raw Image (b) Mid-level concept (c) Scene categories in image (d) Scene categories

Mid-level concepts = {sky, rock, water, building, tree, road, car}
Categories = {coast, mountain, forest, highway, tall building, open country, inside city, street}

Figure 3.1 – Scene category co-occurrences [3].

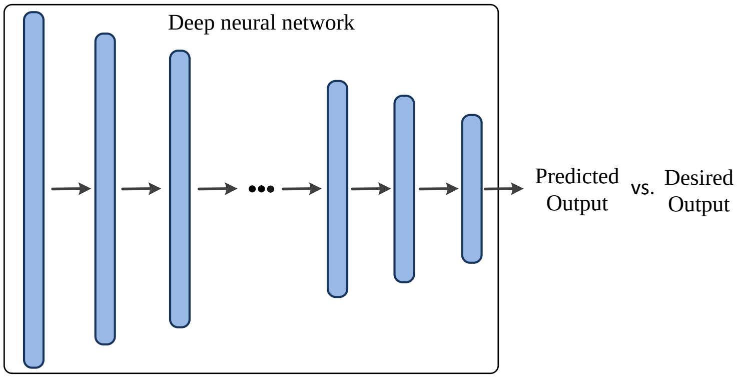


Figure 3.2 – Feature representation of a deep neural network. Each hidden layer learns to transform its input data into a different representation.

The DCNN is a vital branch of deep learning where the results of a certain convolutional layer are the coding vector of the previous layer. It has been around for a long time since the early works of LeCun in the 1980s [85, 45]. Therefore, DCNNs [86, 87, 88] can be seen as an extension of the supervised feature representation and coding techniques. The works in [89, 90, 91] have already verified that the performance derived from a classifier with DCNN coding is superior to most of the conventional feature coding approaches.

3.2.3 Unsupervised Learning Methods

One of the most famous and powerful unsupervised learning strategies is the AE. The AE structure [92] learns the representations of the raw input by reproducing the input data at the output side. One AE is composed of two parts: the encoder and decoder part. The encoder part exploit the latent space representation h by a feature-extracting function f_θ. The feature vector $h = f_\theta(x)$ allows the straightforward computation and efficient encoding. Another part of one AE is the decoder. A function g_θ maps the data from the latent space back to the input space, generating a reconstruction r by $r = g_\theta(h)$. Different from the probabilistic methods that use the maximum likelihood principle to optimize the learning models, the AEs are optimized through the encoder and decoder simultaneously on the task of reconstructing through the iterative learning strategy like SGD or the non-iterative learning method, such as MP inverse. During training, researchers attempt to minimize the reconstruction error and the loss function $L(x, r)$, i.e., a measure of the discrepancy between x and r. Further, to boost the performance of AEs in complex and high dimensional data learning, explosive developments on regularized AEs have been witnessed, such as sparse AEs, denoising AEs and contractive AEs.

Sparse AE The idea of sparse AE is to introduce a form of sparsity regularization in optimizing encoder and decoder weights [93]. The sparse representation of the input data can be achieved in two manners: 1) penalizing the hidden layer bias by making them more negative [93, 94, 95], and 2) penalizing the hidden layer activation by making the output of the hidden layer closer to the saturating value at 0 [96, 97]. The ℓ_1 normalization might be the most intuitive idea for sparse learning as it is widely used in sparse coding [98]. In addition, the close cousin of the ℓ_1 penalty, namely, Student-t penalty [99], has also been widely used in constructing AEs. These sparse learning penalties show excellent performance in data representation and image classification.

Denoising AE Vincent *et al.* [100] proposed the denoising AE by reconstructing the clean data at the output side from the polluted input. In other words, the objective function of the denoising AE is to reconstruct the clean input from a corrupted version. By doing so, simply learning the identity is not enough, the encoder and decoder need to capture the structure of the corrupted input to learn the discriminative knowledge and reveal the effect of the corruption process. The noise considered in the training includes Gaussian noise, salt and pepper noise, etc. Experimental results verified that the discriminative feature can be learned by the denoising AE [101]. Following that, several variant denoising strategies have been proposed based on the score matching parameter estimation technique [102, 103, 104].

Contractive AE Rifai *et al.* [105] proposed the contractive AE to learn the robust representation by adding an analytic contractive penalty to the objective function, i.e., the Frobenius norm of the encoder's Jacobian. Compared to the denoising AE, the advantage of constructive AE is threefold: First, the sensitivity of the latent space representation is penalized. Second, the penalty is analytic and controlled instead of stochastic. Third, there is one hyper-parameter that provides the trade-off between reconstruction and robustness. However, the objective function of denoising AE merges these two parts.

3.3 Moore-Penrose Inverse-based Algorithms

The review of MP inverse-based algorithms starts at introducing the single-layer MP inverse-based neural networks, followed by the multi-layer and hierarchical neural networks.

3.3.1 Single-layer Moore-Penrose Inverse-based Methods

The MP inverse strategy is an analytic learning method. Compared to the BP learning scheme that may become trapped in a local minimum and is sensitive to the learning rate setting, the MP inverse-based strategies have the advantage of quicker convergence. The earliest attempt of the MP inverse technique in a neural network can be traced back to 1992 [26]. Recently, the random thoughts penetrate neural networks, several MP inverse-based algorithms, such as radial basis function (RBF) network [106], RVFL network [107], ELM [42] and SCN [29] have been proposed. The RBF network optimizes its second layer's weights based on a closed-form LS solution while having a simple feature transformation in the first layer. Research in [108] reveals that the learning of the first layer weights is relatively unimportant compared to the learning of the second layer weights, where a random initialization of the weights is enough for competitive performance. Similarly, the RVFL technique has the advantage of fast training speed and excellent generalization performance, which uses random hidden layer features (generated within a suitable range and kept fixed) to construct the architecture.

Huang *et al.* [28] in 2006 provided a single layer network called ELM for image classification and pattern recognition. In ELM, only the output layer parameters are required to be optimized, while no training is needed for the hidden layer parameters. Based on that, the MP inverse-based AE [4] using MP inverse technique for unsupervised feature encoding has been proposed. In recent years, researchers in various real-world applications have made significant contributions to broaden the field scope of MP inverse-based learning strategies [109, 110], and explosive developments on multilayer networks using MP inverse technique have been seen [111, 42].

3.3.2 Multi-layer Moore-Penrose Inverse-based Methods

Recently, the progress of MP inverse-based analytic learning methods has shown an increasing trend towards the incorporation of the fast-learning attribute to deep learning networks. Compared to shallow networks, deep models have better capabilities in processing complex and high-dimensional data. In this direction, multiple ELM-based multilayer networks were proposed. In 2013, the multilayer ELM [42], which stacked multiple l_2 penalty-based ELM autoencoders were proposed. Then, authors in [98] developed a hierarchical framework structured with l_1 penalty-based ELM-AEs to explore the sparse representations. The representative achievements of the multilayer ELM include deep weighted ELM [112], kernel-based ELM [113] and multilayer one-class ELM [25].

In 2016, a novel MP inverse-based architecture called SNN was developed for supervised learning [114, 115]. The single-layer SNN showed superior performance in image classification than the traditional MP inverse-based classifiers. In SNN, each perception in the hidden layer is replaced with a subnetwork composed of a group of ordinary neurons. The weights of each subnetwork are calculated using the pulled-back error matrix from the output layer. Recently, the deep subnet-based structure has been comprehensively investigated. For example, Yang *et al.* [114] developed a two-layer SNN model for dimension reduction and representation learning. The first layer uses the subnetwork structure to encode the features, while the second hidden layer is the feature learning layer used for feature projection. Recently, a hierarchical feature encoding and representation learning framework [23] has been proposed. In [23], hundreds of SNN [1] are embedded in the first layer, generating hundreds of heterogeneous features. Then, the local features generated from the SNN are extracted and recast into a more generalized feature space. Finally, the final category is obtained by a single-layer subnet-based classifier. This hierarchical structure has been successfully applied to different research areas, including EEG signal processing [31] and vigilance estimation [116]. However, state-of-the-art subnet-based strategies [117, 23, 31, 116] face one main drawback: They utilize several sub-modules to learn the encoding of the input and to learn the classifier separately, resulting in sub-optimal results.

Chapter 4

A Width-growth Model with Subnetwork Nodes and Refinement Structure for Representation Learning and Image Classification

This chapter proposes a new subnetwork-based feature refinement and classification model for supervised RL. The novelties of this algorithm are as follows: 1) Different from most multi-layer networks that go deeper with an increased number of network layers, this work architects a model with wider subnetwork nodes. 2) The conventional classification methods adopt a separate search mechanism to derive a generalized feature space and to get the final cognition, but this work proposes a one-step process to find the meaningful latent space and recognize the objects. 3) The traditional feature representation and image classification approaches apply unimodal feature coding, which suffers from being incapable of global knowledge. This work overcomes the pitfall through multimodal fusion that fuses various feature sources into one super-state encoding to achieve higher performance. A cross-domain experimental study on image classification, camera identification, and food image identification shows that the proposed method achieves superior performance compared to the existing models.

4.1 Introduction

Recently, there has been an increasing demand for high-dimensional data processing and learning for the applications of surveillance and unmanned ground vehicles [118, 119, 120]. It has gathered a lot of attention from researchers, who proposed various techniques of feature extraction, feature learning, and feature representation for better classification and recognition performance. The pivotal difficulty of big data processing lies in selecting the optimal feature descriptors, which has been investigated for many years.

In general, each representation extracted from a feature descriptor tends to learn some specific characteristics of input data, and these features often have the pitfall of irrelevant, noisy, or uninformative clues for modelling an accurate learning system [14, 10]. Thus, there is an intuition to obtain a resourceful RL algorithm that encodes and refines these raw features from multiple sources to improve the classification performance. Once all the features are extracted, they are normalized and concatenated into one super-vector. Then, this super-vector can be transformed into a more generalized latent space through a feature refinement strategy. Figure 4.1 depicts an abstract data flow of the proposed learning model with one possible fusion and four independent feature sources. There are four feature descriptors, namely, AlexNet, ResNet, hierarchical matching pursuit (HMP) and spatial pyramidal feature (SPF) that generate output features for each input image with dimensions of 2048, 2048, 3060 and 3000, respectively. The analytic learning strategy using MP inverse strategy has been widely used in neural networks, including RVFL [121] and ELM [122]. Recently, these regression-based feature learning has been exhaustively investigated [123, 42]. Similar to the other AE-based networks, the multilayer analytic learning methods construct the network with several individual AEs.

Although the multilayer analytic learning approaches resulted in improving classification performance, several drawbacks remain. Firstly, almost all these algorithms use “block” models that communicate with each other to establish the network, rather than in a one-step training principle. The one-step learning strategy is to train the network from the raw data to the final target directly, reducing the effort of human design. Researchers have already verified that the generalization performance of a model trained from several designed processes cannot be expected to be perfectly aligned with one-step learning. For example, in optical character recognition (OCR), it is shown to be a better way to apply DCNNs directly to regress the words by themselves [124, 125], instead of recognizing the characters and cluttering them into words separately.

In addition, most multilayer analytic learning methods go deeper in the sense

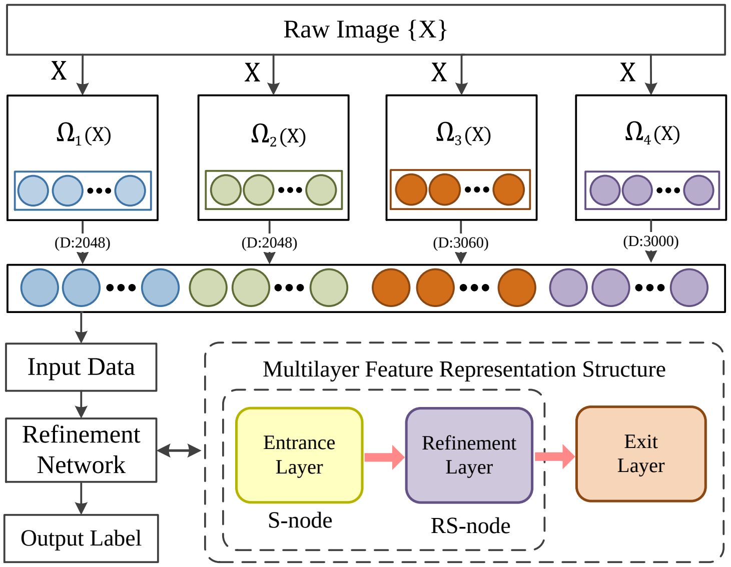


Figure 4.1 – Abstract dataflow of the proposed framework with one possible multi-modal fusion of four separate feature sources (Ω): AlexNet (Ω_1), ResNet (Ω_2), HMP (Ω_3), SPF (Ω_4).

of increasing network depth (i.e., increasing the number of layers), which is a layer stacking approach. This is an easy and safe way of obtaining a high-quality hierarchical network with plenty of training data samples. However, this simple solution may have poor performance with larger network size, especially if the number of labelled samples in the training set is limited [87]. This is the main drawback of such networks, since they are more sensitive to over-fitting, often reducing the reliability of a trained network. SNN [23] is a practical way of enhancing the network's generalization capacity for feature learning. In [23], a single hidden layer could contain multiple subnets and sub-spaces. Inspired by SNN, the work proposed in this chapter addresses the aforesaid shortcomings by introducing an RL based on a "width-growth" architecture with subnetwork node (S-node) and refinement subnetwork node (RS-node). The contributions are as follows:

1) A novel MP inverse-based framework called Wi-HSNN is proposed. To increase the representational power of the network, two subnetwork structures, S-node, and refinement S-node are utilized to find the generalized feature subspaces.

2) A novel one-step learning strategy is proposed. Most of the MP inverse-based

RL methods generate loosely-connected latent space features. To generate the global-level representations, a one-step learning strategy based on SNN is proposed.

3) A highly discriminative representation of the samples is achieved through a multimodal fusion strategy. This work concatenates various feature channels in the input layer, and the super-state encoding can be generated. Further, the definition and mathematical proof of global-level representation are present in this chapter.

The rest of the chapter has the following organization: Section 4.2 presents the proposed width-growth architecture. The extensive experimental results are analyzed in Section 4.3. Then, the concludes concludes its findings and draws directions for future work in Section 4.4.

4.2 The Proposed Algorithm

The notations used in this chapter is shown in Table 4.1.

Mathematically, the SLNN (n input to m output) with a cost function J of a classification problem can be computed as:

$$\text{minimize} \;\; J = \frac{1}{2}\sum_{j=1}^{N} ||\mathbf{T} - g(\mathbf{X}, \mathbf{W}, b)\boldsymbol{\beta}||^2, \tag{4.1}$$

where N is the number of data points, $\mathbf{X} \in \mathcal{R}^{N\times n}$ represents the input of a data, $\mathbf{T} \in \mathcal{R}^{N\times m}$ is the expected output, $(\mathbf{W}, b)$ is the hidden layer parameters, and $\boldsymbol{\beta}$ is the output layer parameters, $g(\cdot)$ is an activation function, such as sigmoid, sine, and tanh. Authors in [123] mentioned that the neuron weights $\mathbf{W}$ and $\boldsymbol{\beta}$ could be called the Fishier vector and calculated via solving linear equations through standard numerical methods, such as SGD and MP inverse. Furthermore, when a MLNN is constructed, the above learning model is updated with the cost function as:

$$\begin{aligned} &\text{minimize} \;\; J = \frac{1}{2}\sum_{j=1}^{N} ||\mathbf{T} - f(\mathbf{X}, \mathbf{W})\boldsymbol{\beta}||^2, \\ &f(\mathbf{X}, \mathbf{W}) = f_n\left(\cdots f_2\left(f_1\left(\mathbf{X}, \mathbf{W}_1\right), \mathbf{W}_2\right)\cdots, \mathbf{W}_n\right), \end{aligned} \tag{4.2}$$

where $\mathbf{W}_i$ and f_i are the i-th hidden layer parameters and activation function respec-

Table 4.1 – Notations used in this Chapter

Notation	Meaning
Γ	the feature learnt from the exit layer (global-level representation)
H_i	feature extracted from the i-th subnet in entrance layer (S-node)
Ψ_i	feature extracted from the i-th RS-node in feature refinement layer
$\mathbf{b}_i^f$	bias of SNN feature layer, which is generated randomly
$\mathbf{b}_i^r$	bias of feature refinement layer, which is generated randomly
$\mathbf{E}_i$	the error matrix
I	the identity matrix
L	the total number of iterations
m	the dimension of output layer (labels)
N	the number of training samples
n	the dimension of input
$\mathbf{P}_i$	the error feedback data in feature refinement layer
D	the dimension of S-node
d	the dimension of RS-node
$\mathbf{T}$	the expected output
$\mathbf{W}_i^f$	weight of the i-th subnetwork in entrance layer (S-node)
$\mathbf{W}_i^r$	weight of the i-th subnetwork in feature refinement layer (RS-node)
$\mathbf{W}_i^v$	parameters of the output layer
$\mathbf{X}$	the input feature

tively.

The proposed model, on the other hand, is an integration of subnetworks, which are built upon several sub-nodes of neurons as depicted in Fig. 4.2. Therefore, the cost function in Eq. (4.2) is further modified as:

$$\begin{aligned} &\text{minimize} \;\; J = \frac{1}{2}||\mathbf{T} - f(\mathrm{H}_i, \mathbf{W}_i^r, \mathbf{b}_i^r) \cdot \mathbf{W}_L^v||^2, \\ &f(\mathrm{H}_i, \mathbf{W}_i^r, \mathbf{b}_i^r) = \Gamma = \sum_{i=1}^{L} g(\mathrm{H}_i \cdot \mathbf{W}_i^r + \mathbf{b}_i^r), \end{aligned} \tag{4.3}$$

where $\mathbf{T} \in \mathcal{R}^{N\times m}$ is the expected output, $\mathbf{X} \in \mathcal{R}^{N\times n}$ is the input matrix, L is the total number of RS-node nodes, $\mathbf{W}_i^r$ and $\mathbf{W}_i^v$ are parameters for refinement layer and output layer respectively. H_i is the entrance layer feature matrix, which is generated through entrance layer parameters ($\mathbf{W}_i^f$, $\mathbf{b}_i^f$) and original feature $\mathbf{X}$. $g(\mathrm{H}_i\mathbf{W}_i^r + \mathbf{b}_i^r)$

is the features extracted from the i-th RS-node, and $\Gamma = \sum_{i=1}^{l} g(\mathrm{H}_i\mathbf{W}_i^r + \mathbf{b}_i^r)$ is the exit layer feature matrix. Note that all the layer bias $\mathbf{b}_i^f$ and $\mathbf{b}_i^r$ are randomly selected. Based on Eq. (4.3), the global-level latent space encoding Γ defined in this dissertation is elaborated as Theorem 4.1.

Theorem 4.1. *Given N arbitrary distinct samples $\{\boldsymbol{X},\ \boldsymbol{T}\}$, where $\boldsymbol{X} \in \boldsymbol{R}^{N\times n}$ is the input data and $\boldsymbol{T} \in \mathcal{R}^{N\times m}$ is the expected output, respectively. We can get the global-level latent space feature $\Gamma = \sum_{i=1}^{L} \Psi_i = \sum_{i=1}^{L} g(\mathrm{H}_i \cdot \boldsymbol{W}_i^r + \boldsymbol{b}_i^r)$, which satisfies $\lim_{L\to\infty} \| \left(\sum_{i=1}^{L} g(\mathrm{H}_i \cdot \boldsymbol{W}_i^r + \boldsymbol{b}_i^r)\right) \cdot \boldsymbol{W}_L^v - \boldsymbol{T}\| = 0$ if*

$$
\begin{aligned}
&\boldsymbol{W}_i^r = \left(\frac{I}{C} + (\mathrm{H}_i)^T\mathrm{H}_i\right)^{-1} (\mathrm{H}_i)^T\boldsymbol{P}_{i-1},\\
&\boldsymbol{P}_{i-1} = \boldsymbol{E}_{i-1} \cdot \left(\frac{I}{C} + (\boldsymbol{W}_{i-1}^v)^T\boldsymbol{W}_{i-1}^v\right)^{-1} (\boldsymbol{W}_{i-1}^v)^T,\\
&\mathrm{H}_i = g(\boldsymbol{X} \cdot \boldsymbol{W}_i^f + \boldsymbol{b}_i^f),\\
&\boldsymbol{W}_L^v = (\frac{I}{C} + \Gamma^T\Gamma)^{-1}\Gamma^T\boldsymbol{T},
\end{aligned}
\tag{4.4}
$$

where $\boldsymbol{P}$ is the pulled back error, $\boldsymbol{W}_i^v$ is the output layer weights, H_i and $\boldsymbol{W}_i^f$ are the i-th S-node's feature and weights, Ψ_i and $\boldsymbol{W}_{ex}^i$ are the i-th RS-node's feature and weights, respectively. Note that $\boldsymbol{W}_i^f$ is randomly assigned for alleviating overfitting.

4.2.1 The Proposed Wi-HSNN

The proposed RL structure has three feature spaces: entrance feature space [H] (entrance layer), refinement feature space [Ψ] (refinement layer) and exit feature space [Γ]. The first space is generated randomly to maintain the universal performance. The second and third spaces are calculated or updated through the pulled back error. The training process for this strategy consists of two core phases: **Initialization** phase and **Iterative subspace learning** phase.

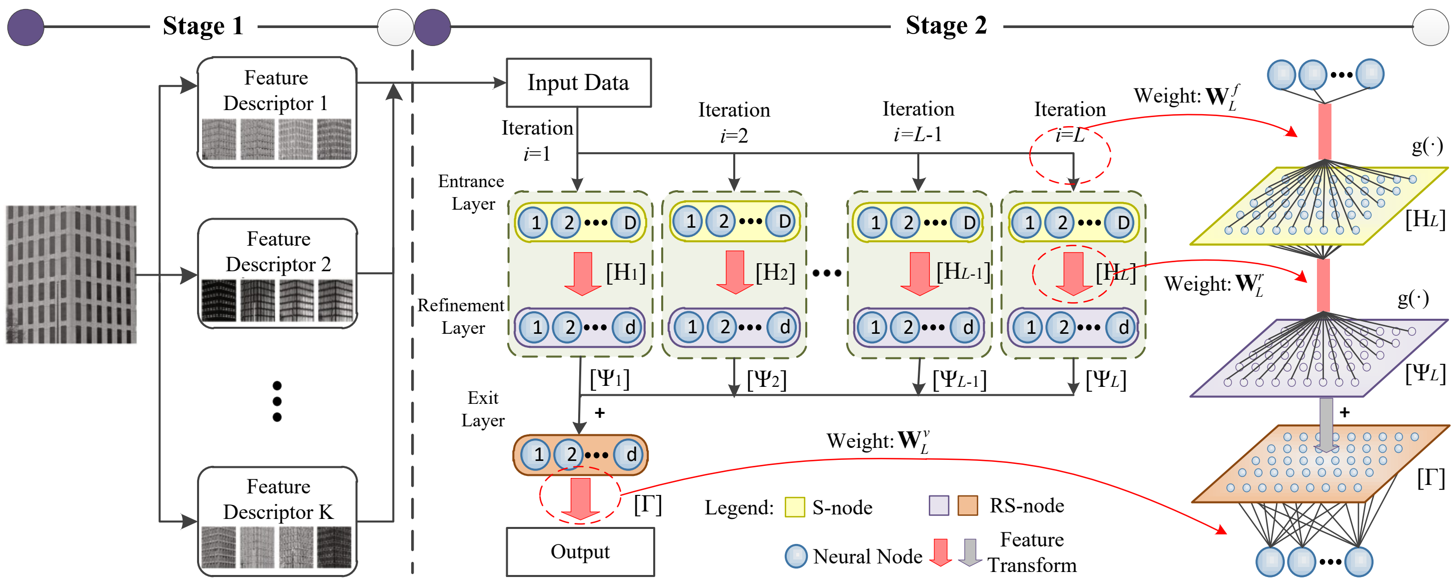


Figure 4.2 – The proposed triple hidden layer "width-growth" structure. In each iteration a new subnetwork pair (S-node and RS-node) are added to the representation learning model to update the optimal feature space: $\mathbf{W}_i^f$ - Entrance layer parameters of i-th RS-node, $\mathbf{W}_i^r$ - Refinement layer parameters of i-th RS-node and $\mathbf{W}_i^v$ - Output layer parameters. Notations: H_i - i-th entrance space feature, Ψ_i - i-th refinement space feature, Γ - global-level representation space feature (exit layer), and $g(\cdot)$ - activation function.

The first phase is to randomly construct the suitable lower feature space. The raw feature is firstly projected randomly from original space [**X**] into entrance feature space [H] and refinement feature space [Ψ]. Then, the least square method is performed to find the projection from the global-level representation space [Γ] to the output label space.

The second phase may be repeated for a given number of times to enhance the quality of the feature representation through re-computations of the output layer parameters, and the feature space [Γ] is further learnt by the current classification results. Assume there are N arbitrary feature samples, $\psi = \{(\mathbf{x}_k, \mathbf{t}_k)\}_{k=1}^{N}$ ($\mathbf{x}_k \in \mathcal{R}^n, \mathbf{t}_k \in \mathcal{R}^m$), $\mathbf{X} \in \mathcal{R}^{N\times n}$ represents the input data and $\mathbf{T} \in \mathcal{R}^{N\times m}$ represents the desired output data, the maximum iteration is L. The training steps for this method is shown as follows:

Step 1: For iteration i=1, the parameters of the entrance layer S-node ($\mathbf{W}_i^f, \mathbf{b}_i^f$) and the refinement layer RS-node ($\mathbf{W}_i^r, \mathbf{b}_i^r$) are randomly initialized, as depicted in Fig. 4.3. Thus, the feature matrices H_i and Ψ_i are computed as

$$\begin{aligned} \mathrm{H}_i &= g(\mathbf{W}_i^f, \mathbf{b}_i^f, \mathbf{X}), \text{ and} \\ \Psi_i &= g(\mathbf{W}_i^r, \mathbf{b}_i^r, \mathrm{H}_i), \end{aligned} \tag{4.5}$$

where $g(\cdot)$ is the activation function. For sine and sigmoid functions, it will be $sin(\cdot)$ and $1/(1+e^{(\cdot)})$ respectively.

Step 2: Set $\Gamma = \Psi_i$. Here, $i = 1$. Calculate the output layer parameters $\mathbf{W}_i^v$ by using the LS method:

$$\mathbf{W}_i^v = (\frac{I}{C} + \Gamma^T\Gamma)^{-1}\Gamma^T \cdot \mathbf{T}, \tag{4.6}$$

where $(I/C + \Gamma^T\Gamma)^{-1}$ is the Moore-Penrose inverse of Γ, C is the regularization term. The error can be obtained $\mathbf{E}_i = \mathbf{T} - \Gamma\mathbf{W}_i^v$.

Step 3: Obtain the desired feature matrix of refinement layer $\mathbf{P}_i$.

$$\mathbf{P}_i = \mathbf{E}_i \cdot \left(\frac{I}{C} + (\mathbf{W}_i^v)^T\mathbf{W}_i^v\right)^{-1} (\mathbf{W}_i^v)^T. \tag{4.7}$$

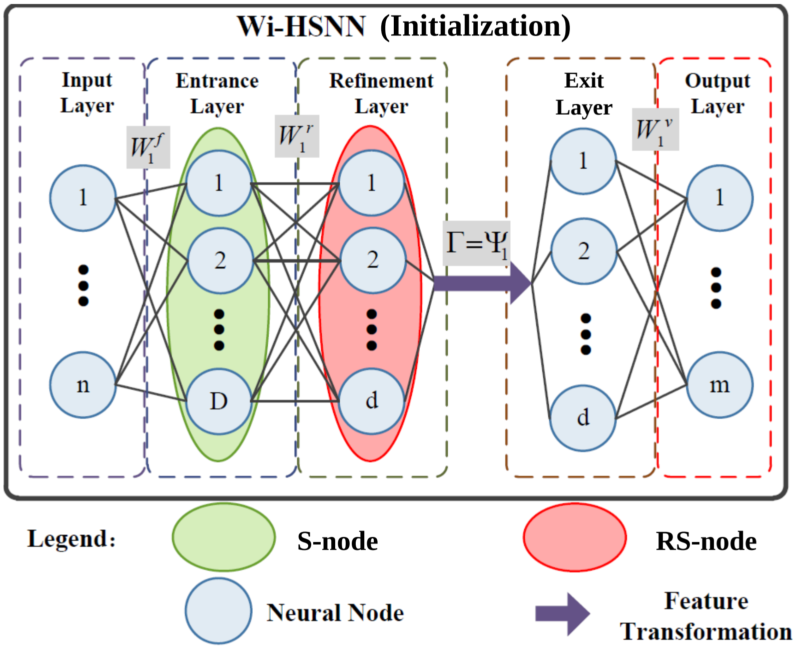


Figure 4.3 – Comparison results of Wi-HSNN, MSNN, MP-H, and MP-ML among all the food image classification datasets.

Step 4: Set i=i+1. Add new S-node and RS-node with D and d hidden nodes to this network. Randomly generate $\mathbf{W}_i^f$. The refinement layer parameters $\mathbf{W}_i^r$ and feature Ψ_i can be described as:

$$\begin{aligned}
&\mathbf{W}_i^r = (\frac{I}{C} + \mathrm{H}_i{}^T\mathrm{H}_i)^{-1}\mathrm{H}_i{}^T \cdot g^{-1}(\mathbf{P}_{i-1}),\\
&\Psi_i = g(\mathbf{W}_i^r, \mathbf{b}_i^r, \mathrm{H}_i),
\end{aligned} \tag{4.8}$$

where $\mathrm{H}_i = g(\mathbf{W}_i^f, \mathbf{b}_i^f, \mathbf{X})$ is the output of the i-th entrance layer S-node, g^{-1} is the inverse of activation function.

Step 5: Update the i-th iteration optimal feature $\Gamma = \sum_{j=1}^{i} \Psi_j$, recalculate the output layer parameters through (4.6). The output error is updated via:

$$\mathbf{E}_i = \mathbf{E}_{i-1} - \mathbf{W}_i^v \Gamma. \tag{4.9}$$

Step 6: Repeat Step 3 - 5 $L-2$ times, and the feature matrix Γ is the generalized feature. The structure of the well-trained Wi-HSNN is shown in Fig. 4.4.

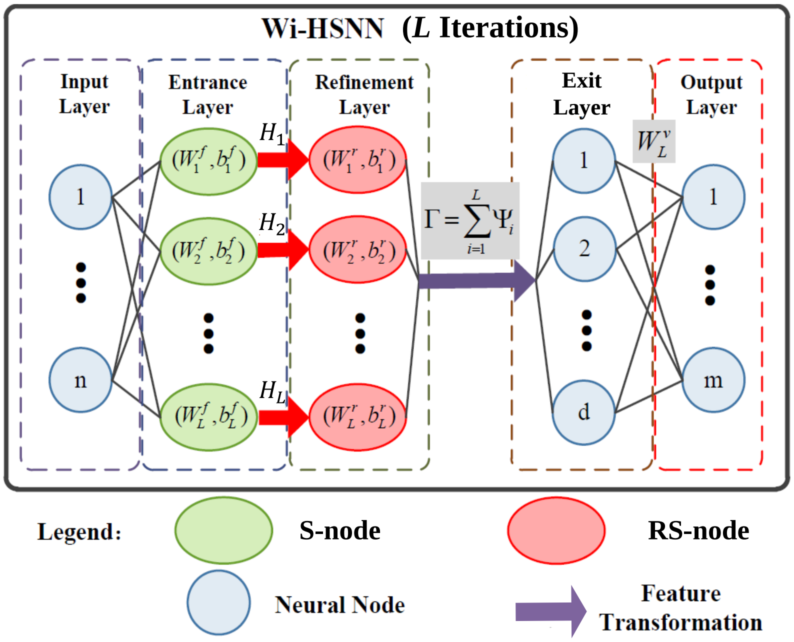


Figure 4.4 – Comparison results of Wi-HSNN, MSNN, MP-H, and MP-ML among all the food image classification datasets.

When the feature matrix Γ_i comes with the optimal feature data, the corresponding output parameter $\mathbf{W}_i^v$ is adopted to calculate the final classification result.

In general, there are three common differences between the proposed subnetwork-based RL model and other multilayer analytic learning methods.

1) Different from other multilayer analytic learning methods, which only construct with neurons, the hidden layers in this model consist of two kinds of subnetwork nodes, S-node and RS-node, the former one extract subspace features from the input data, can be considered as a local feature descriptor; while the RS-node, which is comprised of SNN layer S-node and refinement layer S-node, aims to recast the obtained local features into a more discriminative and generalized feature coding. In this sense, the hidden layers can refine and encode the input features.

2) The framework encodes the optimal feature representations in several training iterations, whereby in each iteration a new S-node and RS-node are added to the structure. The new RS-node is only densely connected to the entrance layer S-node and least-square learning layer without any impact of the previous RS-node, so that the updated space can maintain the image statistics and knowledge cues well.

3) Compared with other stacked multi-layer analytic learning networks: hierarchical network [42], multi-layer network [24], the proposed algorithm can train all the parameters jointly, rather than find the feature spaces and recognize the objects separately, thus resulting in global optimal results. More importantly, it uses a width-growth structure that achieves a highly competitive classification performance.

4.2.2 Algorithmic Summary

Figure 4.2 summarizes the entire process of the proposed representation learning and classification algorithm. It has the following two computational stages:

- **Stage 1: Pre-processing stage** - The feature vectors from various channels and descriptors (Ω) are concatenated into one super-vector. Precisely, the features from different descriptors are normalized in the range, $[0, 1]$ and concatenated horizontally.

- **Stage 2: Width-growth model training stage** - This stage contains two substages.
  Stage 2.1: Initialization ($i = 1$) - The entrance layer weight, $\mathbf{W}_1^f$ and the refinement layer weight, $\mathbf{W}_1^r$ are generated randomly. The output layer weight, $\mathbf{W}_1^v$ is calculated through LS method. The subspace features H_1, Ψ_1, and Γ are generated via Step 1 and Step 2. Furthermore, the pulled back error term $\mathbf{P}_1$ is calculated through Step 3. Hence, the current feature refinement space Γ is randomly generated.

  Stage 2.2: Iterative subspace learning ($2 \leq i \leq L$) - The $\mathbf{W}_i^f$ is randomly initialized to reduce the overfitting problem and avoid getting trapped in local minima. However, the refinement layer weights, $\mathbf{W}_i^r$ are computed with $\mathbf{P}_{i-1}$; hence, the new refinement subspace Ψ_i is guided by the current error term, $\mathbf{E}_{i-1}$. The feature representation space Γ is considered as the aggregation of all the refinement sub-spaces $\Psi_1, \cdots, \Psi_i$. Similar to initialization stage, users need to update the pulled back error term, $\mathbf{P}_i$. Details for this stage are described from Step 4 to Step 6. It is noted that the optimal value for the iteration L is selected empirically.

4.2.3 Proof of The Proposed Wi-HSNN

In this subsection, the Theorem 4.1 is proved. We prove that if the network continues adding subnetworks (S-node and RS-node) to the structure, the output error sequence $||\mathbf{E}_i|| = \| \left(\sum_{j=1}^{i} g(\mathrm{H}_j \cdot \mathbf{W}_j^r + \mathbf{b}_j^r)\right) \cdot \mathbf{W}_i^v - \mathbf{T}\|$ will be decreasing and bounded by zero.

Proof. Suppose $\Gamma(i)$ and $\Gamma(i+1)$ defined in Eq. (4.10) are the data representations of the i-th iteration and $i+1$-th iteration, respectively.

$$\begin{aligned} \Gamma(i) &= \sum_{j=1}^{i} \Psi_j = \Psi_1 + \Psi_2 + \cdots + \Psi_i, \\ \Gamma(i+1) &= \sum_{j=1}^{i+1} \Psi_j = \Psi_1 + \Psi_2 + \cdots + \Psi_{i+1}. \end{aligned} \tag{4.10}$$

Initially, we have the following expression:

$$\Gamma(i+1) = \Gamma(i) + \Psi_{i+1}. \tag{4.11}$$

The L_2 norm of the error matrix L_i^2 of the i-th iteration is

$$L_i^2 = \|\mathbf{E}_i\|^2 = \|\mathbf{T} - \Gamma(i)\mathbf{W}_i^v\|^2 = \|\mathbf{P}_i\mathbf{W}_i^v\|^2. \tag{4.12}$$

Similarly, we have

$$\begin{aligned} L_{i+1}^2 &= \|\mathbf{T} - \Gamma(i+1)\mathbf{W}_{i+1}^v\|^2 \\ &= \|\mathbf{T} - \Gamma(i+1)(\mathbf{W}_i^v + \Delta\mathbf{W}_i^v)\|^2, \end{aligned} \tag{4.13}$$

where $\mathbf{W}_i^v + \Delta\mathbf{W}_i^v = \mathbf{W}_{i+1}^v$ is the optimal output layer weights. Since $\mathbf{W}_{i+1}^v$ is one of the least square solutions, we have the following judgement.

$$\begin{aligned} L_{i+1}^2 &= \min\|\mathbf{T} - \Gamma(i+1)\mathbf{W}_o\|^2 \\ &\leq \|\mathbf{T} - \Gamma(i+1)\mathbf{W}_i^v\|^2 \end{aligned} \tag{4.14}$$

According to Eq. (4.12) and Eq. (4.13), we can get

$$\begin{aligned} L_i^2 - L_{i+1}^2 &= \|\mathbf{E}_i\|^2 - \|\mathbf{E}_{i+1}\|^2 \\ &\geq \|\mathbf{P}_i\mathbf{W}_i^v\|^2 - \|\mathbf{T} - \Gamma(i+1)\mathbf{W}_i^v\|^2 \\ &= \|\mathbf{P}_i\mathbf{W}_i^v\|^2 - \|\mathbf{T} - \left(\Gamma(i) + \Psi_{i+1}\right)\mathbf{W}_i^v\|^2 \\ &= \|\mathbf{P}_i\mathbf{W}_i^v\|^2 - \|\mathbf{T} - \Gamma(i)\mathbf{W}_i^v - \Psi_{i+1}\mathbf{W}_i^v\|^2 \\ &= \|\mathbf{P}_i\mathbf{W}_i^v\|^2 - \|\mathbf{E}_i - \Psi_{i+1}\mathbf{W}_i^v\|^2 \\ &= \|\mathbf{P}_i\mathbf{W}_i^v\|^2 - \|\mathbf{P}_i\mathbf{W}_i^v - \Psi_{i+1}\mathbf{W}_i^v\|^2 \end{aligned} \tag{4.15}$$

Nonetheless, the $(i+1)$-th RS-node and RS-node target at offsetting the feedback data $\mathbf{P}_i$ obtained from the previous iteration, there still has the residual error $\boldsymbol{\sigma}$ between $\mathbf{P}_i$ and Ψ_{i+1}, and $\|\boldsymbol{\sigma}\|^2 << \|\mathbf{P}_i\|^2$. Thus, we have

$$\Psi_{i+1} + \boldsymbol{\sigma} = \mathbf{P}_i. \tag{4.16}$$

With Eq. (4.16), the Eq. (4.15) can be simplified as

$$\begin{aligned} L_i^2 - L_{i+1}^2 &\geq \|\mathbf{P}_i\mathbf{W}_i^v\|^2 - \|\mathbf{P}_i\mathbf{W}_i^v - \left(\mathbf{P}_i - \boldsymbol{\sigma}\right)\mathbf{W}_i^v\|^2 \\ &= \|\mathbf{P}_i\mathbf{W}_i^v\|^2 - \|\boldsymbol{\sigma}\mathbf{W}_i^v\|^2 \\ &= \left(\|\mathbf{P}_i\|^2 - \|\boldsymbol{\sigma}\|^2\right)\|\mathbf{W}_i^v\|^2 \geq 0 \end{aligned} \tag{4.17}$$

Thus, $L_i^2 \geq L_{i+1}^2$. Further, we can have $\|\mathbf{E}_\infty\| \leq \cdots \leq \|\mathbf{E}_{i+1}\| \leq \|\mathbf{E}_i\| \leq \cdots \leq \|\mathbf{E}_1\|$. Thus, we conclude that the sequence $||\mathbf{E}_i|| = \| \left(\sum_{j=1}^{i} g(\mathrm{H}_j \cdot \mathbf{W}_j^r + \mathbf{b}_j^r)\right) \cdot \mathbf{W}_i^v - \mathbf{T}\|$ is decreasing and bounded to zero.

□

4.3 Experimental Results

4.3.1 Experimental Setup

4.3.1.1 The Environment

The experiments were performed in Matlab 2017b programming paradigm on a 2.8 GHz E5-2650 processor and 256 GB memory workstation with a single NVIDIA 1080Ti GPU. All the experiments recorded in this work are the mean average performance from a minimum of three trials.

4.3.1.2 The Datasets

The proposed model was tested on several benchmark datasets from the application areas of image classification to food image classification as summarized in Table 4.2 and Table 4.3. Along with dataset details, the framework settings are also given. These datasets fall into three categories: small-scale, large-scale and camera model identification datasets. The small-scale datasets have a marginally less number of data samples with less than $40K$. Scene-15, LabelMe, Sports, Caltech-101, Caltech-256, MIT-67, and SUN-397 are the seven small-scale datasets. While, Places-365-1, Places-365-2, and Places-365-3 are the three large-scale scene classification datasets that contain more than $40K$ samples. Here, the SPCUP is a camera identification dataset.

Small-scale datasets ($< 40K$ samples) Following the commonly used training settings [23, 3], we took 100 (Scene-15), 30 (Caltech-101/256), 80 (MIT-67) and 50 (SUN-397) samples per category for training, and the rest for evaluation.

Large-scale datasets ($> 40K$ samples) The versions of Places-365 contain a minimum of 1.8 million images comprising 365 unique scene categories. Because of hardware limitations, we randomly selected 500, 1000, and 1500 samples per class to form Places-365-1, Places-365-2, and Places-365-3 datasets, respectively.

Table 4.2 – Summary of the image classification and camera model identification datasets

Purpose	Dataset		Classes	Total Samples	Training Samples	Testing Samples	Neurons in S-node	Neurons in RS-node	Training Iterations (L)
Image Classification	Small	Scene-15	15	4,485	1,500	2,985	100	80	5
		MIT-67[1]	67	6,700	5,360	1,340	300	200	5
		Caltech-101	102	9,144	3,060	6,084	300	200	5
		Caltech-256	257	30,608	7,710	22,898	300	200	5
		SUN-397	397	39,700	19,850	19,850	500	300	5
	Large	Places-365-1	365	182,500	146,000	36,500	500	300	10
		Places-365-2	365	365,000	292,000	73,000	500	300	10
		Places-365-3	365	547,500	438,000	109,500	500	300	10
Camera Identification		SPCUP	6 or 10	2,750	2,000	750	100	80	5

[1] The original MIT-67 set contains 15,620 images. In this chapter, we randomly select 100 samples per category to form the new set.

Table 4.3 – Summary of the food image classification datasets

Type	Dataset	Classes	Total of Samples	Training Samples	Testing Samples	Neurons in S-node	Neurons in RS-node	Training Iterations (L)
Food Image Classification	UEC-100	100	14,361	11,492	2,869	300	200	5
	UEC-256	256	31,394	25,117	6,277	500	300	5
	Food-101	101	101,000	75,750	25,250	300	200	5
	Food-251m	251	71,292	59,298	11,994	500	300	3
	Food-251a	251	130,469	118,475	11,994	500	300	5

Camera calibration dataset (2750 samples) The SPCUP [126] dataset contains a total of 2750 images comprising six manufacturers and ten separate camera models. The camera source identification has been attracting a lot of attention recently. It enables us to discover the most possible source camera model that can be used to verify the authenticity of an image. Currently, the camera identification methods in literature can be basically divided as four categories: Sensor pattern noise (SPN)-based methods [127], Color filter array (CFA) interpolation-based methods [128] and CNN-based methods [126]. In this chapter, the proposed framework is adopted as the basic detection model to verify the performance of the algorithm on a specific application.

Food datasets Image-based food pattern classification poses new challenges for mainstream computer vision algorithms. UEC-100 [129], Food-101 [130], Food-251 [131], and UEC-256 [132] are four commonly used food image classification datasets. Following the previous settings, we took 75% (Food-101) and 80% (UEC-100 and UEC-256) of the total training samples for use in training, then used the rest for evaluation. In addition, Food-251 is one of the latest food classification datasets created in 2019. We randomly selected 50% of training images for training and used all validation images for testing to generate the Food-251m set, and utilized the original Food-251 set (the training set used for training and the validation set used for testing) to validate the algorithm.

4.3.1.3 The Rival Methods and The Configuration of Input

This section evaluates this method with more than 30 state-of-the-art methods, which can be categorized into two families.

Traditional RL methods It includes SLNN-based methods, such as Random forest (RF) [133], Softmax [134], multilayer MP inverse-based representation learning network (MP-ML) [42], hierarchical network (MP-H) [24], and multilayer subnet neural network (MSNN) [23].

To enable fair comparisons, we extracted the same features from the penultimate layer of some DCNNs, like AlexNet [86], VGG16 [46], DenseNet [135], Inception-v3 [87], and ResNet [88]. For AlexNet and VGG16, we adopt the features from the activation of the fully connected layer ($fc7$), which is a 4096-dimensional feature vector. For Inception-v3, we use the 2048-dimensional vector from the average pooling layer (avg_pool). As for ResNet and DenseNet, the 2048- and 1920-dimensional features are extracted from the $fc1000$ layer, respectively. These high-level features are extracted under two conditions: feature extraction using a pre-trained DCNN without target-domain fine-tuning and with target-domain fine-tuning. This chapter denotes them "P"-network name and "T"-network name respectively.

1. "P"-DCNN features: These features are directly extracted from a pre-trained DCNN without the intervention of the target dataset. In this chapter, the ImageNet pre-trained DCNNs are used. Because the pre-trained networks have learnt the knowledge and information from natural images, these features only contain innate priors regarding the ImageNet. This chapter uses the prefix "P" like $P - AlexNet$, $P - ResNet$ to flag that the features belong to this type.

2. "T"-DCNN features: Technically, these features are extracted from a DCNN, which is initialized by pre-trained parameters and fine-tuned on the target dataset. Precisely, an ImageNet pre-trained network is fine-tuned with target domain samples in an end-to-end manner. After a few training epochs (2 to 3), features are extracted from a particular layer as stated earlier. Thus, these features contain a cross-domain knowledge from ImageNet and the target domain. The prefix "T", like $T - VGG16$ is used to notify that the features extracted using this fine-tuning approach.

Camera model identification methods The following algorithms were compared: CNN-based [126] algorithm, Mixed Model [126], feature fusion using texture data [127], and the ensemble classifier [128].

Food Image classification methods These algorithms include the Fishier vector [136], mid-level patches [136], AlexNet-based algorithm [137], early fusion method [138], mid-level fusion algorithm [139], fusion of DCNNs [140], and InceptionNet-based structure [141].

4.3.1.4 Evaluation Matrics

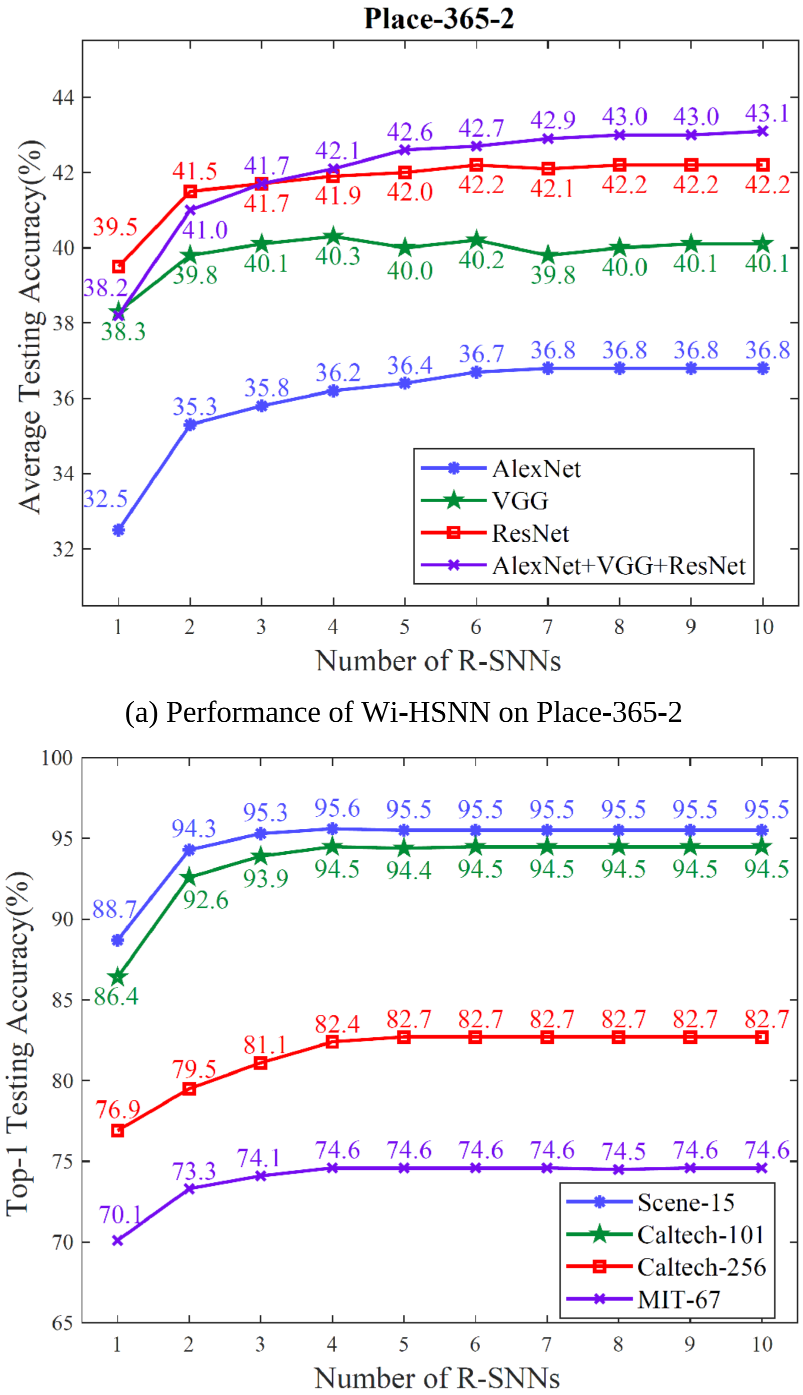
The experimental results of the proposed algorithm are quantitatively compared with many state-of-the-art approaches. In this chapter, the Top-1 accuracy, which is a measure of system bias, is adopted to evaluate the performance of a certain algorithm.

4.3.2 Model Settings

The proposed solution requires a user decision – the choice of iteration L. In this subsection, we empirically set L based on its merit. Recommendations are given for a default configuration, reducing the need for user-tuning of the hyperparameter.

Figure 4.5 (a) presents the Top-1 accuracy of the proposed scheme on a large-scale dataset with unimodal feature coding (AlexNet, VGG-16, and ResNet) and with multimodal feature coding with the aforesaid feature extractors. Figure 4.5 (b) shows the performance of Top-1 testing accuracy on small-scale datasets as new E-node and RS-node are added to the proposed refinement framework. It is observed that the performance of the proposed approach converges after $L = 8$ for large-scale datasets and after $L = 4$ on small-scale datasets. To be consistent and ensure the structure is fully trained, the total number of training iterations L is set to 5 and 10 for small-scale and large-scale datasets, respectively.

Furthermore, to verify the intuition of the multi-view receptive field, a sanity check with a different combination of complementary feature flows is conducted, which is shown in Table 4.4. The investigation results reveal that the fusion of all the complimentary feature flows provides superior performance than those results derived from a single feature channel. For example, on Food-101 and Food-251a datasets, the average performance of all feature combinations gains roughly 8.8% and 6.2% boost over the Wi-HSNN with ResNet features, respectively. The reason could be as follows: Firstly, pattern recognition with single-channel features suffers from coding bias due to one-sided feature description, while framework trained with multiple channel fusion strategy overcomes such pitfall and provides more robust performance. Secondly, the proposed Wi-HSNN transforms features from their original feature space to a generalized latent space. Multiple feature channels enable the network to search a global-level encoding and representation.



(a) Performance of Wi-HSNN on Place-365-2

(b) Performance of Wi-HSNN on small-scale datasets
(AlexNet+ VGG+ResNet)

Figure 4.5 – Top-1 accuracy of Wi-HSNN wrt various hyper-parameter L (Input: "T"-DCNN features).

Table 4.4 – The effectiveness analysis of feature fusion.

Feature	UEC-100	Food-101	Food-251a	UEC-256
ResNet	85.2	82.0	60.2	76.2
Inception	86.5	84.7	63.2	77.7
DenseNet	80.4	82.9	57.6	72.1
ResNet+Inception	87.2	88.9	65.9	79.5
All	**87.7**	**90.8**	**66.4**	**83.1**

4.3.3 Analysis on Image Classification Domain

The classification performance concerning Places-365-1 Places-365-2 and Places-365-3 datasets are tabulated in Table 4.5, where the best results are in boldface format. For Places-365-1, it turns out that the proposed method provides a comparable recognition performance, which is 1.3% better than fine-tune AlexNet, just 0.4% lower than fine-tune VGG network. In the case of Places-365-2 and Places-365-3 datasets, the fused transfer learning results are all better than fine-tune AlexNet with 1.5% and 1.1% improvement respectively. Combined with the performance listed in Table 4.5, we conclude that the proposed framework obtains a competitive performance with an acceptable computational burden.

4.3.4 Analysis on Extended Domains

The detection results for the SPCUP set are shown in Table 4.6. We use the same input feature and pre-processing steps as [127], which is derived from the HSV model and contourlet decomposition results. As seen from Table 4.6, compared with the original feature fusion method, which is the best among these comparison methods, the average identification accuracy of the proposed method increases by 0.5% in the case of 10-camera model identification. Meanwhile, the algorithm has a significant improvement on 6-manufacturer identification, which is over 6% boost. With the same input features, the proposed representation learning algorithm achieves competitive identification performance compared to the work in [127].

Table 4.5 – Classification accuracy for CNN features with various algorithms on Place365 dataset (A.(%) - testing accuracy; T.(h) - training time in hour)

Methods	Places-365-1		Places-365-2		Places-365-3	
	A.(%)	T.(h)	A.(%)	T.(h)	A.(%)	T.(h)
Deep learning models and deep features						
VGG16, 20 epochs	42.3	168	44.2	265	45.8	326
AlexNet, 20 epochs	40.6	33	41.5	57	42.6	72
SVM, P-VGG16	28.2	31	29.9	43	31.1	75
SVM, P-AlexNet	25.9	31	26.4	45	28.5	73
SVM, T-VGG16	35.2	55	36.8	68	37.2	122
SVM, T-AlexNet	30.9	35	32.7	54	34.0	85
Wi-HSNN, unimodal feature coding						
P-VGG16	32.6	1	33.7	2	33.9	3
P-AlexNet	28.9	1	31.2	2	36.8	3
T-VGG16	39.1	25	40.1	42	40.1	55
T-AlexNet	34.1	6	36.8	10	37.5	15
Wi-HSNN, multimodal feature coding						
P-AlexNet P-VGG16	33.9	2	35.1	4	37.4	6
T-AlexNet, T-VGG16	40.7	26	41.4	52	42.2	66
T-VGG, T-Resnet, T-AlexNet	**41.9**	30	**43.0**	57	**43.7**	75

[1] Time consumption on the proposed method with transfer learning features contains CNN training time and representation learning time.
[2] "P"-DCNN features - Features directly extracted from a pre-trained DCNN without an intervention of the target dataset; "T"-DCNN features - Features extracted from a DCNN, which is initialized by a pre-trained parameters and fine-tuned on the target dataset.

Table 4.6 – Performance of the Wi-HSNN on camera model identification dataset.

Methods	10-camera Identification	6-manufacturer Identification
CNN-based [126]	96.5	97.3
Mixed Model [126]	96.7	98.5
Texture, feature fusion [127]	98.1	92.1
Ensemble classifier [128]	97.3	**98.9**
Wi-HSNN	**98.9**	98.4

Table 4.7 – Comparison of methods on Food datasets. The best and the second best results are highlighted in boldface and underlined, respectively [All - input is the concatenated (ResNet, DenseNet, and InceptionNet) feature].

Method	UEC-100	Food-101	Food-251m	Food-251a	UEC-256	Average
Traditional food classification methods from the literature						
Fishier Vector [136]	35.8	38.9	30.1	32.1	38.2	35.0
Mid-level patches [136]	40.3	42.6	38.5	39.4	41.1	40.4
AlexNet-based [137]	78.8	70.4	45.3	48.9	67.6	62.2
Early Fusion [138]	80.0	72.1	47.5	48.2	68.5	63.3
Mid-level Fusion [139]	86.5	87.8	48.7	51.1	78.6	70.5
Fusion of DCNNs [139]	87.7	88.1	54.3	57.9	76.8	73.0
Inception-based [141]	**88.5**	88.0	50.4	52.6	76.2	71.1
Inception-v3 feature with classifiers and representation methods						
Inception + RF [133]	78.2	80.1	50.2	54.2	72.1	67.2
Inception + Softmax [134]	83.0	82.8	54.5	60.4	75.1	71.4
Inception + MP-ML [42]	82.0	81.9	57.1	61.2	77.1	72.1
Inception + MP-H [24]	83.8	80.5	58.3	62.0	76.0	72.4
Inception + MSNN [23]	84.0	83.6	59.2	61.7	76.8	73.3
Inception + Wi-HSNN	86.3	84.7	58.9	63.2	77.7	74.4
Concatenated feature with classifiers and representation methods						
All + RF [133]	83.7	87.6	54.4	58.0	80.3	73.0
All + Softmax [134]	86.3	88.3	58.8	64.3	79.1	75.4
All + MP-ML [42]	84.3	85.9	59.1	62.5	81.5	74.7
All + MP-H [24]	85.3	86.7	60.0	63.7	80.2	74.4
All + MSNN [23]	86.8	88.2	60.2	64.9	79.1	76.0
All + Wi-HSNN	87.7	**90.8**	**61.6**	**66.4**	**83.1**	**78.2**

However, the ensemble classifier in [128] achieves a mere 0.5% of classification accuracy improvement when compared to the other models. It is because the authors of [128] construct a rich variety of demosaicing submodels, partial information of different camera manufacturers could be obtained, resulting in a comprehensive representation.

The quantitative analysis of food image classification is provided in Table 4.7 as a comparison between the proposed Wi-HSNN and other algorithms, ranging from the existing food classification methods to CNN features plus encoding techniques.

Figure 4.6 highlights the overall comparison of the proposed Wi-HSNN with other state-of-the-art representation learning frameworks regarding Top-1 testing accuracy among all comparison datasets. Our findings show that the proposed Wi-HSNN harnessing with multi-CNN high-level features exhibits robustness, with average Top-1 testing accuracy of 90.8%, 61.6%, 66.4%, and 83.1% on Food-101, Food-251m, Food-251a, and UEC-256 datasets. The overall improvements of the proposed model are 2.6%, 1.4%, 1.5%, and 4.0% compared to the current leading results for these datasets, respectively. In addition, when considering the average performance among datasets, Wi-HSNN has a much more consistent performance than MSNN [23]. The proposed Wi-HSNN achieves 2.2% higher mean average Top-1 testing accuracy than MSNN.

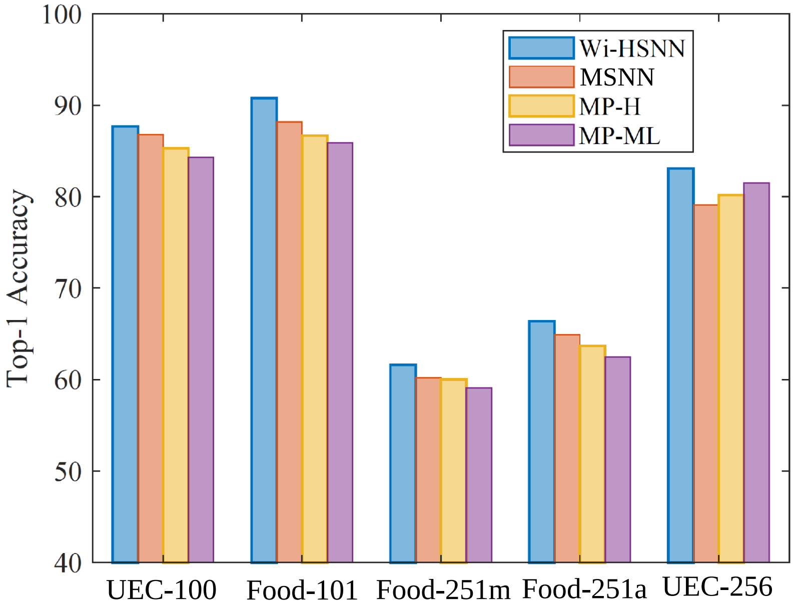


Figure 4.6 – Comparison results of Wi-HSNN, MSNN, MP-H, and MP-ML among all the food image classification datasets.

Figure 4.7 summarizes the performance of Wi-HSNN with other representation learning and classification methods among all of the food and scene datasets via a box-plot. When considering the average performance across categories, the proposed Wi-HSNN has more consistent results than all the other representation learning methods. This could occur because the Wi-HSNN encodes and refines the latent space in a supervised manner, directly updating the optimal feature with the current feedback error term. However, most MP inverse-based encoding methods represent the raw feature and generate the latent space in an unsupervised manner, thus generating loosely connected and poorly discriminative representation.

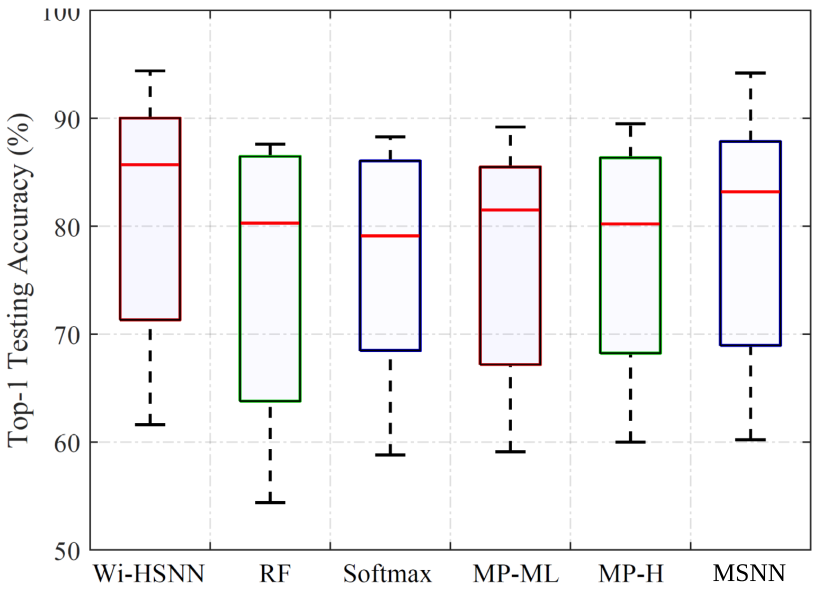


Figure 4.7 – Comparison of Wi-HSNN and other classification and encoding methods when harnessing concatenated (ResNet, DenseNet and InceptionNet) feature.

4.3.5 Timing Analysis

Table 4.8 lists the training time (total training time) and testing time (inference time per frame) of Wi-HSNN and other representation learning techniques [24, 42, 23] on each food dataset. Note that all of the experiments are performed on a CPU platform. From Table 4.8, we can find that Wi-HSNN takes more training time than MP-H and MP-ML, as Wi-HSNN requires extra time for pulling back the error term and engaging in iterative training. For example, on the Food-101 dataset, MP-ML only needs 7.3 minutes to fit the model while Wi-HSNN needs 17.1 minutes. The major overhead of training Wi-HSNN dwells in the optimal feature encoding process, and the training time of the iterative representation learning could be significantly diminished when the batch-by-batch scheme is conducted on a GPU platform. On the other hand, Wi-HSNN records an excellent inference speed with 1.2 ms mean average prediction time on a 2.8 GHz $E5-2650$ processor. However, the MP inverse-based frameworks, such as MP-ML [24], MP-H [42], and MP-Sub [23] demand 0.1 ms, 0.2 ms, and 0.6 ms of more mean average processing time than MSNN, respectively.

Table 4.8 – Processing time with concatenated feature : Tr. (m) is the training time in minute, Te. (ms) is the mean average testing time per frame in millisecond.

Datasets	MP-ML [24]		MP-H [42]		MSNN [23]		Wi-HSNN	
	Tr.	Te.	Tr.	Te.	Tr.	Te.	Tr.	Te.
UEC-100	**1.6**	1.4	1.5	1.4	3.1	1.8	2.7	**1.2**
Food-101	**7.3**	1.4	7.5	1.4	20.6	1.9	17.1	**1.2**
Food-251a	18.4	1.3	**17.7**	1.5	43.9	1.7	40.4	**1.2**
UEC-256	3.3	1.3	**2.9**	1.4	10.7	1.8	10.2	**1.3**

4.3.6 Qualitative Analysis

Fig. 4.8 visualize the optimal feature Γ of the testing set of Food-101 and Food-251 in 2-D space by t-SNE. In these datasets, we loaded the network with concatenated transfer learning features which are extracted from ResNet, AlexNet and VGG, respectively. For every dataset, we recorded the following space: the raw concatenated feature (X), network with one iteration (1 RS-node), network with three iterations (3 RS-nodes) and network with proper iterations (8 RS-nodes). Through visual inspection, we can see that with the iteration increases, the data points with the same category are grouped while the space between different label data points is reserved. Therefore, the proposed algorithm performs well in feature transformation and learning.

4.3.7 Limitations of The Proposed Wi-HSNN

It is noted that the proposed method has the following limitations. First, the optimal feature coding is generated based on the traditional MP inverse strategy, which is a one-batch learning technique. It requires the input data to be processed once. Hence, this model is incapable of handling some very big datasets, such as the original Places-365 dataset with more than 1.8 million samples. Second, it is currently incapable of extracting features from raw images. Thus, the current model can be considered as a feature refiner and classifier. In other words, this algorithm could be a powerful tool in the final stage of pattern classification.

4.4 Conclusion

A new RL strategy is proposed for the problem of image classification and camera model identification. The main contributions of the proposed model are: 1) Feature encoded from subnetwork model rather than traditional neural node, 2) Instead of constructing multi-layer network depth-wise, a robust subnetwork-based neural framework is proposed, which pursues building the optimal feature space through the growth of subnetwork nodes, and 3) Unlike most existing RL methods that rely on solving sub-problems to approximate the global optimization, the proposed method enables the researchers to construct the subspace feature, global feature, and classifier parameters jointly.

The experimental results on benchmark datasets from multiple domains prove that this feature coding and representation learning framework provides promising performances with a flexible structure. As for future direction, it is worth investigating an SNN-based algorithm, which achieves a generalized feature space based on both feature properties and class-specific information.

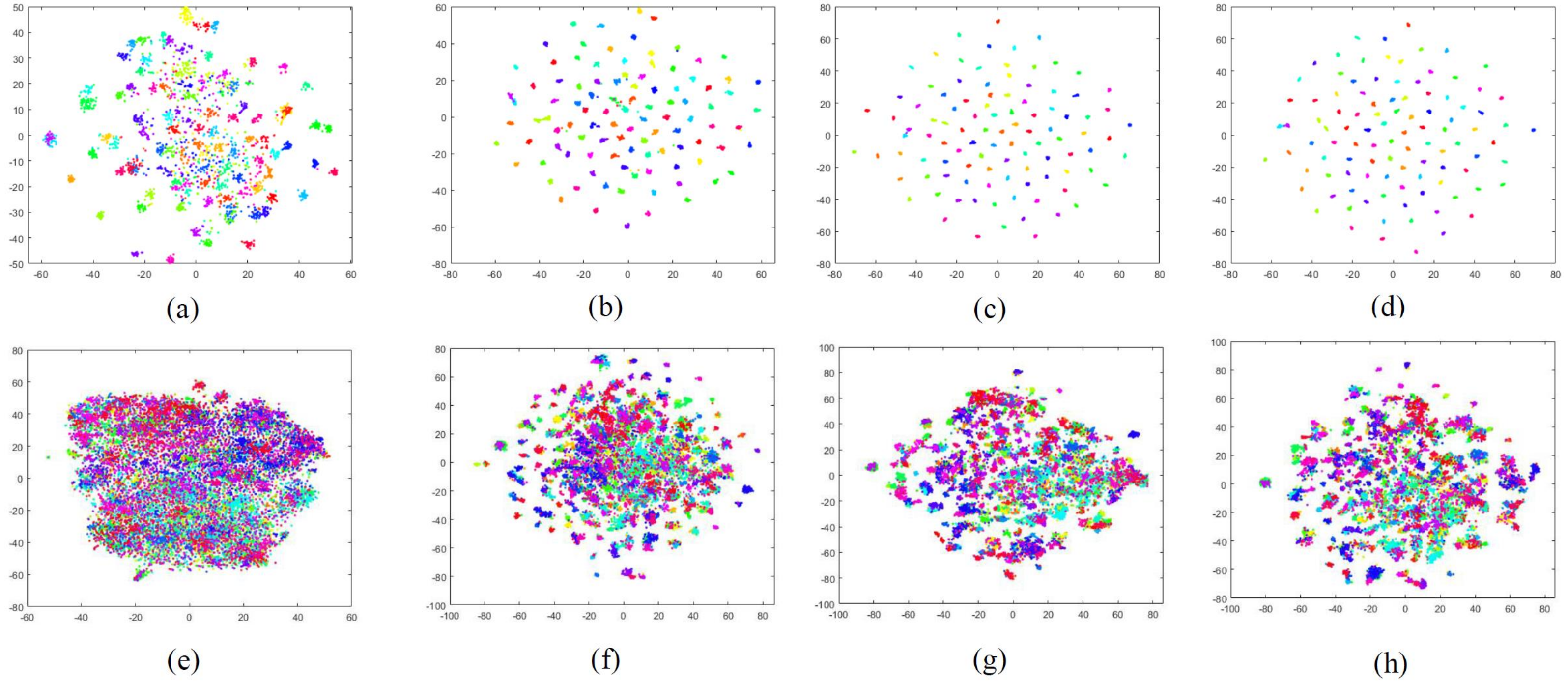


Figure 4.8 – The visualized t-SNE plots of our network on Food-101 and Food-251 datasets, where different color means different category. The t-SNE is plotted with four situations: The raw concatenated feature space $[\mathbf{X}]$, the feature representation with one RS-node $[\Gamma_1]$, the feature representation with two RS-nodes $[\Gamma_2]$, the feature representation with three RS-nodes $[\Gamma_3]$. (a) - (d) t-SNE on Food-101 data set, and (e) - (h) t-SNE on Food-251 dataset.

Chapter 5

Multi-Model Feature Reinforcement Framework using MP Inverse for Big Data Analysis

Fully-connected representation learning (FCRL) is one of the widely used network structures in multi-model image classification frameworks, where the subnetwork-based neural network is a vital branch. However, the subnetwork-based neural network uses the traditional MP inverse as a foundation to learn the data structure, which is a one-batch learning strategy that can be only utilized on small- and median-scale datasets. This paper achieves a robust representation through a proposed batch-by-batch strategy called online-sequential hierarchical subnetwork neural network (OS-HSNN). The novelties of this framework are: It applies an MP inverse-based batch-by-batch learning strategy to handle large-scale datasets so that large datasets such as Places-365 containing 1.8 million images can be processed effectively. The experimental results on multiple domains with a varying number of training samples from $\sim 1K$ to $\sim 2M$ show that the proposed feature reinforcement framework achieves better generalization performance compared with most state-of-the-art FCRL methods.

5.1 Introduction

Recent years have witnessed the feature representation power of deep learning methods, including MLNNs and DCNNs, which have been demonstrated in many computer vision-related applications such as image classification and object recognition. In classical statistics, tasks such as image classification are mainly based on a statistical model $p(x|\lambda)$ describing the possibility of observing x given the parameters of a model λ. However, the high dimensionality and large scale of big data pose a serious problem because the model $p(x|\lambda)$ over high dimensional space of the training data is hard to know and explained clearly with a mathematical equation. Instead, a traditional way is used to process the large patterns of training data by explaining data on a relatively small scale dimension. Traditionally, MLNNs have approached this problem through building various RL algorithms, such as deep autoencoders, multilayer ELM and subnet-based neural network.

In the previous chapter, a novel subnet-based structure named Wi-HSNN was proposed for RL and image classification purposes. Nonetheless, the Wi-HSNN achieves state-of-the-art performance than most of the RL algorithms on various real-world applications, such as camera model identification and food image classification. It cannot efficiently handle large-scale datasets like the original Places-365 and ImageNet datasets. The reason is that it utilizes the traditional MP inverse solution to analytically calculate the optimal weights. Essentially, the MP inverse is a one-batch learning technique where all the training data are processed simultaneously. For example, the Places-365 is composed of 1,803,460 samples, the dimensionality of desired output $\mathbf{T}$ is pretty high, which is $\mathbf{R}^{1,803,460\times 365}$ (requires $>$ 100 GB main memory). It is infeasible to be processed on computers or laptops. It can be only implemented on a workstation. Therefore, a novel MP inverse-based batch-by-batch learning strategy, instead of the traditional one-batch algorithm, is what we urgently need.

In this chapter, a novel batch-by-batch solution for the MP inverse is proposed to efficiently calculate the analytic solution. Batch-by-batch learning is a technique used when the entire dataset is computationally infeasible to be trained at once, and it requires out-of-core algorithms. The optimal parameters of the batch-by-batch implementation are similar to the recursive least-squares algorithm presented in [142]. In particular, the weights of one specific layer are initialized with conventional MP inverse using the first batch of data, then they are updated with the addition of sequential batches. By doing so, we can utilize the proposed subnet-based model in any computer and environment.

5.2 The Proposed Algorithm

The network structure of OS-HSNN is the same as that of the Wi-HSNN (cf. Fig. 4.2). The notations utilized in this chapter are elaborated in Table 5.1.

Table 5.1 – Notations to be used in this chapter

Notation	Meaning
$\boldsymbol{\beta}$	In this chapter, $\boldsymbol{\beta}$ represents the weight $\mathbf{W}_L^v$ for simplification
Γ	the feature learnt from the exit layer (global-level representation)
$\Gamma(x_p)$	the p-th batch of Γ
H_i	feature extracted from the i-th subnet in entrance layer (S-node)
$\mathrm{H}_i(x_p)$	the p-th batch of H_i
Ψ_i	feature extracted from the i-th RS-node in feature refinement layer
$\Psi_i(x_p)$	the p-th batch of Ψ_i
$\mathbf{b}_i^f$	bias of SNN feature layer, which is generated randomly
$\mathbf{b}_i^r$	bias of feature refinement layer, which is generated randomly
D	the dimension of S-node
d	the dimension of RS-node
$\mathbf{E}_i$	the error matrix
$\mathbf{E}_i(x_p)$	the p-th batch of $\mathbf{E}_i$
I	the identity matrix
L	the total number of iterations
M	the total number of sub-batches
m	the dimension of output layer (labels)
N	the number of training samples
n	the dimension of input
$\mathbf{P}_i$	the error feedback data in feature refinement layer
$\mathbf{P}_i(x_p)$	the p-th batch of $\mathbf{P}_i$
$\mathbf{T}$	the expected output
$\mathbf{T}(\mathbf{x}_p)$	the p-th batch of $\mathbf{T}$
$\mathbf{W}_i^f$	weight of the i-th subnetwork in entrance layer (S-node)
$\mathbf{W}_i^r$	weight of the i-th subnetwork in feature refinement layer (RS-node)
$\mathbf{W}_i^v$	parameters of the output layer
$\mathbf{X}$	the input feature
$\mathbf{x}_p$	the p-th batch of $\mathbf{X}$

5.2.1 The Proposed OS-HSNN

The objective function of Wi-HSNN can be summarized as Eq. (5.1).

$$\begin{aligned} &\text{minimize} \;\; J = \frac{1}{2}||\mathbf{T} - f(\mathrm{H}_i, \mathbf{W}_i^r, \mathbf{b}_i^r) \cdot \mathbf{W}_L^v||^2, \\ &f(\mathrm{H}_i, \mathbf{W}_i^r, \mathbf{b}_i^r) = \Gamma = \sum_{i=1}^{L} g(\mathrm{H}_i \cdot \mathbf{W}_i^r + \mathbf{b}_i^r), \end{aligned} \tag{5.1}$$

where $\mathbf{T} \in \mathcal{R}^{N\times m}$ is the expected output, $\mathbf{X} \in \mathcal{R}^{N\times n}$ is the input matrix, L is the total number of RS-node nodes, $\mathbf{W}_i^r$ and $\mathbf{W}_i^v$ are parameters for refinement layer and output layer respectively. H_i is the entrance layer feature matrix generated based on ($\mathbf{W}_i^f$, $\mathbf{b}_i^f$) and the input feature $\mathbf{X}$. $\Psi_i = g(\mathrm{H}_i\mathbf{W}_i^r + \mathbf{b}_i^r)$ is the features extracted from the i-th RS-node, and $\sum_{i=1}^{l} g(\mathrm{H}_i\mathbf{W}_i^r + \mathbf{b}_i^r)$ is the global-level representations.

For simplification, in this chapter, $\boldsymbol{\beta}$ is used to represent the weight $\mathbf{W}_L^v$. The traditional MP inverse solution of $\boldsymbol{\beta}$, i.e., $\mathbf{W}_L^v$, in Eq. (5.1) is calculated as:

$$\boldsymbol{\beta} = (\frac{I}{C} + \Gamma^T\Gamma)^{-1}\Gamma^T\mathbf{T} \tag{5.2}$$

However, the MP inverse is a one-batch learning technique where all the training data are processed simultaneously. This implementation is infeasible when the dataset is too large to be loaded once. Suppose the input feature and the expected output $\mathbf{X}$ and $\mathbf{T}$ are split into M pieces, i.e., $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_M\}$ and $\mathbf{T} = \{\mathbf{T}(\mathbf{x}_1), \mathbf{T}(\mathbf{x}_2), \cdots, \mathbf{T}(\mathbf{x}_M)\}$, the following batch-by-batch learning strategy for MP inverse is proposed.

Theorem 5.1. *Given several chunk of hidden features $\{\Gamma(\boldsymbol{x}_p),\ \boldsymbol{T}(\boldsymbol{x}_p)\}_{p=1}^{M}$, where $\Gamma(\boldsymbol{x}_p)$ is the loaded input data and $\boldsymbol{T}(\boldsymbol{x}_p)$ is the desired output. Assume that the data $\Gamma(\boldsymbol{x}_p)$ is loaded sequentially, $\boldsymbol{\beta}_p = K_p\boldsymbol{\beta}_{p-1} + R_p^{-1}\Gamma(\boldsymbol{x}_p)^T\boldsymbol{T}(\boldsymbol{x}_p)$ is considered as the batch-by-batch solution of Moore-Penrose inverse with K_p and R_p are computed* *by (5.3).*

$$
\begin{aligned}
M_p &= R_{p-1}^{-1}\Gamma(\boldsymbol{x}_p)^T\left(\Gamma(\boldsymbol{x}_p)R_{p-1}^{-1}\Gamma(\boldsymbol{x}_p)^T + I\right)^{-1}\Gamma(\boldsymbol{x}_p) \\
K_p &= \begin{cases} 0, & p = 1 \\ I - M_p, & 2 \le p \le M \end{cases} \\
R_p^{-1} &= \begin{cases} \left[\frac{I}{C} + \Gamma(\boldsymbol{x}_1)^T\Gamma(\boldsymbol{x}_1)\right]^{-1}, & p = 1 \\ K_p R_{p-1}^{-1}, & 2 \le p \le M \end{cases}
\end{aligned}
\tag{5.3}
$$

where $\boldsymbol{\beta}_p$ is the optimal weights which is calculated based on the first p batches of data. I is the identity matrix.

5.2.2 Algorithmic Summary

The training algorithm for the OS-HSNN is present in Algorithm 5.1. The proposed method contains three learning procedures: Procedure I - initialization, Procedure II - error backpropagation, and Procedure III - batch-by-batch learning. The first two procedures are almost the same as per Wi-HSNN. In particular, Procedure I is employed as the initialization of OS-HSNN. The weights $\mathbf{W}_1^f$ and $\mathbf{W}_1^r$ are assigned randomly from a zero-mean Gaussian distribution, where the standard deviation is equal to 0.01. The weight $\mathbf{W}_1^v$ is generated via the proposed batch-by-batch strategy; Procedure II aims to generate the error term $\mathbf{E}_i$ and desired pulled back error term $\mathbf{P}_i$, which could guide the following network construction, i.e., the new feature representation Ψ_{i+1}; Procedure III is the proposed batch-by-batch solution of MP inverse. the weight of one layer is first calculated with the first chunk of data via traditional one-batch MP inverse, while the appropriate parameters are gradually updated with the consideration of new data chunks. Thus, the optimal weights are learnt sequentially.

The training process for this algorithm can be easily split into two continuous phases: Network initialization phase (Line 1-2 in Algorithm 5.1) and iterative feature encoding phase (Line 3-12 in Algorithm 5.1). From Algorithm 5.1, we conclude that in each iteration, the newly added subnet nodes (S-node and RS-node) mainly try to

Algorithm 5.1 The proposed OS-HSNN

Inputs: The concatenated super-state feature vector $(\mathbf{X}, \mathbf{T})$
Outputs: Class label $\mathbf{Y}$

```
Procedure I: $\mathbf{W}_1^f$, $\mathbf{W}_1^r$, $\mathbf{W}_1^v$ and $\Gamma \leftarrow$ *Initialization*
Procedure II: $\mathbf{E}_1$ and $\mathbf{P}_1 \leftarrow$ *Error Backpropagation* $(\mathbf{T}, \Gamma, \mathbf{W}_1^v)$
for each iteration $i$ do
    $\mathbf{W}_i^f \leftarrow$ Assign randomly % The entrance layer weight.
    $\mathrm{H}_i = g(\mathbf{W}_i^f \cdot \mathbf{X} + \mathbf{b}_1^f)$ % The entrance layer feature.
    Procedure III: $\mathbf{W}_i^r \leftarrow$ *Batch-by-batch Learning* $(\mathrm{H}_i, \mathbf{P}_{i-1})$ % The refinement layer weight.
    $\Psi_i = g(\mathbf{W}_i^r \cdot \mathrm{H}_i + \mathbf{b}_i^r)$ % The refinement layer feature.
    $\Gamma = \sum_j^i \Psi_j$ % Update the global-level representation.
    Procedure III: $\mathbf{W}_i^v \leftarrow$ *Batch-by-batch Learning* $(\Gamma, \mathbf{T})$
    Procedure II: $\mathbf{E}_i$ and $\mathbf{P}_i \leftarrow$ *Error Backpropagation* $(\mathbf{T}, \Gamma, \mathbf{W}_i^v)$
end for
$\mathbf{Y} = \Gamma \cdot \mathbf{W}_i^v$    % $\Gamma$ is the global-level representation, $\mathbf{Y}$ is the label.
```

Procedure I: *Initialization*
Inputs: NA
Outputs: $\mathbf{W}_1^f$, $\mathbf{W}_1^r$, $\mathbf{W}_1^v$ and Γ

```
$\mathbf{W}_1^f \leftarrow$ Assign randomly    % Assign randomly with $\mu$=0, $\sigma$=0.01 of Gaussian distribution
$\mathrm{H}_1 = g(\mathbf{W}_1^f \cdot [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_M] + \mathbf{b}_1^f)$
$\mathbf{W}_1^r \leftarrow$ Assign randomly    % Assign randomly with $\mu$=0, $\sigma$=0.01 of Gaussian distribution
$\Psi_1 = g(\mathbf{W}_1^r \cdot \mathrm{H}_1 + \mathbf{b}_1^r) = g(\mathbf{W}_1^r \cdot [\mathrm{H}_1(\mathbf{x}_1), \mathrm{H}_1(\mathbf{x}_2), \cdots, \mathrm{H}_1(\mathbf{x}_M)] + \mathbf{b}_1^r)$
$\Gamma = \Psi_1$
Procedure III: $\mathbf{W}_1^v \leftarrow$ *Batch-by-batch Learning* $(\Gamma, \mathbf{T})$
return $\mathbf{W}_1^f$, $\mathbf{W}_1^r$ and $\mathbf{W}_1^v$
```

Procedure II: *Error Backpropagation* $(\mathbf{T}, \Gamma, \mathbf{W}_v)$
Inputs: Γ=[$\Gamma(\mathbf{x}_1)$, $\Gamma(\mathbf{x}_2)$, $\cdots$, $\Gamma(\mathbf{x}_M)$], $\mathbf{T}$=[$\mathbf{T}(\mathbf{x}_1)$, $\mathbf{T}(\mathbf{x}_2)$, $\cdots$, $\mathbf{T}(\mathbf{x}_M)$], $\mathbf{W}_v$
Outputs: $\mathbf{E}_i$, $\mathbf{P}_i$

```
for each sub-batch $p$ do
    $\mathbf{E}_i(\mathbf{x}_p)$=$\mathbf{T}_p - \Gamma(x_p) \cdot \mathbf{W}_v$
end for
for each sub-batch $p$ do
    $\mathbf{P}_i(x_p) = g^{-1}\left(\mathbf{E}_i(x_p) \cdot (\frac{I}{C} + \mathbf{W}_v^T \cdot \mathbf{W}_v)^{-1} \cdot \mathbf{W}_v^T\right)$
end for
$\mathbf{E}_i = [\mathbf{E}_i(\mathbf{x}_1), \cdots, \mathbf{E}_i(\mathbf{x}_M)]$, $\mathbf{P}_i = [\mathbf{P}_i(\mathbf{x}_1), \cdots, \mathbf{P}_i(\mathbf{x}_M)]$
return $\mathbf{E}_i$ and $\mathbf{P}_i$;
```

Procedure III: *Batch-by-batch Learning* $(\Gamma, \mathbf{T})$
Inputs: Γ=[$\Gamma(\mathbf{x}_1), \cdots, \Gamma(\mathbf{x}_M)$], $\mathbf{T}$=[$\mathbf{T}(\mathbf{x}_1), \cdots, \mathbf{T}(\mathbf{x}_M)$]
Outputs: $\boldsymbol{\beta}_p$

```
$R_1^{-1} = [\frac{I}{C} + \Gamma(\mathbf{x}_1)^T \Gamma(\mathbf{x}_1)]^{-1}$
$\boldsymbol{\beta}_1 = R_1^{-1} \Gamma(\mathbf{x}_1)^T \mathbf{T}(\mathbf{x}_1)$
for each sub-batch $p$ do
    $K_p = I - R_{p-1}^{-1} \Gamma(\mathbf{x}_p)^T (\Gamma(\mathbf{x}_p) R_{p-1}^{-1} \Gamma(\mathbf{x}_p)^T + I)^{-1} \Gamma(\mathbf{x}_p)$
    $R_p^{-1} = K_p R_{p-1}^{-1}$
    $\boldsymbol{\beta}_p = K_p \boldsymbol{\beta}_{p-1} + R_p^{-1} \Gamma(\mathbf{x}_p)^T \mathbf{T}(\mathbf{x}_M) \leftarrow$ Updated the weight
end for
return $\boldsymbol{\beta}_p$;
```

help the model find the knowledge from the dataset which has not been learnt. Both the phases are elaborated as follows:

Network initialization phase: The entrance layer weights, $\mathbf{W}_1^f$, refinement layer weight $\mathbf{W}_1^r$ and exit layer weight $\mathbf{W}_1^v$ are obtained via Procedure I (Line 1). The error $\mathbf{E}_1$ and feedback error $\mathbf{P}_1$ are obtained through Procedure II (Line 2).

Iterative feature encoding phase: This part may be iterated several times to learn and approximate the rest feature refinement spaces. In each iteration, a new subnetwork in the entrance layer and refinement layer is added, respectively. Firstly, the parameters for the entrance layer are generated randomly (Line 4-5), whereas the weight in the refinement layer is calculated according to both the pulled back error and the output of the newly added entrance layer subnetwork. The new subnets are used to offset the current error (Line 6). Furthermore, after receiving the current optimal feature representations, users could calculate weight $\mathbf{W}_i^v$ through batch-by-batch strategy to map the coding to label space (Line 7-9). Finally, the error term is calculated to provide the next learning direction (Line 10).

5.2.3 Proof of The Proposed OS-HSNN

In this subsection, the batch-by-batch learning strategy for OS-HSNN is first proved. Based on Theorem 5.1, in Theorem 5.2 a proof of its convergence is given.

Proof. Suppose we have $\Gamma_{(p-1)}, \Gamma_{(p)}, \mathbf{T}_{(p-1)}, \mathbf{T}_{(p)}$, which are defined as equation (5.4).

$$\Gamma_{(p)} = \begin{bmatrix} \Gamma(\mathbf{x}_1) \\ \Gamma(\mathbf{x}_2) \\ \vdots \\ \Gamma(\mathbf{x}_p) \end{bmatrix} = \begin{bmatrix} \Gamma_{(p-1)} \\ \Gamma(\mathbf{x}_p) \end{bmatrix}, \quad \mathbf{T}_{(p)} = \begin{bmatrix} \mathbf{T}(\mathbf{x}_1) \\ \mathbf{T}(\mathbf{x}_2) \\ \vdots \\ \mathbf{T}(\mathbf{x}_p) \end{bmatrix} = \begin{bmatrix} \mathbf{T}_{(p-1)} \\ \mathbf{T}(\mathbf{x}_p) \end{bmatrix}. \tag{5.4}$$

Initially, the MP inverse result of $\boldsymbol{\beta}_{p-1}$ with $p-1$ sets of data are settled by the following unconstrained optimization solution [26].

$$\boldsymbol{\beta}_{p-1} = \left[\frac{I}{C} + \Gamma_{(p-1)}^T \Gamma_{(p-1)}\right]^{-1} \Gamma_{(p-1)}^T \mathbf{T}_{(p-1)}, \tag{5.5}$$

where C is the regularization term. Wherein, $\left(\frac{I}{C} + \Gamma^T_{(p-1)}\Gamma_{(p-1)}\right) \rightarrow R_{p-1}$ and $\left(\frac{I}{C} + \Gamma^T_{(p)}\Gamma_{(p)}\right) \rightarrow R_p$. Thus, $\boldsymbol{\beta}_p$ and $\boldsymbol{\beta}_{p-1}$ can be rewritten as:

$$\boldsymbol{\beta}_{p-1} = R^{-1}_{p-1}\Gamma^T_{(p-1)}\mathbf{T}_{(p-1)}, \text{ and } \boldsymbol{\beta}_p = R^{-1}_p\Gamma^T_{(p)}\mathbf{T}_{(p)}. \tag{5.6}$$

Based on (5.4):

$$\begin{aligned} R_p &= \frac{I}{C} + \begin{bmatrix}\Gamma_{(p-1)} \\ \Gamma(\mathbf{x}_p)\end{bmatrix}^T \begin{bmatrix}\Gamma_{(p-1)} \\ \Gamma(\mathbf{x}_p)\end{bmatrix} \\ &= \frac{I}{C} + \Gamma^T_{(p-1)}\Gamma_{(p-1)} + \Gamma(\mathbf{x}_p)^T\Gamma(\mathbf{x}_p). \end{aligned} \tag{5.7}$$

Then, update R_p with R_{p-1}

$$R_p = R_{p-1} + \Gamma(\mathbf{x}_p)^T\Gamma(\mathbf{x}_p). \tag{5.8}$$

Using Sherman-Morrison-Woodbury (SMW) formula [143], the inverse of R_p can be attained:

$$\begin{aligned} R^{-1}_p &= (R_{p-1} + \Gamma(\mathbf{x}_p)^T\Gamma(\mathbf{x}_p))^{-1} \\ &= R^{-1}_{p-1} - R^{-1}_{p-1}\Gamma(\mathbf{x}_p)^T(I + \Gamma(\mathbf{x}_p)R^{-1}_{p-1}\Gamma(\mathbf{x}_p)^T)^{-1}\Gamma(\mathbf{x}_p)R^{-1}_{p-1}. \end{aligned} \tag{5.9}$$

Subsequently, the Eq. (5.6) can be rewritten as

$$\begin{aligned} \boldsymbol{\beta}_p =& \left(R^{-1}_{p-1} - R^{-1}_{p-1}\Gamma(\mathbf{x}_p)^T(I + \Gamma(\mathbf{x}_p)R^{-1}_{p-1}\Gamma(\mathbf{x}_p)^T)^{-1} \cdot \Gamma(\mathbf{x}_p)R^{-1}_{p-1}\right) \cdot \\ & \begin{bmatrix}\Gamma^T_{(p-1)} & \Gamma(\mathbf{x}_p)^T\end{bmatrix} \begin{bmatrix}\mathbf{T}_{(p-1)} \\ \mathbf{T}(\mathbf{x}_p)\end{bmatrix} \\ =& \left(I - R^{-1}_{(p-1)}\Gamma(\mathbf{x}_p)^T(\Gamma(\mathbf{x}_p)R^{-1}_{p-1}\Gamma(\mathbf{x}_p)^T + I)^{-1} \cdot \Gamma(\mathbf{x}_p)\right) \cdot \\ & R^{-1}_{p-1} \begin{bmatrix}\Gamma^T_{(p-1)} & \Gamma(\mathbf{x}_p)^T\end{bmatrix} \begin{bmatrix}\mathbf{T}_{(p-1)} \\ \mathbf{T}(\mathbf{x}_p)\end{bmatrix}. \end{aligned} \tag{5.10}$$

Furthermore, for simplicity, we denote K_p as:

$$K_p = I - R_{p-1}^{-1}\Gamma(\mathbf{x}_p)^T(\Gamma(\mathbf{x}_p)R_{p-1}^{-1}\Gamma(\mathbf{x}_p)^T + I)^{-1}\Gamma(\mathbf{x}_p). \tag{5.11}$$

Substitute K_p into Eq. (5.9), R_p can be rewritten as.

$$R_p^{-1} = K_p R_{p-1}^{-1}. \tag{5.12}$$

Meanwhile, substitute K_p into Eq. (5.10), the weight $\boldsymbol{\beta}_p$ can be simplified to the following equation:

$$\begin{aligned}\boldsymbol{\beta}_p &= K_p R_{p-1}^{-1}\begin{bmatrix}\Gamma_{(p-1)}^T & \Gamma(\mathbf{x}_p)^T\end{bmatrix}\begin{bmatrix}\mathbf{T}_{(p-1)} \\ \mathbf{T}(\mathbf{x}_p)\end{bmatrix} \\ &= K_p R_{p-1}^{-1}(\Gamma_{(p-1)}^T\mathbf{T}_{(p-1)} + \Gamma(\mathbf{x}_p)^T\mathbf{T}(\mathbf{x}_p)) \\ &= K_p R_{p-1}^{-1}\Gamma_{(p-1)}^T\mathbf{T}_{(p-1)} + K_p R_{p-1}^{-1}\Gamma(\mathbf{x}_p)^T\mathbf{T}(\mathbf{x}_p).\end{aligned} \tag{5.13}$$

So, the updated weight $\boldsymbol{\beta}_p$ can be written with Eq. (5.6) in the case of only new training data is available:

$$\boldsymbol{\beta}_p = K_p\boldsymbol{\beta}_{p-1} + R_p^{-1}\Gamma(\mathbf{x}_p)^T\mathbf{T}(\mathbf{x}_p). \tag{5.14}$$

□

Now, based on Theorem 5.1, we give the second theorem. Theorem 5.2 proofs that if the network continue adding the subnetwork, the output error sequence $\|\mathbf{E}_i\|$ will be decreasing and finally get stable.

Theorem 5.2. *Suppose the error sequence $\{\boldsymbol{E}_i\}_{i=1}^{\infty}$ is generated based on Algorithm 5.1. As $i \to \infty$, the ℓ_2 norm of error sequence L_i^2 vanishes, thus the sequence L_i^2 converge and bound below by zero.*

Proof. In this chapter, the subspace feature combination is processed with a specific operator, $f(\cdot)$. Suppose Γ_i stands for the global-level feature representation with i-th iteration. The data representation of $(i+1)-th$ subnetwork nodes Ψ_{i+1} can be expressed as:

$$\Gamma_{i+1} = f\left(\begin{bmatrix}\Gamma_i \\ \Psi_{i+1}\end{bmatrix}\right) = \Gamma_i + \Psi_{i+1}, \tag{5.15}$$

where Γ_i is the feature representation with i subnetwork nodes, while Ψ_{i+1} is the feature of the $(i+1) - th$ newly added subnetwork. Still, we use $\boldsymbol{\beta}$ instead of $\mathbf{W}_v^L$ in this proof for simplification (Note: $\boldsymbol{\beta}_i$ refers to the optimal weight calculated from the first i batches of data).

$$\begin{aligned} L_{i+1}^2 &= \left\| \mathbf{T} - f\left(\begin{bmatrix}\Gamma_i \\ \Psi_{i+1}\end{bmatrix}\right) \cdot \boldsymbol{\beta}_{i+1} \right\| \\ &= \left\| \mathbf{T} - f\left(\begin{bmatrix}\Gamma_i \\ \Psi_{i+1}\end{bmatrix}\right) \cdot (\boldsymbol{\beta}_i + \Delta\boldsymbol{\beta}) \right\| \\ &\leq \left\| \mathbf{T} - f\left(\begin{bmatrix}\Gamma_i \\ \Psi_{i+1}\end{bmatrix}\right) \cdot \boldsymbol{\beta}_i \right\|, \end{aligned} \tag{5.16}$$

where $\boldsymbol{\beta}_{i+1} = \boldsymbol{\beta}_i + \Delta\boldsymbol{\beta}$ is the optimal output weight. In addition, we mentioned that the $(n+1)$-th subspace feature aimed to learn the pulled back feature matrix. As for feature reinforcement layer, the feature Γ_{i+1} attempts to offset pulled matrix $\mathbf{P}$, while the output weight $\boldsymbol{\beta}_{i+1}$ focuses on fitting output error $\mathbf{E}_i$. However, the residual error σ still exists. Then, we have

$$\mathbf{E}_i = \Psi_{i+1} \cdot \boldsymbol{\beta}_i + \sigma. \tag{5.17}$$

Substitute Eq. (5.15) Eq. (5.17) into Eq. (5.16) and simplify the equation, we can get

$$\begin{aligned} L_{i+1}^2 &\leq \left\| \mathbf{t} - f\left(\begin{bmatrix}\Gamma_i \\ \Psi_{i+1}\end{bmatrix}\right) \cdot \boldsymbol{\beta}_i \right\| \\ &= \left\| \mathbf{T} - \Gamma_i \boldsymbol{\beta}_i + \Psi_{i+1} \boldsymbol{\beta}_i \right\| \\ &= \left\| \mathbf{E}_i - \Psi_{i+1} \boldsymbol{\beta}_i \right\| = \left\| \sigma \right\|. \end{aligned} \tag{5.18}$$

Because $\|\sigma\| \leq \|\mathbf{E}_i\|$, we have the following judgement.

$$L_i^2 = \left\| \mathbf{T} - f\left(\begin{bmatrix} \Psi_1 \\ \vdots \\ \Psi_i \end{bmatrix}\right) \cdot \boldsymbol{\beta}_i \right\| \geq \left\| \mathbf{T} - f\left(\begin{bmatrix} \Gamma_i \\ \Psi_i \end{bmatrix}\right) \cdot \boldsymbol{\beta}_i \right\| = L_{i+1}^2. \tag{5.19}$$

Thus, we can have $\|\mathbf{E}_\infty\| \leq \cdots \leq \|\mathbf{E}_{i+1}\| \leq \|\mathbf{E}_i\| \leq \cdots \leq \|\mathbf{E}_1\|$. Further we have $\lim_{i\to\infty} \|\mathbf{E}_i\| = 0$. So the sequence $\|\mathbf{E}_i\|$ is decreasing and bounded below by zero.

□

5.3 Experimental results

In the following part, the batch-by-batch strategy for OS-HSNN is validated on different application domains, including image classification and ship-target detection.

5.3.1 Experimental Setup

5.3.1.1 The Environment

The experiments performed in this chapter were conducted in MATLAB 2018b on a computer with a 256 GB memory and a 2.8 GHz $E5-2650$ processor. The feature extraction using DCNNs is carried out on a single NVIDIA 1080Ti GPU.

5.3.1.2 The Dataset

We applied the proposed algorithm on 8 commonly used image classification datasets, which are described in Table 5.2. Based on the number of training samples, the adopted datasets can be divided into two categories: small-scale datasets (Scene-15, Caltech-101/256) and large-scale datasets (Food-251, Places-365-1/2/3, Places-365). The small-scale datasets are generally a small number of samples ($< 100K$), whereas large-scale datasets have marginally large samples ($> 100K$). It is worth noting that despite the number of training samples, the experiments conducted in this chapter are the results of the proposed model using the batch-by-batch strategy. The details of the image classification datasets are described as follows.

Table 5.2 – Summary of the image classification datasets

Mode	Datasets	Classes	Training samples	Testing samples	Neurons in S-node	Neurons in RS-node	Iterations
Batch-by-batch	Scene-15 [144]	15	1,500	2,985	500	300	5
	Caltech-101 [145]	102	3,060	6,084	500	300	5
	Caltech-256 [146]	257	7,710	22,898	500	300	5
	Food-251 [131]	251	118,475	11,994	1000	500	5
	Places-365-1 [147]	365	146,000	36,500	1000	500	5
	Places-365-2 [147]	365	292,000	73,000	1000	500	8
	Places-365-3 [147]	365	438,000	109,500	1000	500	8
	Places-365 [147]	365	1,803,460	365,000	1000	500	8

Small-scale datasets: Following the commonly used training settings, in this chapter, we take 100 (Scene-15) and 30 (Caltech-101/256) images per category for training; the rest of the images are for testing.

Large-scale datasets: As far as we know, Places-365 and ImageNet could be the largest datasets in the image classification area. Therefore, the Places-365 set was applied to evaluate the proposed representation learning algorithm with a batch-by-batch learning fashion. Precisely, (i) we randomly selected 500, 1,000 and 1,500 samples per class from the training set to generate Places-365-1, Places-365-2, and Places-365-3 (each set 80% samples for training, the rest for testing), and (ii) use the original Places-365 set (training set for training, validation set for testing) to validate the algorithm. In addition, the Food-251 dataset was the latest food classification dataset, created in 2019. All 118,475 training images were used for training and 11,994 validation images for testing.

Furthermore, to fully validate the effectiveness of the proposed OS-HSNN, one more experiment was conducted on the ship-target detection task. In this chapter, the HF-radar dataset [148] is utilized for comparison. It consists of 200 RD images equally divided into training and testing sets. Also, there are 15,192 image patches with each patch in the size of 20×20, having either one of the following class labels: vessel-target, sea clutter, ionospheric clutter, and others.

5.3.1.3 The Rival Methods

The proposed feature representation algorithm is compared with the following state-of-the-art RL methods, which can be basically divided into two parts:

BP-based RL methods with fine-tuned DCNN features: Several state-of-the-art BP-based RL methods, including the weight-decay regularization-based autoencoder (WD-AE) [149], denoise autoencoder with Gaussian making noise (DAE^G) [101], sparse autoencoder (SAE) [150] were tested for comparison. After the optimal encoding was learnt by each representation learning methods, the output label for each sample was obtained through a softmax classifier.

MP inverse-based RL methods with fine-tuned DCNN features: As the proposed algorithm is a MP inverse-based feature RL algorithm, several representation learning networks, i.e., MP inverse-based multilayer network (MP-ML) [42], hierarchical network (MP-H) [24], and the hierarchical subnet-based neural network (HSNN) [23] were further compared.

Note that the features loaded in our network were first carried out with min-max normalization, and the quantity analysis stated in this chapter is Top-1% testing accuracy. All the experiments in this chapter are the mean average of minimum three experiments.

5.3.1.4 Configurations of The Rival Methods

For the BP-based RL algorithms, the total number of training epochs and the mini-batch size were both set as 100. The initial learning rate was defined as 0.01, and was reduced to one tenth every 10 epochs. The input corruption rate of DAE^G was 0.5. For H-ELM and M-ELM, the optimal regularization term C and number of hidden neurons on each dataset were searched within the grid $\{500,\ 1,000,\ 2,000\} \times \{10^{-4}, 10^{-2},\ 10^0,\ 10^2\ 10^4\}$. As for the HSNN, the number of subnetwork in it was set to 8, each subnetwork contains 1000 hidden neurons, and the optimal regularization term was optimized within the grid $\{10^{-4}, 10^{-2},\ 10^0,\ 10^2\ 10^4\}$.

5.3.1.5 Configurations of The Input Features

Different from the experiments listed in Chapter 4 that use the “P”-DCNN (**pre-trained model**) and “T”-DCNN (**3 training epochs**) features to validate the proposed algorithm, in this chapter, we utilize the fine-tuned DCNN features, i.e., “FT”-DCNN (**10 training epochs**) features for validation.

“FT”-DCNN features - In this chapter, the high-level features that are extracted from the top layer of DCNN are loaded as the raw feature. Note that the “FT”-DCNN features are the data extracted from a particular layer from a DCNN with **10 training epochs**. The details for DCNN features used in this study are as follows: For VGG16, we adopt the features from the fully-connected layer (”$fc7$”), which is a 4096-dimensional feature vector. For Inception-v3, we used the 2048-

dimensional vector from the average pooling layer ("*avg_pool*"). As for ResNet, the 2048-dimensional feature was extracted from the "$fc1000$" layer. The prefix "FT-" has been used to denote the features which belong to this category. For example, FT-AlexNet, FT-GoogleNet, etc.

5.3.2 Model Settings

In this chapter, the proposed framework adds one decision - the choice of training mode (one-batch fashion or batch-by-batch strategy) - to the training process of OS-HSNN. Several indices, including total training time, top-1 testing accuracy and peak memory usage (PMU) in training are used to empirically evaluate the performance of the proposed method in different training modes. For a fair comparison, in this experiment, we provided the same initialization weights $\mathbf{W}_f^i$ for each S-node and $\mathbf{W}_r^1$ for the first RS-node. Through Table 5.3, one can easily summarize the following conclusions: (i) When handling the same input features, the proposed two strategies provide almost the same testing performance. (ii) Compared to the one-batch strategy, the batch-by-batch training mode significantly reduces the PMU of the training subnetwork model, only needs 10% to 20% more processing time. The reason for the same testing performance is that: Authors in [28] have already verified that Eq. (5.2) is the optimal least-squares solution when the rank of feature matrix equals the number of hidden neurons. From the derivation of the proposed method, i.e., Eq. (5.4) to Eq. (5.14), it can be seen that the proposed batch-by-batch strategy and original one-batch solution Eq. (5.2) can achieve the same training error and generalization accuracy when the rank of $\Psi(x_1) = d$, which means the number of initialization data should not be less than the neuron number in each subnet d. Thus, the proposed method with batch-by-batch strategy could be conducted and employed in any processing environment because of its flexible memory usage.

Fig. 5.1 shows the generalization performance of the proposed OS-HSNN on one of the largest datasets Places-365 by a plot in terms of Top-1 classification accuracy in percentage. When the new S-node and RS-SNN are added to this network (i.e., a new iteration), the algorithm generally provides better performance than that without the new subnet nodes. Note that the optimization of all the weights in OS-HSNN is

Table 5.3 – Performance comparison of OS-HSNN w/ FT-VGG16 features: Tr.(m) - training time in minutes, Acc.(%) - testing accuracy, PMU(GB) - peak memory usage, and N/batch - Batch-by-batch strategy w/ N number of samples per batch

Datasets	Type	Tr. (m)	Acc. (%)	PMU (GB)
Food-251	One-batch	**14.0**	57.94	11.8
	20K/batch	15.2	57.86	2.0
	10K/batch	15.6	57.82	**1.0**
Places-365-3	One-batch	**106.1**	45.81	42.7
	20K/batch	122.1	45.73	2.0
	10K/batch	126.2	45.79	**1.0**
Places-365	One-batch	**645.1**	47.58	173.5
	20K/batch	699.1	47.60	2.0
	10K/batch	710.4	47.64	**1.0**

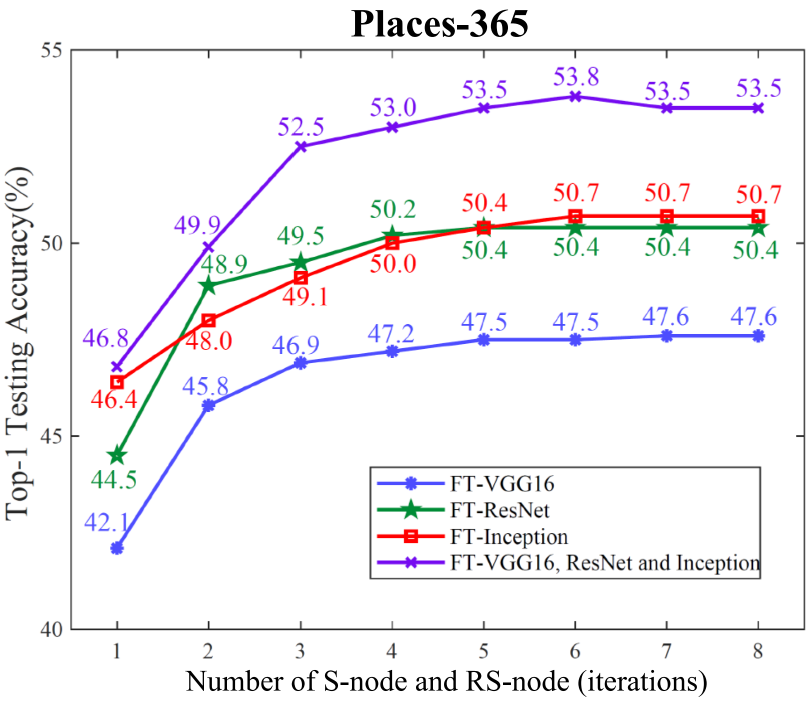


Figure 5.1 – Top-1 testing accuracy of OS-HSNN with fine-tuned features on Places-365 dataset when new S-node and RS-node are added to the framework.

a batch-by-batch manner, which means the peak memory usage of this model only depends on the size of the input batch. Thus, we can conclude that the OS-HSNN is effective when handling large-scale datasets.

5.3.3 Analysis on Image Classification Domain

The comparison results of all datasets including three small-scale and five large-scale datasets are tabulated in Table 5.4. Note that the traditional MP-based representation learning methods (MP-ML, MP-H and HSNN) cannot process large datasets due to the main memory limitation. To allow a fair comparison, these methods are conducted with the batch-by-batch strategy proposed in this study. Thus, large-scale datasets, such as the Places-365 dataset with 1.8 million data samples, can be processed and compared. Consequently, the following conclusions can be drawn: (i) The proposed algorithm with the same feature shows comparable or superior performance than the other state-of-the-art BP-based and MP inverse-based representation learning methods. Overall, the average result of the proposed OS-HSNN with FT-ResNet feature among all datasets is 63.7%, providing 1.8% and 0.2% improvement, compared with the best MP inverse-based (HSNN) and BP-based (SAE) encoding algorithms. (ii) When considering multi-modal learning, the OS-HSNN with concatenated super-state vector (FT-VGG, FT-Inception and FT-ResNet) further boosts the classification accuracy than single-channel "FT" feature. As can be seen in Table. 5.4, our results on Scene-15, Caltech-101, and Caltech-256 were 93.8%, 93.6%, and 84.9%, respectively; 3.1%, 1.9%, and 4.0% better than the results with FT-Inception feature. Therefore, the profits for the multi-model learning strategy were verified.

Table 5.4 – Top-1 testing accuracy (%) for representation learning tasks. Values in red and blue are the best results w/ FT-ResNet and FT-Inception among other methods respectively. Our best results are underlined.

Methods	Scene-15	Caltech-101	Caltech-256	Food-251	Places-365-1	Places-365-2	Places-365-3	Places-365	Average
BP-based representation learning methods with fine-tuned DCNN features									
FT-ResNet, WD-AE [149]	90.4	91.3	76.8	57.7	42.6	44.7	44.4	45.8	61.8
FT-ResNet, DAE^G [101]	89.5	90.7	78.9	57.5	42.1	44.2	45.4	46.0	61.8
FT-ResNet, SAE [150]	89.7	91.0	78.7	57.4	42.8	44.1	45.1	46.5	61.9
FT-Inception, WD-AE [149]	90.3	91.3	77.7	60.5	44.6	45.9	46.3	47.5	63.0
FT-Inception, DAE^G [101]	89.9	91.4	77.9	60.6	44.2	46.2	46.5	47.8	63.1
FT-Inception, SAE [150]	89.6	91.5	80.4	61.7	45.0	46.4	47.1	48.5	63.8
MP inverse-based RL methods with fine-tuned DCNN features									
FT-ResNet, MP-ML [42]	88.2	91.4	79.4	60.1	43.5	44.9	45.8	46.4	62.5
FT-ResNet, MP-H [24]	87.9	91.2	79.6	60.6	42.2	44.7	46.3	46.9	62.4
FT-ResNet, MSNN [23]	89.5	91.5	80.5	59.5	44.5	46.5	46.4	49.6	63.5
FT-Inception, MP-ML [42]	90.3	91.2	80.0	59.9	40.4	44.2	45.1	48.2	62.4
FT-Inception, MP-H [24]	90.0	90.7	80.1	59.4	40.9	44.5	45.2	48.5	62.4
FT-Inception, MSNN [23]	90.0	91.4	80.6	61.8	45.9	47.4	48.2	49.5	64.4
The Proposed OS-HSNN with fine-tuned DCNN features									
FT-VGG16	90.0	91.6	75.4	57.9	43.0	44.9	45.9	47.6	62.0
FT-ResNet	90.1	91.8	80.6	60.2	43.9	45.7	46.9	50.4	63.7
FT-Inception-V3	90.7	91.7	80.9	62.3	45.9	47.8	48.6	50.7	64.7
FT-Inception, ResNet	91.6	92.8	84.2	65.6	47.9	49.3	49.4	51.9	66.6
FT-VGG,Inception,ResNet	**93.8**	**93.6**	**84.9**	**66.7**	**49.3**	**50.6**	**51.6**	**53.5**	**68.0**

Furthermore, Fig. 5.2 summarizes the results among all the image classification datasets via box-plots, where the plot is generated based on the best results of our method. The average figure of each method is listed in Table 5.4. When the average testing accuracy is considered, the proposed OS-HSNN has better average accuracy than most of the fine-tuned DCNNs. While the other state-of-the-art representation learning methods have strong power in processing small-scale datasets, they fail to exhibit the same ability in large-scale datasets due to the redundant parameters. For instance, When processing FT-Inception features, the proposed OS-HSNN gains 3.2%, 2.9%, 2.2%, 2.5%, 2.2%, and 1.2% more accuracy than those with WD-AE, DAE^G, SAE, MP-ML, MP-H, and MSNN on Places-365 respectively.

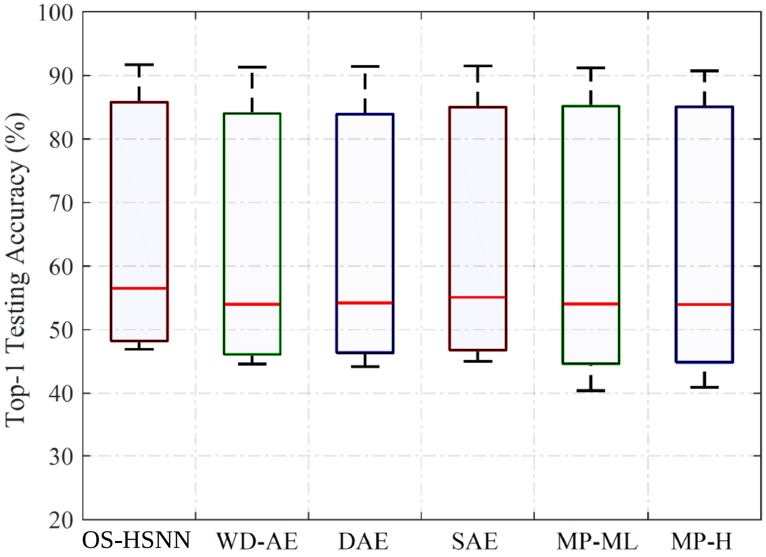


Figure 5.2 – Aggregated performance of various RL models across all the datasets.

5.3.4 Analysis on Ship-target Detection Domain

To evaluate the performance of the proposed OS-HSNN algorithm on the real-world domain, a commonly used radar signal processing dataset - HF-radar set is used for evaluation.

HF-radar dataset [148] is a radar signal processing set derived from a high-frequency surface wave radar (HFSWR), located on the coast of Bohai Bay of China.

Fig. 5.3 shows a typical radar range-Doppler (RD) image, which consists of five components: ship targets, ground clutter, background noise, sea clutter, and ionospheric clutter. The traditional ship target detection methods can be summarized through three categories: least-squares-based methods, wavelet transform (WT)-based method [151, 152, 153], and constant false alarm rate (CFAR)-based methods [154, 155]. Moreover, as ship target detection algorithm can be categorized as an object detection problem, we compared this study with several DCNN-based methods, such as faster region-based convolutional neural networks (faster R-CNN) [156], you only look once (YOLO)-v2 [157], and single-shot multibox detector (SSD) [158]. In this chapter, we propose that the OS-HSNN can be adopted as a target detection model to evaluate the performance in a specific domain.

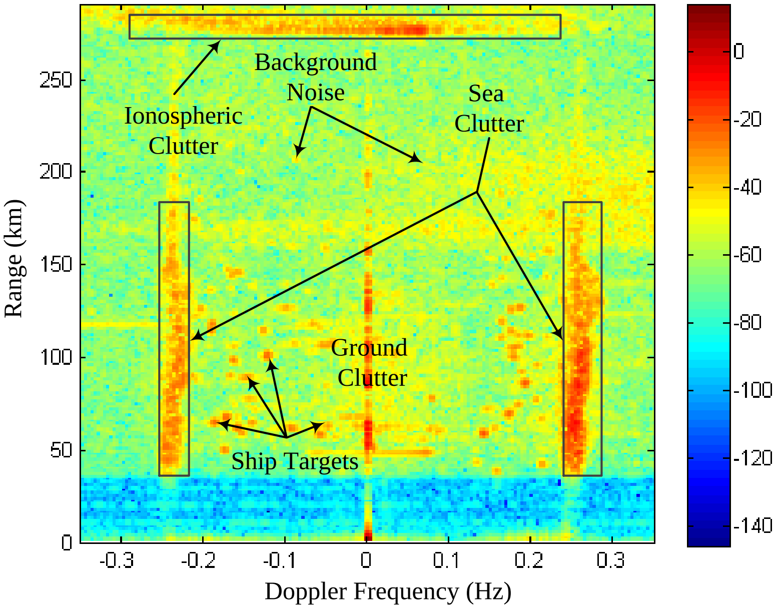


Figure 5.3 – A sample of an Range-Doppler image.

Four evaluation indices [159] for evaluating ship detection datasets are used, which are detection possibility P_d, false alarm possibility P_f, missing ratio M_r, and error ratio E_r, respectively.

$$\begin{aligned} P_d &= \frac{TP}{(TP+FN)}, \quad P_f = \frac{FP}{(FP+TP)}, \\ M_r &= 1 - P_d, \text{ and } \quad E_r = P_f + M_r, \end{aligned} \tag{5.20}$$

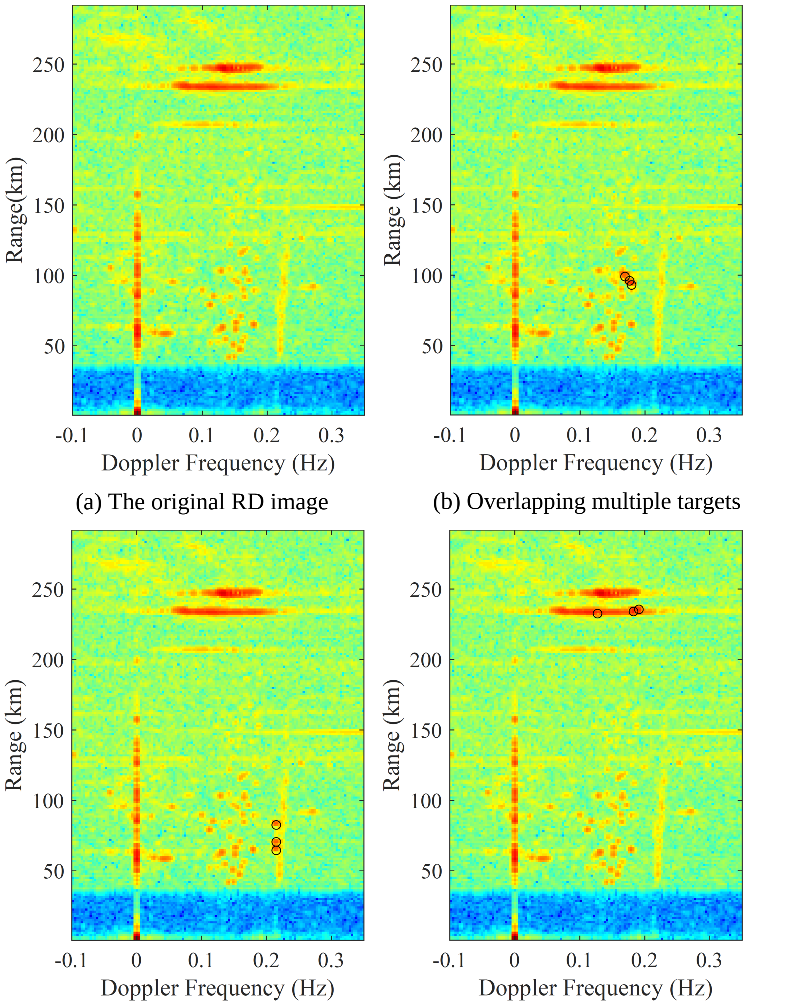
where TP, FN, and FP refer to true positive, false negative, and false positive respectively.

In order to obtain fair experiment results, the loaded feature in this part was the same as [148], which was extracted from Haar-like descriptor. It is apparent from Table 5.5 that the proposed OS-HSNN boosts detection performance, where the P_d of OS-HSNN exceeds that of [148] by almost 2%. Meanwhile, the P_f, M_r, and E_r of this study are all smaller or equal to those of the other methods.

Table 5.5 – RD image ship target detection results. The best and 2nd best results are in red and blue respectively

Method	P_d (%)	P_f (%)	M_r (%)	E_r (%)
This work	94.2	5.5	5.8	11.3
Regression-based [148]	92.6	5.8	7.4	13.2
CFAR-based [154]	85.4	13.4	14.6	27.7
Wavelet-based [153]	90.3	8.3	9.7	18.0
Faster R-CNN [156]	91.7	7.2	8.3	15.5
YOLO-v2 [157]	92.4	6.7	7.6	14.3
SSD [158]	92.4	6.9	7.6	14.5

Furthermore, three additional sets of experiments are conducted with synthetically generated targets representing the following three environmental conditions: overlapping multiple vessel-targets (OMVT), partially covered vessel-targets by sea clutter (PCVT), and ionospheric clutter interfered vessel-targets (IIVT). Particularly, for each condition, we add $3,000$ randomly generated targets in the SNR range of 20 dB to 50 dB to the test RD images (Note that the HFSWR captures the vessel-targets with SNR range [20 dB, 50 dB]). Thus, in total, $9,000$ synthetic vessel-targets are tested. Figure 5.4 presents one sample for each condition with denoted synthetic vessel-targets. Figure 5.5 shows the performance of the OS-HSNN in comparison to other traditional vessel-target detection methods, such as the wavelet-based method [153] and CFAR-based strategy [154]. The experiments were carried out 20 times independently for better analysis, and the results are shown by box-plots. Through this comparative study, it is found that the OS-HSNN has robustness in detecting vessel-targets even in challenging environmental conditions with an average detection rate of 81.97%, while the other methods, the wavelet-based method [153], CFAR-based strategy [154] show poor detection performance with an average detection rate of 77.34% and 72.24 %, respectively.



(a) The original RD image (b) Overlapping multiple targets

(c) Targets interfered by sea clutter (d) Targets interfered by ionospheric clutter

Figure 5.4 – Samples of synthetically added vessel-targets (the black small circles denote the synthetic targets).

Moreover, two sample vessel-target detection results are shown in Fig. 5.6 in support of the quantitative comparisons listed in Table 5.5. These visual results show that the proposed OS-HSNN detects and segments the vessel-targets in the RD images very tightly to the human-annotated ground truth.

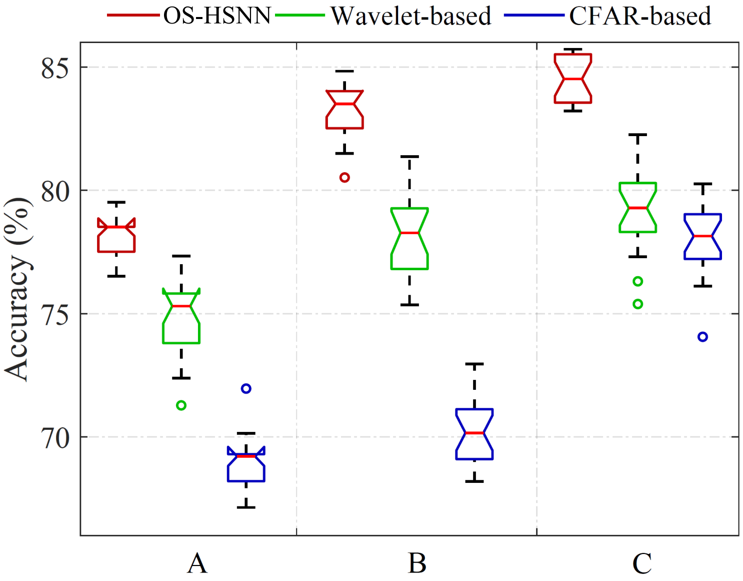
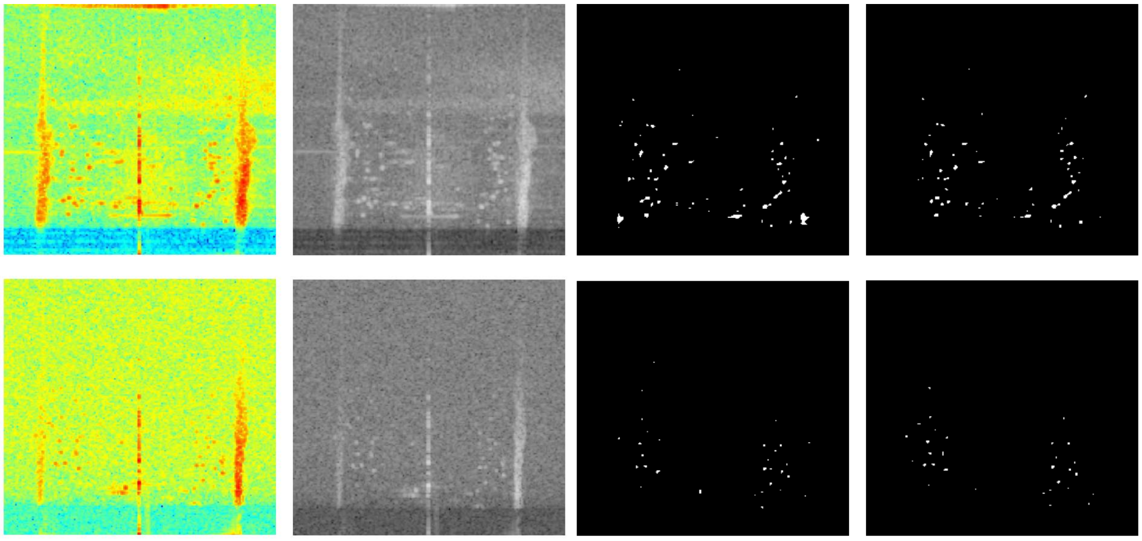


Figure 5.5 – Detection rate analysis of various methods in challenging environmental conditions: OMVT - overlapping multiple vessel-targets, PCVT - partially covered vessel-targets by sea clutter, and IIVT - ionospheric clutter interfered vessel-targets.



(a) The original RD image (b) The gray-scale image (c) Results of OS-HSNN (c) Ground truth

Figure 5.6 – Two sample detection results are shown row-wise. Columns one to four refer to the original RD images, respective gray-scale value, the detected targets (the bright pixels) localized by the OS-HSNN, and the ground truth respectively.

5.3.5 Timing Analysis

In this part, the training and testing time complexities of the proposed and the other MP inverse-based representation learning networks are tabulated in Table 5.6. Note that both the training time and inference time recorded in this table do not contain the VGG-16 feature learning and extraction part. To allow fair comparisons, all these methods are applied with the proposed batch-by-batch strategy. From Table 5.6, it can be found that OS-HSNN takes 5% to 10% longer time for training. The reason is that it searches the optimal feature representations by pulling back the error term in each training iteration. However, it offers a better generalization performance than the other methods.

Table 5.6 – Processing time w/ FT-VGG16: Tr. (s) is the total training time in second, Te. (ms) is the mean average testing time per frame in millisecond.

Datasets	MP-ML [24]		MP-H [42]		OS-HSNN		MSNN [23]	
	Tr.	Te.	Tr.	Te.	Tr.	Te.	Tr.	Te.
Scene15	**46.1**	1.3	56.7	1.6	56.4	**1.2**	59.2	1.3
Caltech101	**54.8**	1.2	79.8	1.5	82.0	**1.2**	89.6	1.3
Food251	**791.4**	1.3	906.7	1.7	1001.2	**1.2**	1022.5	1.3
Place365-1	**1682.9**	1.5	2248.0	1.8	2562.5	**1.4**	2610.7	1.5

Furthermore, the proposed OS-HSNN records an inference speed with 1.2 ms mean average testing time, which is the same as the Wi-HSNN (cf. Table 4.8). The reason is that the OS-HSNN and Wi-HSNN have the same number of parameters and structures, and the procedures in the testing stage are the same.

5.4 Conclusion

A batch-by-batch subnetwork-based multi-modal feature reinforcement strategy is proposed to both refine the meaningful knowledge from various sources as well as classify the input objects. The key characteristics are as follows: (i) It encodes multiple-modality features, generates optimal representations, and classifies the patterns with one single structure. (ii) The batch-by-batch representation learning technique enables the proposed OS-HSNN to efficiently handle big data. For instance, it can process a dataset with more than 1.8 million images without the requirement of a high-performance computing (HPC) device.

The experiments on eight image classification and ship detection datasets show that the proposed refinement framework with a flexible network structure outperforms the conventional algorithms. From an application point of view, it can be employed to detect ship-targets from HFSWR range-Doppler images.

Chapter 6

Hierarchical One-Class Model with Subnetwork for RL and Outlier Detection

The multilayer one-class classification (OCC) frameworks have been widely investigated for anomaly and outlier detection. However, most multilayer OCC algorithms suffer from loosely-connected feature coding, making the generated latent space that cannot properly generate a highly discriminative representation between object classes. This problem is more noticeable when handling large-scale datasets with high complexity. To alleviate this deficiency, two novel OCC frameworks, namely OCC structure using hierarchical subnetwork neural network (OC-HSNN) and maximum correntropy-based OCC structure using hierarchical subnetwork neural network (MCOC-HSNN), are proposed in this chapter. The novelties are as follows: i) The subnetwork is used to build the discriminative latent space; ii) Existing works utilize mean square error (MSE) to learn low-dimensional features, the MCOC-HSNN uses maximum correntropy criterion (MCC) for discriminative feature encoding; iii) A brand-new OCC dataset called CO-Mask is gathered. Experimental results on the visual classification domain with a varying number of training samples from 6,131 to 513,061 demonstrate that the proposed OC-HSNN and MCOC-HSNN achieve superior performance compared to the existing multilayer OCC models.

6.1 Introduction

One-class classification is a typical machine learning task aiming to develop processing algorithms where the negative pattern is either absent or not perfectly defined. The input patterns from one category (denoted as the target class) are well characterized, while there is little information on the negative category (outlier class) [160, 161]. Different from the multi-class classification methods that classify the input pattern into one of the pre-defined categories, the OCC strategies handle the problems where the unknown data does not belong to any of those classes. The OCC algorithms are, therefore, deemed as more suitable for solving some real-world applications, such as anomaly detection [162], document classification [163], and social media rumour detection [164].

In recent years, there has been a considerable amount of work proposed in the field of OCC. The earliest study of OCC can be traced back to 1962 in [165], where authors stated that the estimation of the target class's probability density function could be helpful to the outlier detection, and the negative samples were identified according to a pre-set threshold. Consequently, several improved OCC schemes, such as one-class linear programming method [166], one-class minmax probability machine [167], and tree-based one-class classifier [168] were proposed. Recently, the one-class algorithms based on the MP inverse strategy have been widely investigated. The one-class ELM [169] is a single layer neural network, having the advantages of quicker training time and excellent generalization performance. Following that, explosive developments on multilayer MP inverse-based one-class networks have been witnessed [25, 161, 170, 171, 172]. These strategies generally apply the AE as a cornerstone of their learning, using the two-step modelling approaches: first, the low-dimensional representation of input data is obtained through the use of AE; then an estimation network, the single-layer one-class classifier, is applied on the compact encoding for the final classification. Compared to other OCC algorithms, multilayer one-class networks stack multiple AEs for deep feature learning, leading to higher accuracy and robustness.

However, the existing MP inverse-based OCC approaches have several limitations: First, these algorithms were only validated on small-scale and medium-scale datasets, such as MNIST and NORB datasets with a sample size of no more than 60,000 samples. However, the MP inverse-based OCC algorithms have rarely analyzed large-scale datasets with more than $100K$ training patterns.

Second, the multilayer MP inverse-based OCC networks suffer from generating local-level encodings. Similar to the MP inverse-based multi-class classification algorithms that utilize the two-step learning strategy for pattern classification, the multilayer OCC methods use two independent sub-modules to extract representative features and to classify the input pattern. However, this strategy is inefficient. Hence, we propose a one-step learning algorithm to encode the raw features and do the classification simultaneously. The subnet-based RL algorithms [23, 173, 174] have shown the benefits of discriminative feature learning, and the proposed Wi-HSNN in Chapter 4 has already achieved superior performance over the other state-of-the-art multilayer MP inverse-based algorithms in multi-class classification.

Third, the existing multilayer OCC algorithms are not powerful in suppressing non-Gaussian noise and outliers in the input data. The MCC is an effective tool in impulsing noise and outliers, which has been commonly utilized in MP inverse-based one-class processing. However, the recently proposed MCC-based methods [170, 171] only employed MCC strategy on the final classification. In other words, they still utilize the MSE criterion to generate optimal coding. The success of machine learning frameworks generally depends on data representation, as it enables us to differentiate between different concepts [60]. For example, DCNNs have proven to be superior to the traditional methods in most practical applications [175]. In the state-of-the-art DCNNs, like ResNet [88] and DenseNet [176], hundreds of convolutional layers operating as Gabor filters and colour blob detectors are embedded [177], while only one simple fully-connected layer like softmax is used on the top to predict the label.

To overcome the above-identified limitations, especially the first and second, this chapter proposes the OC-HSNN to learn the optimal representation and classify the input pattern, simultaneously. Then, to further address the third issue, the MCOC-HSNN is provided to handle the non-Gaussian noise and to learn the robust low-dimensional representations. The contribution of this chapter is threefold:

1. **A novel OCC scheme called OC-HSNN:** A novel multilayer subnet-based structure for OCC is proposed. Further, we believe utilizing MP inverse to handle large-scale OCC datasets is meaningful - as far as we know, no existing research has attempted to do so.

2. **A MCC-based method named MCOC-HSNN:** In the presence of non-Gaussian noise, the MCOC-HSNN is designed in low-dimensional RL to improve the performance of OC-HSNN.

3. **New OCC dataset:** A dataset (CO-Mask) for misinformation detection is created for this study with texts collected from the "big three" news agencies (Associated Press, Reuters, and Bloomberg) on wearing masks as a way to curb COVID-19.

6.2 Related Works on One-class Classification

The OCC problems have been widely investigated over the past few decades. In 1962, Parzen *et al.* [165] estimated the probability density function of the target as the measurement for anomaly detection, and the outliers were identified with a thresholding strategy. Consequently, the work in [166] provided a linear programming dissimilarity-data description (LPDD) classifier to detect the outliers, which utilized the reformulation for general dissimilarity representations. Lanckriet *et al.* [167] built the one-class minimax probability machine to minimize the worst-case probability of data patterns, given only the mean value and covariance matrix of the data distribution. In the work of [168], the authors proposed a tree-based architecture for OCC. It develops a minimum spanning (MS) tree with the target-class samples only, and the testing pattern is identified by the use of the closest edge of the MS tree. Li *et al.* [178] proposed an outlier detection strategy using an improved one-class support vector machine. In this method, the low-rank constraint is first employed to divide the data into several groups, then the linear solution for each cluster is learned separately. Consequently, the authors in [179] perform multi-view low-rank analysis (MLRA) for outlier detection, and the cross-view low-rank coding is utilized to reveal the intrinsic structures of the input. Wang *et al.* [180] introduced a one-class discriminative subspace (BODS) classifier for outlier detection, where a pair of subspaces are used for discriminative regularity modelling.

With the rise of enthusiasm in deep learning, the multilayer OCC strategies gained traction. AE is the most common architecture utilized in these algorithms, using iterative learning schemes like BP as the baseline of its training [149, 101, 150]. Recently, AEs learned with MP inverse techniques have also been developed [25, 172, 161]. Compared with the BP-based learning strategies, the MP inverse has the benefits of fast training speed and good generalization performance. The authors in [25] first proposed a multilayer OCC structure to detect the outliers. The AEs trained with MP inverse are utilized to learn the latent space coding, whereas the OC-ELM is used for classification. Following that, Wang *et al.* [172] proposed a hierarchical one-class algorithm to discriminatively learn the low-dimensional features. Instead of applying the traditional MP inverse-based AEs to learn the representations, the sparse matrix AE (SMA) is used. In [161], a within-class scatter information (WSI) constraint-based one-class structure (OC-WSI) is proposed to enhance the encoding discriminability. Specifically, the estimated covariance matrix is designed to minimize the reconstruction error and the within-class scatter of the hidden space representations. However, the traditional MP inverse-based OCC algorithms use MSE as the criterion to measure the error matrix, which is insufficient in handling non-Gaussian noise.

To address the above-mentioned limitation, MCC-based OCC strategies have been proposed [171, 170]. For example, Cao *et al.* [170] proposed a multilayer MCC-based OCC network. First, multiple AEs developed with MSE criterion are stacked to learn the optimal low-dimensional representations, and then the OC-ELM trained with MCC strategy (MCOC-ELM) is employed to identify the outliers. Similarly, the work in [171] first built a multilayer framework with the stack of WSI constraint-based AEs, and then the MCOC-ELM is adopted as the top-net to detect the outliers. Besides, in that paper, the top-net is extended for kernel learning for better generalization performance.

However, the state-of-the-art MP inverse-based OCC algorithms have the following limitations: i) These strategies only verify their frameworks on small-scale datasets; ii) All these OCC algorithms learn the low-dimensional encoding and do the final classification with two separate stages, resulting in sub-optimal representations and limited generalization performance; iii) The MCC is only utilized in the final stage classification, which is not efficient. The latent space is still built through the MSE criterion. To alleviate these deficiencies, in this chapter, two novel OCC schedules using subnet-based structures are introduced.

6.3 The Proposed Algorithms

The structure of the proposed OC-HSNN and MCOC-HSNN is very similar to those of the Wi-HSNN and OS-HSNN. The main difference between OC-HSNN and MCOC-HSNN lies in the optimization of the weight. The diagram of these two models is shown as Fig. 6.1. Further, for the notations used in this chapter, please refer to Table 4.1.

6.3.1 The Proposed OC-HSNN

The detailed learning steps of the proposed OC-HSNN are described in Algorithm 6.1. The differences between the proposed OC-HSNN and MCOC-HSNN are: The OC-HSNN utilizes MSE to calculate the subnet weight $\mathbf{W}_i^r$ (cf. Line 10 in Algorithm 6.1), while the MCOC-HSNN uses the MCC for subspace encoding (cf. Line 13 in Algorithm 6.1). The model consists of two learning stages: Stage 1 - initialization, and Stage 2 - iterative subspace encoding. The former stage aims to warm up the network, while the latter stage can incrementally enrich the encoding Γ by adding new subnet nodes.

Different from the multi-class classification that aims to minimize the distance between the prediction and the ground truth, the goal of OCC is to reject a small number of samples during the training stage according to the output. In the network training stage, only the target-class samples are fed as the input, and the ground truth of training samples is set to 1. Thus, the OC-HSNN uses a set of target samples to learn the target distribution and details. In the testing stage, the closer the network output $\mathbf{Y}_j$ (j refers to the index of testing pattern, $1 \leq j \leq N$) approaches 1, the more likely the input data belongs to the target class. In particular, a threshold μ is utilized to separate the outliers from the targets: The j-th input is considered as the outlier if the output distance $\varepsilon_j = |\boldsymbol{O}_j - 1|$ is larger than the threshold μ. In this sense, we consider the output $\mathbf{Y}_j$ of data points that are smaller than $1 - \mu$ (or larger than $1 + \mu$) as the outliers.

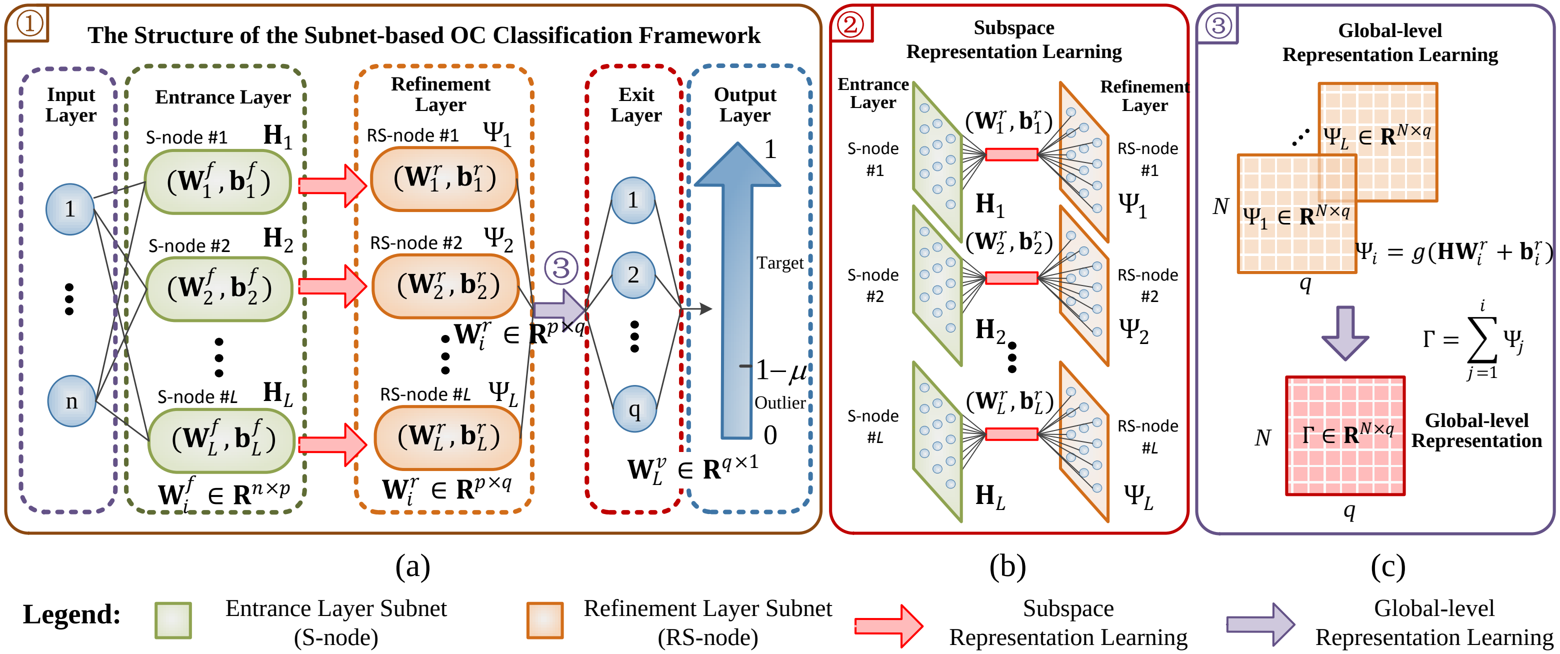

①
The Structure of the Subnet-based OC Classification Framework
Input Layer
Entrance Layer
Refinement Layer
Exit Layer
Output Layer
S-node #1
S-node #2
S-node #L
$\mathbf{H}_1$
$\mathbf{H}_2$
$\mathbf{H}_L$
$(\mathbf{W}_1^f, \mathbf{b}_1^f)$
$(\mathbf{W}_2^f, \mathbf{b}_2^f)$
$(\mathbf{W}_L^f, \mathbf{b}_L^f)$
$\mathbf{W}_i^f \in \mathbf{R}^{n \times p}$
RS-node #1
RS-node #2
RS-node #L
Ψ_1
Ψ_2
Ψ_L
$(\mathbf{W}_1^r, \mathbf{b}_1^r)$
$(\mathbf{W}_2^r, \mathbf{b}_2^r)$
$(\mathbf{W}_L^r, \mathbf{b}_L^r)$
$\mathbf{W}_i^r \in \mathbf{R}^{p \times q}$
$\mathbf{W}_i^r \in \mathbf{R}^{p \times q}$
③
1
n
1
2
q
$\mathbf{W}_L^v \in \mathbf{R}^{q \times 1}$
1
Target
$1-\mu$
Outlier
0
(a)
②
Subspace Representation Learning
Entrance Layer
Refinement Layer
S-node #1
S-node #2
S-node #L
RS-node #1
RS-node #2
RS-node #L
$(\mathbf{W}_1^r, \mathbf{b}_1^r)$
$(\mathbf{W}_2^r, \mathbf{b}_2^r)$
$(\mathbf{W}_L^r, \mathbf{b}_L^r)$
$\mathbf{H}_1$
$\mathbf{H}_2$
$\mathbf{H}_L$
Ψ_1
Ψ_2
Ψ_L
(b)
③
Global-level Representation Learning
$\Psi_L \in \mathbf{R}^{N \times q}$
$\Psi_1 \in \mathbf{R}^{N \times q}$
N
q
$\Psi_i = g(\mathbf{H}\mathbf{W}_i^r + \mathbf{b}_i^r)$
$\Gamma = \sum_{j=1}^{i} \Psi_j$
N
$\Gamma \in \mathbf{R}^{N \times q}$
q
Global-level Representation
(c)
Legend:
🟩 Entrance Layer Subnet (S-node)
🟧 Refinement Layer Subnet (RS-node)
Subspace Representation Learning
Global-level Representation Learning

Figure 6.1 – The structure of the OC-HSNN and MCOC-HSNN are depicted as (a), while (b) and (c) show the details of subspace RL and the latent space RL. The main difference between these two models lies in calculating subsapce weight $\mathbf{W}^r$: The OC-HSNN and MCOC-HSNN use MSE and MCC to learn the optimal $\mathbf{W}^r$, respectively.

Algorithm 6.1 The proposed OC-HSNN and MCOC-HSNN

Inputs: The input data $(\mathbf{X}, \mathbf{T})$, the maximum number of subnetwork L
Outputs: The network output $\mathbf{Y}$

```
• Stage 1: Initialization
Procedure I: $\mathbf{W}_1^f$, $\mathbf{W}_1^r$, $\Gamma$, and $\mathbf{W}_1^v \leftarrow$ Assign weight
$\mathbf{E}_1 = \mathbf{T} - \Gamma \cdot \mathbf{W}_1^v$    % Obtain the output error.
$\mathbf{P}_1 = \mathbf{E}_1(\mathbf{W}_1^v)^\dagger = \mathbf{E}_1 \cdot \left(\frac{I}{C} + (\mathbf{W}_1^v)^T \mathbf{W}_1^v\right)^{-1} (\mathbf{W}_1^v)^T$ %Pull back the error from the output layer
    to the refinement layer.
• Stage 2: Iterative Subspace Encoding
for $(i = 2, i <= L, i++)$ do
    $\mathbf{W}_i^f$, $\mathbf{b}_i^f$ % Randomly assign weight $\mathbf{W}_i^f$.
    $\mathrm{H}_i = g(\mathbf{X} \cdot \mathbf{W}_i^f + \mathbf{b}_i^f)$
    if OC-HSNN then
        Procedure II: $\mathbf{W}_i^r$, $\Psi_i \leftarrow$ MSE-based Weight Learning ($\boldsymbol{X}$, $\boldsymbol{P}$)
    end if
    if MCOC-HSNN then
        Procedure III: $\mathbf{W}_i^r$, $\Psi_i \leftarrow$ MCC-based Weight Learning ($\boldsymbol{X}$, $\boldsymbol{P}$)
    end if
    $\Gamma = \sum_{j=1}^{i} \Psi_j$% $\Psi_j$ is the subspace features, while $\Gamma$ refers to the global-level (latent space)
        representations.
    $\mathbf{W}_i^v = (\frac{I}{C} + \Gamma^T \Gamma)^{-1} \Gamma^T \mathbf{T}$
    $\mathbf{E}_i = \mathbf{T} - \Gamma \mathbf{W}_i^v$
    $\mathbf{P}_i = \mathbf{E}_i(\mathbf{W}_i^v)^\dagger = (\mathbf{T} - \Gamma \mathbf{W}_i^v) \cdot \left(\frac{I}{C} + (\mathbf{W}_i^v)^T \mathbf{W}_i^v\right)^{-1} (\mathbf{W}_i^v)^T$
end for
$\mathbf{Y} = \Gamma \cdot \mathbf{W}_L^v$  % $\mathbf{Y}$ is the results.
```

Procedure I: *Assign weight*
Inputs: NA
Outputs: $\mathbf{W}_1^f$, $\mathbf{W}_1^r$, Γ, and $\mathbf{W}^v$

```
$\mathbf{W}_1^f$, $\mathbf{b}_1^f$    % Orthogonal random generation.
$\mathrm{H}_1 = g(\mathbf{X} \cdot \mathbf{W}_1^f + \mathbf{b}_1^f)$
$\mathbf{W}_1^r$, $\mathbf{b}_1^r$    % Orthogonal random generation.
$\Psi_1 = g(\mathrm{H}_1 \cdot \mathbf{W}_1^r + \mathbf{b}_1^r)$
$\Gamma = \Psi_1$    % Obtain current latent representation.
$\mathbf{W}^v = (\frac{I}{C} + \Gamma^T \Gamma)^{-1} \Gamma^T \mathbf{T}$% The weight are generated with MSE.
return $\mathbf{W}_1^f$, $\mathbf{W}_1^r$, $\Gamma$, and $\mathbf{W}^v$
```

Procedure II: *MSE-based Weight Learning* $(\mathbf{X}, \mathbf{P})$
Inputs: $\mathbf{X}$, $\mathbf{P}$
Outputs: $\mathbf{W}$, H

```
$C$, $\mathbf{b}$    % A preset regularization term $C$ and bias $\mathbf{b}$.
$\mathbf{X}^\dagger = (\frac{I}{C} + \mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T$
$\mathbf{W} = \mathbf{X}^\dagger \mathbf{P}$ % $\mathrm{X}^\dagger$ is the MP inverse.
$\mathrm{H} = g(\mathbf{X} \cdot \mathbf{W} + \mathbf{b})$
return $\mathbf{W}$, H
```

Procedure III: *MCC-based Weight Learning* ($\mathbf{X}$, $\mathbf{P}$)
Inputs: $\mathbf{X}$, $\mathbf{P}$
Outputs: $\mathbf{W}^q$, H

```
$K$, $\eta$, $b$, $\mathbf{W}^0 = 0$ % $K$ is the maximum iteration, $\eta$ is the tolerance error
for ($q = 1, q <= K, q++$) do
    $\boldsymbol{\lambda}_j = \mathbf{X}_j \mathbf{W}^{q-1} - \mathbf{P}_j$, $j = 1, 2, \cdots, N$% Calculate $\lambda$ for each sample
    $[\boldsymbol{\Lambda}]_{jj} = \psi(\boldsymbol{\lambda}_j)$, $j = 1, 2, \cdots, N$% Calculate $\Lambda$ matrix
    $\mathbf{W}^q = (\mathbf{X}^T \boldsymbol{\Lambda} \mathbf{X} + \frac{2I}{C})^{-1} \mathbf{X}^T \boldsymbol{\Lambda} \mathbf{P}$
    if $J(\mathbf{W}^q) - J(\mathbf{W}^{q-1}) \leq \eta$ then
        break;    % Exit FOR loop.
    end if
end for
$\mathrm{H} = g(\mathbf{X} \cdot \mathbf{W} + b)$
return $\mathbf{W}^q$, H
```

As for the choosing of μ, it is determined automatically using the training error where a pre-set percentage of samples is assumed to be the outlier. Here, we set the percentage to be 10% as an example. First, the output ε_j of each training sample is sorted in a descending order: $\varepsilon_{(1)} \geq \varepsilon_{(2)} \geq, \cdots, \geq \varepsilon_{(N)}$; Then, the optimal threshold μ is chosen to be $\varepsilon_{(t)}$ where the index t is determined by rejecting the first $t-1$ number of samples as outliers (in this case, $t = N \times 10\%$).

6.3.2 The Proposed MCOC-HSNN

The OC-HSNN applies the traditional MSE as the criterion to build the objective function. To enhance the robustness of the proposed method, the MCC is adopted to calculate $\mathbf{W}_i^r$ for discriminative feature learning. In this subsection, the correntropy-based objective function is first given, and the characteristics of the proposed MCOC-HSNN are then introduced.

6.3.2.1 The Objective Function with Correntropy

Given a dataset $\{\mathbf{X}, \mathbf{P}\}$ with N number of arbitrary samples, where $\mathbf{X}$ and $\mathbf{P}$ are the input training sample and the pulled back error (expected target), respectively. Suppose H is the entrance layer feature. Then, an MSE-based objective function $J(\cdot)$

can be computed as in Eq. (6.1).

$$\min\ J = \frac{C}{2}||\mathrm{H}\mathbf{W} - \mathbf{P}||_2^2 + \frac{1}{2}||\mathbf{W}||^2, \tag{6.1}$$

where $\mathbf{W}$ is the refinement layer weight. C is the regularization term, which is a positive value. Here, the optimal $\mathbf{W}$ is estimated using MP inverse method as in Eq. (6.2) [181].

$$\mathbf{W} = \left(\frac{I}{C} + \mathrm{H}^T\mathrm{H}\right)^{-1} \mathrm{H}^T\mathbf{P}, \tag{6.2}$$

where I is the identity matrix.

In order to improve the robustness of the model, the correntropy-based objective function in Eq. (6.3) is employed instead of the MSE-based loss function.

$$\min\ J = -\frac{C}{2}V(\mathrm{H}\mathbf{W}, \mathbf{P}) + \frac{1}{2}||\mathbf{W}||^2. \tag{6.3}$$

where $V(\mathrm{H}\mathbf{W}, \mathbf{P})$ is computed using the Gaussian kernel, which is defined in Eq. (6.4) shown as below.

$$V(\mathrm{H}\mathbf{W}, \mathbf{P}) \approx \frac{1}{N}\sum_{j=1}^{N} G_\sigma(\mathrm{H}_j\mathbf{W} - \mathbf{P}_j - k_j), \tag{6.4}$$

where H_j, k_j and $\mathbf{P}_j$ refer to the hidden layer feature, the center and the target value of the j-th input data, respectively. By substituting Eq. (6.4) into Eq. (6.3), we can have the following equation.

$$J = -\frac{C}{2N}\sum_{j=1}^{N} G_\sigma(\mathrm{H}_j\mathbf{W} - \mathbf{P}_j - k_j) + \frac{1}{2}||\mathbf{W}||^2. \tag{6.5}$$

By taking the partial derivative of Eq. (6.5) wrt $\mathbf{W}$ to zero, we have the following

$$\begin{aligned}
\frac{\partial J}{\partial \mathbf{W}} &= \mathbf{W} - \frac{C}{2N}\frac{\partial}{\partial \mathbf{W}}\left(\sum_{j=1}^{N} G_\sigma(\mathrm{H}_j\mathbf{W} - \mathbf{P}_j - k_j)\right) \\
&= \mathbf{W} - \frac{C}{2N}\sum_{j=1}^{N}\frac{\partial}{\partial \mathbf{W}}\left(\exp\left(-\frac{(\mathrm{H}_j\mathbf{W} - \mathbf{P}_j - k_j)^2}{2\sigma^2}\right)\right) \\
&= \mathbf{W} - \frac{C}{2N}\sum_{j=1}^{N}\exp\left(-\frac{(\mathrm{H}_j\mathbf{W} - \mathbf{P}_j - k_j)^2}{2\sigma^2}\right)\cdot \mathrm{H}_j^T \cdot \left(-\frac{1}{\sigma^2}\right) \\
&\quad \cdot (\mathrm{H}_j\mathbf{W} - \mathbf{P}_j - k_j) \\
&= \mathbf{W} + \frac{C}{2N}\sum_{j=1}^{N}\frac{\exp\left(-\frac{(\mathrm{H}_j\mathbf{W} - \mathbf{P}_j - k_j)^2}{2\sigma^2}\right)}{\sigma^2}\mathrm{H}_j^T(\mathrm{H}_j\mathbf{W} - \mathbf{P}_j - k_j) \\
&= \mathbf{W} + \frac{C}{2N}\sum_{j=1}^{N}\frac{G_\sigma(\mathrm{H}_j\mathbf{W} - \mathbf{P}_j - k_j)}{\sigma^2}\mathrm{H}_j^T(\mathrm{H}_j\mathbf{W} - \mathbf{P}_j - k_j),
\end{aligned} \tag{6.6}$$

wherein $(\mathrm{H}_j\mathbf{W} - \mathbf{P}_j - k_j) \rightarrow \boldsymbol{\lambda_j}$, so that Eq. (6.6) can be rewritten as

$$\left(\frac{C}{2N}\sum_{j=1}^{N}\frac{G_\sigma(\boldsymbol{\lambda}_j)}{\sigma^2}\mathrm{H}_j^T\mathrm{H}_j + I\right)\mathbf{W} = \frac{C}{2N}\sum_{j=1}^{N}\frac{G_\sigma(\boldsymbol{\lambda}_j)}{\sigma^2}\mathrm{H}_j^T\mathbf{P}_j. \tag{6.7}$$

Therefore, the optimal solution for minimizing the objective function J can be expressed as

$$\mathbf{W} = (\mathrm{H}^T\boldsymbol{\Lambda}\mathrm{H} + \frac{2I}{C})^{-1}\mathrm{H}^T\boldsymbol{\Lambda}\mathbf{P}, \tag{6.8}$$

where $\boldsymbol{\Lambda}$ is a diagonal matrix with $[\boldsymbol{\Lambda}]_{jj} = \psi(\boldsymbol{\lambda_j})$, $\psi(\cdot)$ is the projection of a vector into the Hilbert space. Here, it is a Gaussian kernel. The $\boldsymbol{\Lambda}$ is calculated according to the weight $\mathbf{W}$. Thus, the optimal solution is obtained through a fixed-point iterative process described in Algorithm 6.1 - Procedure III.

6.3.2.2 The Characteristics of MCOC-HSNN

Algorithm 6.1 details the learning steps of the proposed MCOC-HSNN. Compared to OC-HSNN, the MCOC-HSNN is characterized by two properties.

- The MCOC-HSNN utilizes the MCC scheme to architect the subspace (optimize the weight $\mathbf{W}_i^r$). The generated low-dimensional encoding in MCOC-HSNN is not sensitive to the outliers and non-Gaussian noise, thus showing better generalization performance. However, it increases the computational workload, as the users need to use the fixed-point iteration algorithm to find the optimal weight.

- In this chapter, we hypothesize that the performance of one model depends on the encoding of the input data: When a distinctive latent space feature (Γ) is achieved, there is no need to further use MCC to optimize the output layer weight ($\mathbf{W}^v$). A simple LS strategy is enough to fit the objective function. Thus, in MCOC-HSNN, only the subspace weight ($\mathbf{W}_i^r$) are generated with the use of MCC. This characteristic is experimentally verified in Section 6.4.2 - *Model Settings*.

6.4 Experimental Results

6.4.1 Experimental Setup

6.4.1.1 The Environment

All of the algorithms outlined in this chapter are evaluated in the MATLAB 2020a environment. The experiments are conducted in a workstation with a 256 GB main memory and a 2.8 GHz E5-2650 processor.

6.4.1.2 The Dataset

In this chapter, 27 OCC datasets ranging from image classification domain to misinformation detection domain are used to evaluate the proposed OC-HSNN and MCOC-HSNN. The details of these datasets are elaborated in Table 6.1. For all these datasets,

Table 6.1 – Summary of Datasets

Datasets	Training Data	Testing Data[1]	Features	Class[2]
MNIST-2	6,742	10,000 / 1,135	784	10
MNIST-4	6,131	10,000 / 1,010	784	10
MNIST-5	5,842	10,000 / 982	784	10
NORB-1/2/3	4,860	24,300 / 4,860	2,048	5
Fashion-2/3/4/5	6,000	10,000 / 1,000	784	10
CIFAR-10[3]	5,000	10,000 / 1,000	3,072	10
Place-A	119,168	36,500 / 2,400	2,048	365
Place-B	171,650	36,500 / 3,500	2,048	365
Place-C	165,394	36,500 / 3,400	2,048	365
Place-D	56,849	36,500 / 1,200	2,048	365
Place-ABCD	513,061	36,500 / 10,500	2,048	365
Food+	118,475	48,494 / 11,994	2,048	251
CO-Mask	4,907	3,725 / 1,310	3,000	2

[1] Testing Data: Total number of testing data / The number of target data.
[2] Class: The number of classes in the original dataset.
[3] CIFAR-10: Each class in CIFAR-10 is treated as the target class separately.

one specific category (at least one class) is manually labelled as the target class, whereas the rest is defined as the outlier class. The datasets can be divided into four families: small-scale and large-scale visual datasets, food identification dataset, and misinformation detection dataset.

Small-scale visual datasets (MNIST-2/4/5, NORB-1/2/3, Fashion-2/3/4/5, and CIFAR-10) have fewer training samples with less than 50 K patterns. Following the previous works [170, 161, 25, 171], the number followed the symbol "-" is used to show the specific label of the target class. For the MNIST dataset, classes 2 (MNIST-2), 4 (MNIST-4), and 5 (MNIST-5) are applied as the target category, and the rest categories are labelled as the outlier class, respectively. Similar situations occur with NORB (NORB-1/2/3), Fashion (Fashion-2/3/4/5) dataset. For the MNIST-2/4/5 and Fashion-2/3/4/5 datasets, the raw images are loaded as the input ($n = 28 \times 28$). For the NORB dataset, the zero phase component analysis whitening [182] is used for pre-processing, and the input dimension n of NORB-1/2/3 datasets is 2,048. As for the CIFAR-10 dataset, this dataset contains 10 labels. In each setup, one of the classes is considered as the target class, whereas the rest classes are the outliers. Hence, the training set for CIFAR-10 contains 5,000 patterns. To comprehensively evaluate the proposed methods, the raw images and high-level features extracted by rotation prediction [183] are utilized as the algorithms' input, respectively.

Large-scale visual datasets (Place-A/B/C/D/ABCD) have a relatively large number of patterns that have more than 50 K training patterns. The Places-365 dataset, which contains 365 categories, is one of the largest scene classification dataset with a sample size of over 1.8 million samples. Four large OCC datasets Place-A/B/C/D are prepared by merging the categories of initials "A", "B", "C", or "D" respectively as the target class, while the rest are considered as the outlier class. To further evaluate the performance of OC-HSNN and MCOC-HSNN, one more test is conducted by comparing algorithms with the Place-ABCD dataset that merges the classes starting with "A", "B", "C", and "D" as the target, and the rest of the classes are labelled as outliers. Consequently, for Place-A/B/C/D/ABCD, the validation set containing 36,500 images is used for testing.

For the large-scale datasets (Place365-A/B/C/D/ABCD), the Inception-V3 [87] is utilized to extract the high-level feature from the raw samples. The Inception-V3 is obtained from the ImageNet pre-trained model. The 2048-dimensional feature vector from the average pooling layer "avg_pool" is extracted as the input ($n = 2,048$). For fair comparison, the high-level feature is fed to all the OCC methods.

Food identification dataset (Food+). Food-251 [131] is one of the largest food image classification datasets, having 118,475 training images and 11,994 testing patterns. Here, the food images are considered as the target class, whereas the non-food patterns belong to the outlier class. Specifically, we enriched the Food-251 dataset to conform to the food image identification by merging 365,000 non-food images (randomly selected from the Place-365 dataset) into the testing set of Food-251. Table 6.1 shows the statistics of this dataset which is named Food+.

Misinformation detection dataset (CO-Mask). Misinformation has many negative impacts on society. For instance, false or misleading statements have proved counterproductive in the world's combat against COVID-19, which inflected millions of people around the world [184]. While existing research garnered important insights on rumour detection in social media, there is still a paucity of advancements in our field that could help detect tendencies for news, in general, to be misleading. In this chapter, we approach misinformation detection in digital platforms through OCC algorithms [185]. The main reason is that misinformation (or fake news) is arbitrarily labelled according to the user's volition, while the true news is collected similarly [186, 164]. Therefore, true news and misinformation are considered as the target class and outlier class respectively.

In this chapter, we created one misinformation detection dataset ("CO-Mask")[1] to verify the effectiveness of the proposed OCC methods. The CO-Mask dataset includes statements from the "big three" news agencies (Associated Press, Reuters, and Bloomberg) related to people's attitude toward wearing masks during the COVID-19. It contains 4,907 training samples (true news samples) and 3,725 testing patterns. The embeddings (300-dimensional vectors) of each text pattern are obtained via fastText[1]. The number of word vectors in the sequence is 10. Thus, the input dimensionality of each vector is 3,000.

6.4.1.3 The Rival Methods

The proposed methods are evaluated with three families of algorithms:

The traditional methods. Several BP-based AEs, such as weight-decay regularization-based AE (WD-AE) [149], denoise AE with Gaussian noise (DAE^G) [101] or with binary noise (DAE^B) [101], and sparse AE (SAE) [150] are tested. Furthermore, some advanced OCC algorithms, such as the deep convolutional generative adversarial network called AnoGAN [187], deep support vector data description (DSVDD) [188] and self-supervised learning-based kernel density estimation (SSLKDE) [189] are compared.

[1] CO-Mask dataset can be download here.

[1] FastText can be download here.

The MP inverse-based methods. In this chapter, some MSE-based OCC algorithms, including single-layer one-class algorithm (OC-S) [169], multilayer-based one-class neural network (OC-M) [42], hierarchical one-class model (OC-H) [98], multilayer kernel-based one-class structure (OC-K) [25], OC-based SMA algorithm (OC-SMA) [172], and OC-WSI [161] are used for evaluation.

The maximum correntropy-based one-class classification (MCOC) methods. The multilayer MCOC network (MCOC-M) [170], the multilayer MCOC network using kernel learning (MCOC-K) [171], hierarchical MCOC network (MCOC-H) [98], WSI constraint-based hierarchical MCOC network (MCOC-WSI) [171] utilized for comparison.

6.4.1.4 Configurations of the OCC Methods

The detailed hyperparameter settings of the proposed work and the compared rival methods are summarized in Table 6.2 for reproducibility. For DSVDD, AnoGAN and SSLKDE, we follow the hyperparameter settings elaborated in [188], [187], and [189], respectively. It is worth noting that for all the proposed and rival algorithms, following the commonly used training settings [170, 171], the threshold μ is determined by rejecting 10% of training samples as outliers.

6.4.1.5 Evaluation Metrics

In this chapter, the G-mean, F_1 score, and area under the curve (AUC) score are utilized as the measurements to assess the performance of algorithms.

$$\begin{aligned}
&\mathrm{F}_1 = \frac{2R \cdot P}{P+R}, \\
&\mathrm{G-mean} = \sqrt{R \cdot \frac{TN}{TN+FP}}, \\
&P = \frac{TP}{TP+FP}, \quad \text{and} \quad R = \frac{TP}{TP+FN},
\end{aligned} \tag{6.9}$$

Table 6.2 – Hyperparameter settings of the proposed method and the compared rival algorithms

Methods	Parameters
MP inverse methods	C = selected within $\{1.0^{-5}$, 1.0^{-3}, 1.0^{0}, 1.0^{3}, $1.0^{5}\}$, HN = grid searched within $\{1000, 2000, 3000\}$, AE = 3, the kernel function in OC-K [25] = $e^{\frac{\\|x_i - x_j\\|}{2\sigma^2}}$, and outlier rate = 10%.
Traditional methods	BS = 100, ILR = grid searched within $\{1.0^{-2}$, 1.0^{-3}, $1.0^{-4}\}$, TE = 8, DR = 0.1 per 10 epochs, HN = grid searched within $\{1000, 2000, 3000\}$, AE = 3, the corruption rate in DAE^{G} and DAE^{B} [101] = 0.5, and the outlier rate = 10%.
The MCOC methods	C = selected within $\{1.0^{-5}$, 1.0^{-3}, 1.0^{0}, 1.0^{3}, $1.0^{5}\}$, HN = grid searched within $\{1000, 2000, 5000\}$, AE = 3, maximum iteration K = 50, tolerance error $\eta = 1.0^{-5}$, and outlier rate = 10%.
OC-HSNN	C = selected within $\{1.0^{-5}$, 1.0^{-3}, 1.0^{0}, 1.0^{3}, $1.0^{5}\}$, nodes in S-node = 500, nodes in RS-node = 500, L = 5, and outlier rate = 10%.
MCOC-HSNN	C = selected within $\{1.0^{-5}$, 1.0^{-3}, 1.0^{0}, 1.0^{3}, $1.0^{5}\}$, nodes in S-node = 500, nodes in RS-node = 500, L = 5, maximum iteration K = 50, tolerance error $\eta = 1.0^{-5}$, and outlier rate = 10%.

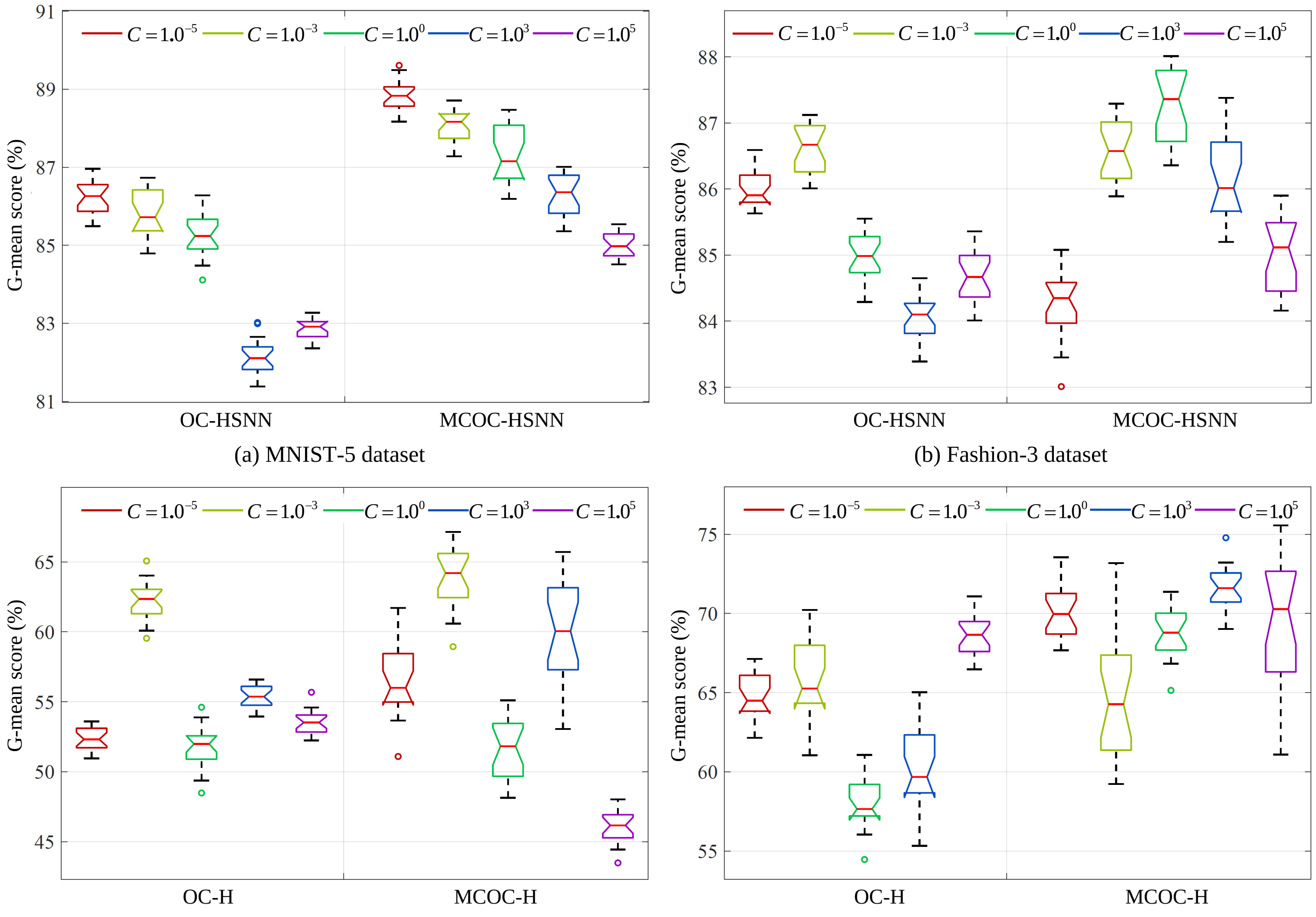
Notations: HN - the number of hidden layer nodes, AE - the number of autoencoders, BS - batch size, ILR - initial learning rate, TE - total number of retraining epochs, DR - decay rate, outlier rate - the μ is determined by rejecting 10% of training samples as outliers.

where P and R refer to the precision rate and recall rate. TP, TN, FP, and FN are true positive, true negative, false positive, and false negative, respectively. For the computation of AUC score, it measures the 2-dimensional area underneath the receiver operating characteristic (ROC) curve. For each algorithm and dataset, at least five trials are conducted.

6.4.2 Model Settings

Figure 6.2 shows the sensitive performance of the proposed methods and the rival algorithms as the hyperparameter C changes. The experiments were conducted 20 times for comprehensive comparison. It is observed that the C of the proposed OC-HSNN and MCOC-HSNN is not sensitive to the generalization performance. For example, as C changes, the average G-mean score of the proposed OC-HSNN on the MNIST-5 dataset ranges from 82.2% to 86.4%. However, the average G-mean score of the rival method, i.e., OC-H on the MNIST-5 dataset ranges from 52.5% to 62.2%.

To validate the intuition of the proposed MCOC-HSNN where only the subspace weight ($\mathbf{W}_i^r$) are calculated with MCC in MCOC-HSNN, a sanity check with a different combination of groups ranging from the purely MSE-based network learning (S_1) to the purely MCC-based optimization (S_4) is conducted. From Table 6.3, one can conclude the following: i) Compared to S_1 and S_2, the proposed MCOC-HSNN (S_3) provides a much superior performance in terms of the G-mean score. ii) Compared to S_4, the developed MCOC-HSNN (S_3) shows a similar generalization performance, but records just half of the training time. The reason is that the weight calculated by the S_4 strategy is inefficient as the generated latent space $\mathbf{W}_i^r$ using MCC is discriminative enough to classify the outliers from the target. There is no need to calculate the $\mathbf{W}^v$ and $\mathbf{P}$ through the fixed-point iteration algorithm. Thus, the merits of the proposed MCOC-HSNN are obvious.



(a) MNIST-5 dataset

(b) Fashion-3 dataset

(c) MNIST-5 dataset

(d) Fashion-3 dataset

Figure 6.2 – Sensitive performance of the proposed methods and rival methods on MNIST-5 and Fashion-3 datasets. The performance of the proposed methods is not sensitive to the hyper-parameter C. For example, with various C, the average G-mean score of the proposed OC-HSNN on MNIST-5 dataset ranges from 82.2% to 86.4%. However, the average G-mean score of OC-H on MNIST-5 dataset ranges from 52.5% to 62.2%.

Table 6.3 – Performance of the proposed method with various combinations [TRT: training time in minute, GM: G-mean scrore, S_1 - $\mathbf{W}^r$ and $\mathbf{W}^v$ are calculated w/ MSE; S_2 - $\mathbf{W}^r$ is calculated w/ MSE, while $\mathbf{W}^v$ is obtained w/ MCC; S_3 - $\mathbf{W}^r$ is calculated w/ MCC, while $\mathbf{W}^v$ is calculated w/ MSE; S_4 - $\mathbf{W}^r$ and $\mathbf{W}^v$ are calculated w/ MCC]

Methods	MNIST-4		NORB-1		Fashion-2	
	GM (%)	TRT (m)	GM (%)	TRT (m)	GM (%)	TRT (m)
S_1	66.9	5.7	81.5	4.6	75.5	5.6
S_2	71.6	10.5	81.7	7.6	77.6	9.2
S_3	73.1	10.2	**82.1**	7.5	**78.9**	8.4
S_4	**73.3**	19.6	82.0	15.4	**78.9**	17.7

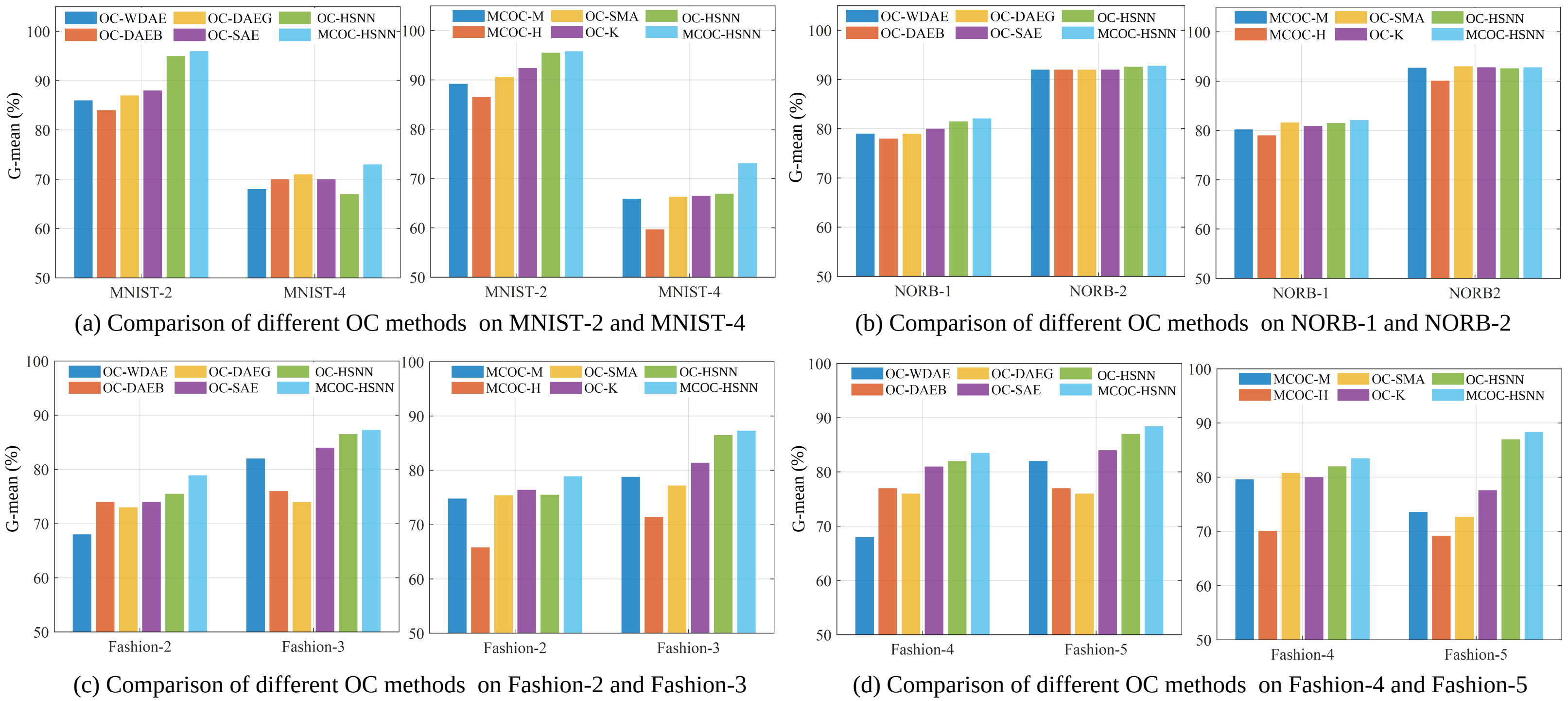
6.4.3 Analysis on Visual Classification Domain

6.4.3.1 Small-scale Datasets

In this evaluation, the raw images are loaded as the input. Table 6.4 and Fig. 6.3 show the comparison results with the other state-of-the-art OCC algorithms. Through observation, the following two conclusions can be drawn: Firstly, the proposed OC-HSNN and MCOC-HSNN outperform the existing MSE-based and MCC-based OCC algorithms on most of the small-scale datasets. For example, when compared to the best MCC-based results in the literature, the MCOC-HSNN boosts the performance by 2.7%, 5.2%, 7.2% and 10.9% on MNIST-4, MNIST-5, Fashion-3 and Fashion-5, respectively. Similarly, as for the MSE-based methods, the proposed OC-HSNN improves the G-mean score by 9.3% and 14.5% than the best MSE-based results in the literature on MNIST-5 and Fashion-5 datasets, respectively. Secondly, the MCOC-HSNN records a superior performance over the OC-HSNN algorithm from 0.3% to 2.5% improvement in G-mean score. Thus, the MCOC-HSNN is more powerful in processing small-scale datasets.

Table 6.4 – G-mean score comparison of the proposed and rival OCC algorithms. Values in **BOLDFACED** and UNDERLINED are the best and the second best results among the OCC methods

Methods	MNIST-2	MNIST-4	MNIST-5	NORB-1	NORB-2	NORB-3	Fashion-2	Fashion-3	Fashion-4	Fashion-5
OCC Algorithms Using MSE										
OC-S [169]	84.1±0.7	58.4±0.3	65.9±2.2	78.7±0.5	86.8±0.1	65.3±1.0	67.7±0.9	64.8±2.4	70.1±0.3	66.4±3.5
OC-K [25]	85.9±0.5	62.4±0.1	73.3±1.0	80.4±0.3	90.3±0.1	69.2±0.2	69.8±1.1	71.1±2.6	**82.6±0.3**	66.5±2.1
OC-M [42]	88.5±0.5	65.3±5.6	69.7±2.4	79.7±0.1	92.4±0.1	66.5±1.4	74.4±1.7	76.3±2.4	78.7±0.4	71.1±3.7
OC-H [98]	84.6±1.2	57.3±4.8	62.4±1.4	78.9±0.6	88.4±1.1	67.0±0.5	63.7±1.0	68.3±1.2	66.5±1.6	66.9±1.0
OC-SMA [172]	86.7±0.3	59.0±4.8	72.9±3.8	79.4±0.5	90.5±0.4	67.4±1.8	70.1±2.6	72.9±1.8	73.1±0.8	71.1±3.7
OC-WSI [161]	90.6±2.2	66.3±0.1	77.0±3.5	**81.6±0.1**	**92.9±0.1**	67.0±1.5	75.4±0.1	77.2±1.4	80.8±0.1	72.6±2.8
OC-HSNN	**94.5±0.9**	**66.8±0.1**	**86.3±0.9**	81.5±0.4	92.6±0.1	**68.9±0.1**	**75.5±0.2**	**86.5±0.7**	82.1±0.2	**87.1±1.3**
OCC Algorithms Using MCC										
MCOC-K [171]	88.4±1.4	62.7±0.1	74.6±1.7	81.3±0.5	90.0±0.1	69.0±1.0	72.4±0.2	79.9±0.6	**84.0±0.6**	75.4±2.4
MCOC-M [170]	89.2±2.1	65.8±3.3	74.2±1.3	80.2±0.4	92.7±0.5	66.8±1.9	74.8±0.4	78.8±1.4	79.6±0.5	73.5±1.3
MCOC-H [98]	86.5±1.4	59.7±3.1	64.5±2.0	79.0±0.3	90.1±0.7	67.1±1.0	65.8±0.1	71.4±1.3	70.1±1.1	69.3±3.0
MCOC-WSI [171]	92.2±0.7	70.4±0.1	83.6±1.1	**82.8±0.1**	**93.8±0.1**	68.1±0.5	77.4±0.1	80.1±0.8	81.4±0.3	77.5±1.7
MCOC-HSNN	**94.8±0.6**	**73.1±0.1**	**88.8±0.3**	82.1±0.3	93.6±0.1	**69.3±0.1**	**78.8±0.1**	**87.3±0.5**	83.5±0.4	**88.4±1.0**



(a) Comparison of different OC methods on MNIST-2 and MNIST-4

(b) Comparison of different OC methods on NORB-1 and NORB-2

(c) Comparison of different OC methods on Fashion-2 and Fashion-3

(d) Comparison of different OC methods on Fashion-4 and Fashion-5

Figure 6.3 – Performance of the proposed OC-HSNN and MCOC-HSNN with BP-based and MP inverse-based OCC algorithms. (a) and (b) show the performance comparison of OCC methods on MNIST-2/4 and NORB-1/2 datasets respectively. (c) and (d) are the results of different OCC strategies on Fashion-2/3/4/5 datasets.

Table 6.5 – AUC score comparison of the proposed and rival OCC algorithms on CIFAR-10 dataset. Values in **BOLD-FACED** and UNDERLINED are the best and the second best results among the OCC methods

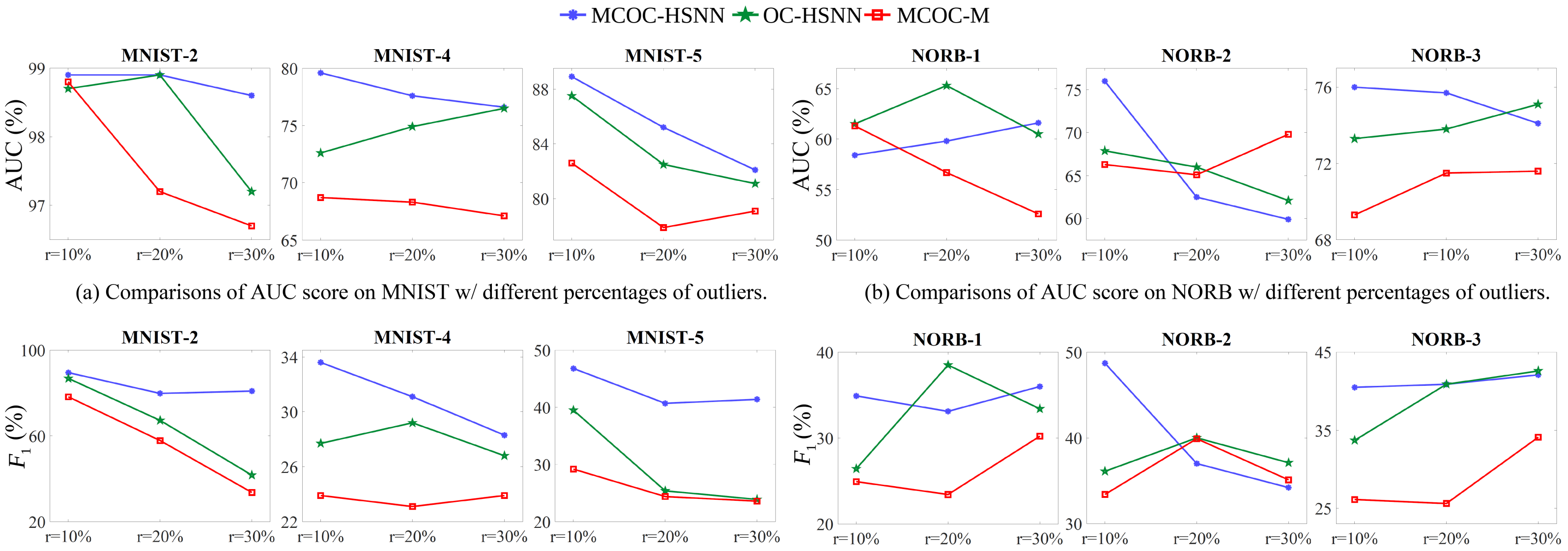
Datasets	Airplane	Automobile	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck	Mean
Inputs: The Raw Images											
OC-M [42]	60.3±2.7	63.5±0.5	57.7±1.9	58.4±0.9	65.9±0.6	62.0±0.4	65.2±1.6	66.8±2.2	74.6±1.3	71.2±1.6	64.6±1.4
OC-H [98]	59.6±2.2	60.2±0.9	57.3±1.3	58.2±1.2	61.7±1.0	60.3±0.7	64.6±0.3	65.4±1.8	68.2±1.1	73.3±1.9	62.9±1.2
MCOC-M [170]	65.2±1.5	70.3±1.4	58.4±1.1	58.7±0.4	70.8±1.9	65.3±2.2	76.3±1.1	70.4±0.6	77.9±0.5	77.4±1.3	69.2±1.2
AnoGAN [187]	67.1±2.5	54.7±3.4	52.9±3.0	54.5±1.9	65.1±3.2	60.3±2.6	58.5±1.4	62.5±0.8	75.8±4.1	66.5±2.8	61.8±2.6
DSVDD [188]	61.7±4.1	65.9±2.1	50.8±0.8	59.1±1.4	60.9±1.1	65.7±2.5	67.7±2.6	67.3±0.9	75.9±1.2	73.1±1.2	64.8±1.8
OC-HSNN	73.9±0.3	74.5±0.1	60.5±0.2	62.3±0.1	70.6±0.1	**69.0±0.4**	78.6±0.2	70.7±0.1	**81.3±0.0**	78.8±0.1	72.0±0.2
MCOC-HSNN	**75.9±0.2**	**74.2±0.1**	**63.1±0.1**	**63.9±0.3**	**71.1±0.3**	68.2±0.5	**79.7±0.0**	**71.0±0.1**	81.1±0.3	**79.4±0.2**	**72.8±0.2**
Inputs: High-level Features Extracted from Rotation Prediction [183]											
OC-M [42]	88.0±1.0	96.9±0.7	85.5±0.4	78.7±0.4	88.0±0.3	86.4±0.8	93.6±0.5	95.0±0.3	95.2±0.6	93.7±0.2	90.1±0.5
OC-H [98]	87.3±1.3	96.5±0.4	86.4±0.5	79.1±0.6	87.2±0.3	86.5±1.0	93.1±0.7	94.8±0.2	95.0±1.3	93.9±0.1	90.0±0.6
MCOC-M [170]	88.5±0.7	97.6±0.5	85.3±0.5	78.5±0.3	88.6±0.1	86.9±0.7	94.1±1.0	95.2±0.6	95.7±1.1	94.3±0.1	90.5±0.6
SSLKDE [189]	88.5±0.3	97.5±0.3	**88.2±0.3**	78.3±1.0	**90.2±0.2**	88.2±0.6	94.6±0.3	97.0±0.1	95.7±0.1	**94.9±0.1**	91.3±0.3
OC-HSNN	88.7±0.1	97.6±0.1	87.8±0.2	80.1±0.3	89.4±0.2	89.1±0.2	94.8±0.4	96.7±0.0	96.0±0.1	94.8±0.1	91.5±0.2
MCOC-HSNN	**89.4±0.2**	**97.7±0.1**	88.0±0.2	**80.2±0.2**	90.0±0.1	**89.7±0.1**	**95.2±0.4**	**97.3±0.0**	**96.3±0.1**	**94.9±0.1**	**91.8±0.2**

Table 6.5 shows the performance comparison of the proposed OCC algorithms with other state-of-the-art rival algorithms, including AnoGAN [187], DSVDD [188] and SSLKDE [189], on the CIFAR-10 datasets. It is worth noting that the best results of the OCC algorithms on each dataset are marked in boldface. Note that the experiments contain two different scenarios, i) algorithms trained with raw CIFAR-10 images, and ii) algorithms proceed with high-level features extracted from the rotation prediction [183]. For the first scenario, it is found that the proposed MCOC-HSNN records 2.5% to 14.2% AUC improvement compared to the DSVDD algorithm. As for the second scenario, the proposed MCOC-HSNN achieves 0.9%, 0.2%, 1.7%, 1.5%, 0.6% and 0.6% improvement in terms of AUC scores over the SSLKDE on classes airplane, automobile, cat, dog, frog and ship, respectively. When compared to the mean average AUC score, the MCOC-HSNN outmatches the SSLKDE by 0.5%, which is the best among these OCC algorithms. Hence, the effectiveness of the proposed algorithms is verified.

To further validate the robustness of the proposed OC-HSNN and MCOC-HSNN, we conducted a sanity analysis with different noise rates as shown in Fig. 6.4. The number of selected noise samples is set to be 10%, 20%, and 30% of the total number of samples of the target category, respectively. Through observation, we can conclude that the proposed OC-HSNN and MCOC-HSNN show better performance than the MCOC-M algorithm. For example, the mean average F_1 score of the proposed MCOC-HSNN with 10%, 20% and 30% outlier rate show 11.3%, 21.9%, and 47.3% improvement compared to the MCOC-M on MNIST-2 dataset, respectively.

6.4.3.2 Large-scale Datasets

The performance of the OC-M, MCOC-S, OC-HSNN and MCOC-HSNN wrt large-scale datasets are shown in Fig. 6.5 and Table 6.6. For a fair comparison, the high-level features extracted from Inception-V3 are utilized as the input to all the proposed and rival methods. Rather than simply comparing the algorithms with the G-mean score, the training time and the mean average inference time per frame are evaluated as well.



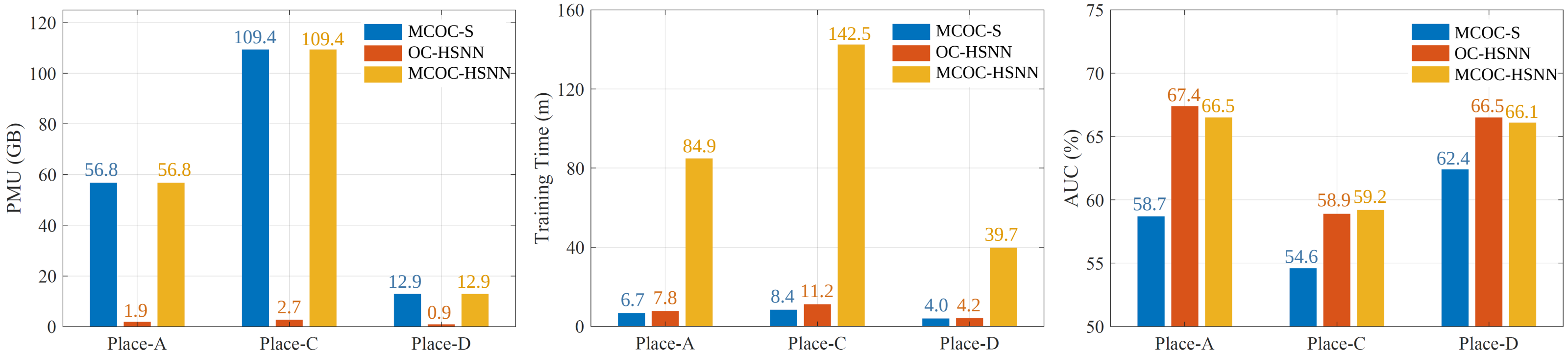
(a) Comparisons of AUC score on MNIST w/ different percentages of outliers.

(b) Comparisons of AUC score on NORB w/ different percentages of outliers.

(c) Comparisons of F_1 score on MNIST w/ different percentages of outliers.

(d) Comparisons of F_1 score on NORB w/ different percentages of outliers.

Figure 6.4 – Performance of the OC-HSNN, MCOC-HSNN, and MCOC-M wrt different percentage of outlier ratio. (a) and (b) are the AUC score of different methods, (c) and (d) are the comparison results in terms of the F_1 score.



(a) Comparison of peak memory usage on Place-A/C/D datasets

(b) Comparison of training time on Place-A/C/D datasets

(c) Comparison of AUC score on Place-A/C/D datasets

Figure 6.5 – Performance comparison of OC-HSNN, MCOC-HSNN and MCOC-S on Place-A/C/D datasets.

Table 6.6 – Comparison of OCC algorithms on large-scale datasets. Values in BOLDFACED are the best results among OCC methods [GM: G-mean score (%), TRT (m): training time in minute, TET (ms): mean average inference time per frame in millisecond]

Datasets	OC-M [42]			OC-H [98]			MCOC-M [170]			OC-HSNN			MCOC-HSNN		
	GM	TRT	TET	GM	TRT	TET	GM	TRT	TET	GM	TRT	TET	GM	TRT	TET
Place-A	40.2±1.4	8.5	1.6	43.7±1.5	9.6	1.7	40.6±1.3	144.8	1.5	53.5±0.5	9.4	1.3	**53.6±0.6**	162.9	1.3
Place-B	32.9±3.0	12.8	1.6	35.6±2.1	15.7	1.6	33.5±2.4	304.7	1.3	**42.2±1.8**	14.8	1.4	41.9±1.7	337.5	1.3
Place-C	33.4±0.9	12.1	1.4	37.9±0.2	15.5	1.5	31.7±0.7	297.9	1.5	**43.8±0.4**	14.6	1.3	43.5±0.1	310.7	1.3
Place-D	42.7±0.2	4.8	1.5	42.2±0.1	5.9	1.5	42.8±0.3	76.2	1.4	54.6±0.2	6.6	1.2	**54.8±0.1**	80.6	1.2
Place-ABCD	25.9±1.5	44.6	1.4	27.8±1.2	50.1	1.7	ROM[1]	ROM[1]	ROM[1]	**39.1±0.8**	46.4	1.3	ROM[1]	ROM[1]	ROM[1]

[1] ROM: result not available due to the fact of out of memory

From Fig. 6.5 and Table 6.6, we observe that: i) The proposed OC-HSNN and MCOC-HSNN have similar (roughly 10% more) training complexity compared to the OC-M and MCOC-M, respectively. Although the proposed methods need extra time for data feedback and weight calculation, the dimensionality of each subnetwork in OC-HSNN and MCOC-HSNN is small; ii) The proposed frameworks provide faster inference speed. This is because the major timing overhead of the proposed methods in the testing stage dwells in the feedforward process. The proposed methods have fewer parameters; iii) The OC-HSNN outmatches the comparison algorithms while providing a competitive performance compared to the MCOC-HSNN strategy. Although the MCOC-HSNN shows slightly better performance on the Place-A dataset, the algorithm needs around 12 times more training time over that of the OC-HSNN strategy. This is because the MCOC-HSNN utilizes a fixed-point scheme to calculate the subspace weight $\mathbf{W}_i^r$, causing more computational burden.

Furthermore, Fig. 6.5 (a) shows that the peak memory usage (PMU) of the proposed MCOC-HSNN is much higher than that of the OC-HSNN, as the dimension of $\mathbf{\Lambda}$ in Eq. (6.8) is very high. For example, for the Place-C dataset, the $\mathbf{\Lambda} \in \mathcal{R}^{165,394\times165,394}$ occupies around 109 GB or 218 GB memory in single-precision or double-precision array, respectively. Thus, the MCC-based strategies have stringent computational requirements for the large-scale dataset. Considering all the evaluation metrics including the training time and PMU, the OC-HSNN has a certain superiority over MCOC-HSNN on large-scale datasets.

6.4.4 Analysis on Extended Domains

6.4.4.1 Food Identification

Various OCC algorithms, including the MSE-based and the MCC-based schemes, are compared in Table 6.7. The proposed OC-HSNN and MCOC-HSNN outperform the other MSE-based and MCC-based algorithms from 3.8% to 9.0% in terms of G-mean score. The F_1 score and AUC of the proposed models are all higher than those of the comparison algorithms. Also, compared to MCOC-HSNN, the OC-HSNN achieves similar performance on Food+. For example, the OC-HSNN and MCOC-HSNN provide 50.3% and 50.1% of G-mean scores, respectively.

Table 6.7 – Comparison of OCC algorithms on extended domains [GM: G-mean score (%)]

Methods	Food+			CO-Mask		
	GM	F_1	AUC	GM	F_1	AUC
OCC Algorithms Using MSE						
OC-S [169]	41.3	39.6	57.2	78.8	40.6	85.2
OC-M [42]	42.4	44.0	60.1	79.4	43.7	89.6
OC-WSI [161]	46.1	43.0	61.0	81.1	50.4	91.1
OC-HSNN	**50.3**	**44.4**	**64.4**	**83.4**	**54.4**	**91.2**
OCC Algorithms Using MCC						
MCOC-S [170]	42.1	41.1	58.6	80.1	47.2	88.4
MCOC-M [170]	43.1	42.0	58.9	82.6	47.8	90.2
MCOC-WSI [171]	46.3	43.0	61.2	82.8	51.8	89.4
MCOC-HSNN	**50.1**	**44.4**	**64.0**	**87.8**	**54.6**	**94.8**

6.4.4.2 Misinformation Detection

To fully evaluate the performance of the OC-HSNN and MCOC-HSNN, one more experiment is carried out on the CO-Mask dataset. Experimental results in Table 6.7 show that i) the MCOC-HSNN provides the best performance among the OCC algorithms, and ii) the OC-HSNN shows superior performance over the existing OCC models, while having a very competitive performance compared to the MCOC-HSNN. For example, the OC-HSNN learning pipeline gains a 4.0% of more F_1 score than that of the OC-WSI framework, only 0.2% less of F_1 score compared to the MCOC-HSNN.

6.4.5 Discussion

6.4.5.1 OC-HSNN vs. MCOC-HSNN

The above outcomes demonstrate the following:

- The MCOC-HSNN achieves better generalization performance over OC-HSNN on small-scale datasets. The proposed MCC-based feature learning scheme is powerful in suppressing non-Gaussian noise and large outliers. Therefore, the MCOC-HSNN framework is capable of generating more robust low-dimensional encoding for the input data.

- On the other hand, compared to MCOC-HSNN, the OC-HSNN has the advantage of handling large-scale datasets. The benefits of the OC-HSNN are: Firstly, the OC-HSNN shows a competitive G-mean score compared to the MCOC-HSNN. Secondly, the subspace representations generated through the OC-HSNN framework are much faster than MCOC-HSNN since the coding matrix is updated with the analytic solution instead of the fixed-point iteration. Thirdly, the OC-HSNN demands a fewer amount of memory usage while the MCOC-HSNN needs access to industrial-scale computational resources to process big datasets.

6.4.5.2 Limitation

Although the proposed OC-HSNN and MCOC-HSNN show superior performance compared to the existing OCC algorithms, they still have the following two limitations: Firstly, the subspace representations are only optimized with the category information. However, in complex tasks, they are inefficient. One practical way is to combine the supervised and unsupervised learning simultaneously. Secondly, the structure is incapable of extracting high-level features as the models do not contain convolutional layers. In other words, when handling large-scale datasets like Place-365, the models need the feature extraction frameworks, such as VGG, ResNet, and Inception-V3, to extract the abstractive features.

6.5 Conclusion

To achieve a highly discriminative representation, a novel multilayer one-class classification algorithm OC-HSNN containing the subnetwork structure is proposed. Consequently, a robust framework called MCOC-HSNN using the maximum correntropy criterion for subspace feature learning is further developed. The contributions of this chapter are: i) The latent space features are encoded through subnetworks instead of conventional neurons; ii) Unlike the existing one-class networks that solve several sub-problems to approximate the global optimization, the proposed methods search the optimal representations and do the final one-class classification jointly; iii) Instead of simply applying maximum correntropy criterion in the final layer classification, in this chapter, the latent space encoding of MCOC-HSNN is generated with the use of maximum correntropy criterion; iv) A new dataset called CO-Mask is gathered.

Experimental results on 27 one-class classification datasets, ranging from MNIST-4 having 6,131 training samples to Place-ABCD containing 513,061 patterns, prove that the proposed OC-HSNN and MCOC-HSNN have promising performance when compared to the existing one-class classification algorithms. As for future direction, it would be worthwhile to develop a novel structure by considering unsupervised input details and supervised label information simultaneously.

Chapter 7

Semi-supervised Manifold Regularization via a Subnetwork-based Representation Learning Model

Semi-supervised classification with limited labelled training samples is a challenging task in the area of data mining. MP inverse-based manifold regularization (MR) is a widely used technique in tackling semi-supervised classification. However, most of the existing MP inverse-based MR algorithms can only generate loosely-connected feature encoding, which is generally less effective in data representation and feature learning. To alleviate this deficiency, we introduce a new semi-supervised hierarchical subnet neural network termed SS-HSNN. The key contributions of this model are as follows: 1) A novel MP inverse-based MR model using a subnetwork structure is introduced. The MR is utilized in the learning of the optimal MP inverse-based weights. 2) A new semi-supervised dataset called HFSWR-RDE is gathered. Experimental results on multiple domains show that the SS-HSNN achieves superior performance over the other semi-supervised learning algorithms, demonstrating fast inference speed and better generalization ability.

7.1 Introduction

The task of semi-supervised classification has attracted much attention in many real-world applications, such as video action recognition [190, 191], text mining [192], and remote image recognition [193, 194]. The collection of labelled data in these tasks is more difficult than the collection of unlabelled data, as labelled data incur more costs in terms of skilled human experts [195]. Taking medical diagnostics as an example, the collected data require not only expensive medical equipment, but also time-consuming analysis in terms of labelling by experts. Thus, it is crucial to devise efficient machines to learn discriminative features from a limited amount of labelled data and a large amount of unlabelled data.

The traditional semi-supervised learning algorithms can be divided into four categories: generative-based strategies [196, 197], self-training (ST) algorithms [198, 199], co-training (CT) methods [200, 201], and manifold regularization (MR) approaches [202, 203]. In general, semi-supervised algorithms need to pre-define assumptions associated with the distribution of the data $\mathcal{P}_X$ to a practical classification task [204]. For instance, the MR learning schemes assume that when the data points distribute on a low-dimensional manifold $\mathcal{M}$, and a classifier is smooth on this area, the nearby points $\mathbf{x}_1$ and $\mathbf{x}_2$ on $\mathcal{M}$ should have similar conditional probabilities $P(\mathbf{y}|\mathbf{x}_1)$ and $P(\mathbf{y}|\mathbf{x}_2)$. Recently, many approaches have been proposed to explore the potential of MR approaches [202, 203, 205, 206, 207, 208, 30, 209]. In this direction, the MP inverse-based method [203, 205, 208, 30, 209] is one of the widely adopted MR strategies for semi-supervised classification. Compared to the other MR algorithms, the MP inverse-based algorithm has the advantages of faster learning speed along with comparable performance.

Nonetheless, while the MP inverse-based MR approaches are capable of providing superior performance in semi-supervised learning tasks, they still face several pitfalls. First, due to the randomization mechanism, the number of hidden neurons in MR algorithms is usually very high. The researchers in [210] prove that an MP inverse-based network can achieve great generalization performance as long as the number of hidden layer neurons is large enough. However, this is inefficient because there are many redundant and meaningless features in such high-dimensional representations. Also, too many hidden layer units in one neural network may lead to high generalization error due to overfitting and high variance.

Second, most of the MP inverse-based MR frameworks encode features and find the final cognition with successive stacked learning blocks, which results in poor generalization performance. The subnet-based neural network, which offers a practical way to enhance a network's capacity to discriminate data representation, has already been applied in supervised classification and recognition, such as Wi-HSNN [173], OS-HSNN [211] and OC-HSNN. Thus, in this chapter, we propose a subnet-based multilayer RL framework for semi-supervised classification.

Driven by the motivations mentioned above, we propose a new MP inverse-based MR paradigm: a semi-supervised hierarchical subnet-based neural network (SS-HSNN) that can generate highly discriminative global representations for semi-supervised classification. Compared with the traditional semi-supervised learning algorithms, this structure provides competitive generalization performance based on a one-step training strategy. The contributions of this chapter are threefold:

1. **A novel MP inverse-based MR network.** A subnet-based framework called SS-HSNN is proposed for semi-supervised classification.

2. **A new dataset.** A new vessel target detection dataset called HFSWR-RDE has been prepared in this chapter. The HFSWR-RDE dataset is a challenging but comprehensive dataset for semi-supervised object detection.

3. **Comprehensive comparison.** A set of experiments was conducted on multiple MP inverse-based MR approaches over different datasets, which included image classification, text categorization, and radar target detection.

7.2 Related Works on Semi-supervised Classification

A semi-supervised paradigm serves as an extension of supervised and unsupervised learning that makes use of a small amount of labelled data along with a large amount of unlabelled data. Essentially, semi-supervised classification approaches can be divided into four families: generative methods, ST methods, CF methods, and MR methods. The generative methods [196, 197, 212] assume a mathematical model $P(x, y) = P(y)P(x|y)$, where $P(x|y)$ is the mixture distribution estimated by the unlabelled data. For the ST algorithms [198, 199, 213, 214], a classifier is first built using the labelled training samples. The trained classifier is utilized to generate the pseudo-label for the unlabelled training patterns. Then, the most confident unlabelled samples are appended to the labelled training set with the predicted tags. Following that, the classifier is retrained with the new labelled set, and the procedure is repeated [215]. The CT methods [200, 201, 216, 217] assume that two different feature views can provide complementary information on the input data. Hence, two separate classifiers with two different sets of features are first trained on the labelled samples. Then, the most confident results of each classifier on the unlabelled training samples are iteratively added to the training set to expand the labelled data [218].

In recent years, the MR algorithms [202, 203, 207, 208, 205, 206] have become a hotspot area for research in semi-supervised classification. Essentially, the MR methods can build a learning graph in which the vertices are the labelled and unlabelled training instances. The edge between the samples portrays the similarity of the two patterns. In particular, the MR algorithms assume that the labels of samples are smooth over the graph [215]. The Laplacian support vector machine (LapSVM) [202] builds its model upon the standard SVM structure, and it creates a specific kernel that defines similarity as a function of a mixture of geodesic and ambient distances [219]. The separating hyperplane of LapSVM in the feature space is the same as that of the traditional SVM framework, but the kernel is different: the SVM utilizes the labelled data to build the kernel, whereas the LapSVM defines a kernel that uses both labelled and unlabelled samples to reflect the intrinsic geometry. The Laplacian regularized least-squares (LapRLS) [202] bases its structure on the L_2 norm-based regression model and the intrinsic geometry of the Laplacian kernel. Huang et al. [203] combined MR with an MP inverse-based single-layer network and extended the structure [28] from a supervised domain to a semi-supervised domain. Based on [203], the work in [205] provides a multilayer structure that can efficiently handle semi-supervised classification tasks. Aside from utilizing the Laplacian kernel, recent studies have also extended the semi-supervised approach to other regularization techniques and learning systems, such as Hessian regularization [207, 208] and broad learning system (BLS) [30]. Zhao et al. [206] employed an advanced BLS [30] for semi-supervised classification, and the semi-supervised BLS achieved superior generalization performance when compared to other MP algorithms using different semi-supervised datasets. Additionally, the MR algorithms have also been investigated in many real-world applications. It has been verified that the MR technique is powerful for semi-supervised classification and usually facilitates state-of-the-art performance on a wide range of datasets, such as EEG signal-processing [205] and action-recognition [209].

Despite the successes of the above-mentioned algorithms, the accuracy and stability of the MR algorithms can be further improved. Most MR approaches focus on boosting generalization performance by increasing the number of hidden layers instead of optimizing the network's topology. For example, the study in [205] can be considered a multilayer version of a single-layer structure [203]. The only difference between these two models lies in the number of stacked hidden layers. In this chapter, we propose a novel MP inverse-based MR algorithm using a subnetwork structure for semi-supervised representation learning and classification.

7.3 The Proposed Algorithm

The annotations used in this chapter can be found in Table 4.1 and Table 7.1.

Table 7.1 – Notations to be used in this chapter

Notation	Meaning
Γ_l	the exit layer representation of the labelled data.
Γ_u	the exit layer representation of the unlabelled data.
H_l	the entrance layer representation of the labelled data.
H_u	the entrance layer representation of the unlabelled data.
Ψ_l	the refinement layer representation of the labelled data.
$\mathbf{L}$	the Laplacian matrix.
L	the total number of iteration.
l	the total number of labelled training samples.
$\mathbf{P}_i$	the error feedback data from the i-th iteration.
u	the total number of unlabelled samples.
$\mathbf{X}_l$	the labelled training data.
$\mathbf{X}_{tes}$	the testing data.
$\mathbf{X}_u$	the unlabelled training data.

7.3.1 Manifold Regularization

The MR assumes that both the labelled and unlabelled samples have the same distribution space $\mathcal{M}$, and the conditional probabilities of two nearby points $P(\mathbf{y}|\mathbf{x}_1)$ and $P(\mathbf{y}|\mathbf{x}_2)$ should be similar. Suppose we have few labelled data and a large number of unlabelled data: $\mathbf{X} = \{\mathbf{X}_l; \mathbf{X}_u\}$. We denote $\{\mathbf{X}_l, \mathbf{T}\}$ as the labelled data, and $\{\mathbf{X}_u\}$

as unlabelled patterns, l and u are the number of labelled and unlabelled samples, respectively. The MR algorithms minimize the following loss function to achieve the smoothness assumptions of the input data [202]:

$$L_{loss} = \frac{1}{2} \sum_{i}^{l+u} \sum_{j}^{l+u} w_{ij} ||P(\mathbf{y}|\mathbf{x}_i) - P(\mathbf{y}|\mathbf{x}_j)||^2, \tag{7.1}$$

where w_{ij} is the pairwise similarity between samples $\mathbf{x}_i$ and $\mathbf{x}_j$. In particular, the similarity weight w_{ij} is defined as the thermonuclear equation [203]:

$$w_{ij} = \begin{cases} \exp\left(-\frac{||\mathbf{x}_i - \mathbf{x}_j||^2}{2\delta^2}\right), & \mathbf{x}_i \in K(\mathbf{x}_i) \, \mathrm{or} \, \mathbf{x}_j \in K(\mathbf{x}_j) \\ 0, & \text{otherwise,} \end{cases} \tag{7.2}$$

where $K(x)$ denotes the set of k neighbors of point x. Eq. (7.1) can be approximated by the following expression:

$$\hat{L}_{loss} = \frac{1}{2} \sum_{i}^{l+u} \sum_{j}^{l+u} w_{ij} ||\hat{\mathbf{y}}_i - \hat{\mathbf{y}}_j||^2, \tag{7.3}$$

where $\hat{\mathbf{y}}_i$ and $\hat{\mathbf{y}}_j$ are the output of $\mathbf{x}_i$ and $\mathbf{x}_j$, respectively. It can be further simplified by the following:

$$\hat{L}_{loss} = Tr(\hat{\mathbf{Y}}\mathbf{L}\hat{\mathbf{Y}}) \tag{7.4}$$

where $Tr(\cdot)$ represents the trace of a matrix. The Laplacian matrix $\mathbf{L}$ is defined by $\mathbf{L} = \mathbf{D} - \mathbf{W}$, $\mathbf{D}$ is a diagonal matrix with its diagonal elements $\mathbf{D}_{ii} = \sum_{j}^{l+u} w_{ij}$, $\mathbf{W}$ is a matrix formed with w_{ij}, and the Laplacian matrix $\mathbf{L}$ can be represented by $\mathbf{D}^{-1/2}(\mathbf{D} - \mathbf{W})\mathbf{D}^{-1/2}$.

7.3.2 Preliminary

Huang et al. [203] proposed a semi-supervised learning algorithm, which incorporates the Laplacian matrix into a single-layer structure to leverage the unlabelled data to boost the generalization performance in the case of sparse labelled patterns. It is formulated as follows [203].

$$\begin{aligned} \min \ & \frac{1}{2}||\mathbf{W}_2||^2 + \frac{C}{2}||\mathbf{E}||^2 + \frac{\lambda}{2}Tr(\mathbf{Y}^T\mathbf{L}\mathbf{Y}), \\ \text{s.t. } & \mathbf{E} = \mathbf{T} - g(\mathbf{X}_l, \mathbf{W}_1, \mathbf{b}_1) \cdot \mathbf{W}_2, \\ & \mathbf{Y} = g(\mathbf{X}, \mathbf{W}_1, \mathbf{b}_1) \cdot \mathbf{W}_2, \end{aligned} \tag{7.5}$$

where the Laplacian matrix $\mathbf{L}$ and the network output $\mathbf{Y}$ are built based on labelled and unlabelled patterns, $\mathbf{T}$ is the expected output, C is the regularization parameter, λ is the tradeoff parameter, and $\mathbf{W}_1$ and $\mathbf{W}_2$ are the hidden and output weights, respectively.

Similarly, in this chapter, the objective function of the proposed semi-supervised classification algorithm termed SS-HSNN is defined by Eq. (7.6).

$$\begin{aligned} \min \ & \frac{1}{2}||\mathbf{W}^v||^2 + \frac{C}{2}||\mathbf{E}||^2 + \frac{\lambda}{2}Tr(\mathbf{Y}^T\mathbf{L}\mathbf{Y}), \\ \text{s.t.} \, & \mathbf{E} = \mathbf{T} - \Gamma_l \mathbf{W}^v = \mathbf{T} - \sum_{i=1}^{L} g(\mathrm{H}_l, \mathbf{W}_i^r, \mathbf{b}_i^r) \cdot \mathbf{W}^v, \\ & \mathbf{Y} = \Gamma \mathbf{W}^v = \sum_{i=1} g(\mathrm{H}, \mathbf{W}_i^r, \mathbf{b}_i^r) \cdot \mathbf{W}^v, \end{aligned} \tag{7.6}$$

where $\Gamma = \{\Gamma_l; \Gamma_u\}$ is the global-level features with l labelled and u unlabelled data, $\mathrm{H} = \{\mathrm{H}_l; \mathrm{H}_u\}$ is the entrance layer features, L stands for the number of subnet nodes, ($\mathbf{W}_i^r$, $\mathbf{b}_i^r$) are the parameters of the i-th RS-node, $\mathbf{W}^v$ is the output layer weight, and $g(\cdot)$ is the activation function. Substitute the constraints into the objective equation, Eq. (7.6) is equivalent to minimize the following equation:

$$\frac{1}{2}||\mathbf{W}^v||^2 + \frac{1}{2}C||\mathbf{T} - \Gamma_l \mathbf{W}^v||^2 + \frac{\lambda}{2}Tr\left((\mathbf{W}^v)^T \Gamma^T \mathbf{L} \Gamma \mathbf{W}^v\right). \tag{7.7}$$

We set the gradient to zero:

$$\nabla L_{\mathrm{SS-HSNN}} = \mathbf{W}^v + \Gamma_l^T C(\mathbf{T} - \Gamma_l \mathbf{W}^v) + \lambda \Gamma^T \mathbf{L} \Gamma \mathbf{W}^v = 0. \tag{7.8}$$

The weight $\mathbf{W}^v$ of the SS-HSNN can be solved:

$$\begin{aligned} &\mathbf{W}^v = (I + C\Gamma_l^T\Gamma_l + \lambda\Gamma^T\mathbf{L}\Gamma)^{-1}\Gamma_l^T C\mathbf{T}, \\ &\Gamma = \sum_i^s g(\mathrm{H}, \mathbf{W}_i^r, \mathbf{b}_i^r), \quad \text{and } \Psi_l = \sum_i^s g(\mathrm{H}_l, \mathbf{W}_i^r, \mathbf{b}_i^r). \end{aligned} \tag{7.9}$$

Thus, we obtain the analytic solution of Eq. (7.6).

7.3.3 The Proposed SS-HSNN

The structure of the proposed algorithm is depicted as Fig. 7.1, which is similar to Wi-HSNN. The learning procedure of the proposed SS-HSNN is elaborated in Algorithm 7.1. Similarly, the learning steps have two stages: Stage 1 - Initialization stage, and Stage 2 - Iterative learning stage.

Compared with Wi-HSNN, the SS-HSNN have the following key characteristics: 1) The SS-HSNN is a more advanced RL algorithm for semi-supervised classification. By the use of the Laplacian matrix, the SS-HSNN is capable of handling the semi-supervised task efficiently. 2) The SS-HSNN uses the MR to optimize the refinement and exit layer weights and to pull back the error to the refinement layer ($\mathbf{P}_i$). By doing so, the subnet model can process the labelled and unlabelled data simultaneously.

7.4 Experimental Results

7.4.1 Experimental Setup

7.4.1.1 The Environment

All of the experiments conducted in this study were tested using MATLAB 2020b on a workstation with a 256 GB memory and 2.8 GHz E5-2650 processor.

Algorithm 7.1 The proposed SS-HSNN

```
Inputs: The concatenated super-state feature vector $(\mathbf{X}, \mathbf{T})$
Outputs: Class label $\mathbf{Y}$

• Stage 1: Initialization Stage $(i = 1)$
Procedure I: $\mathbf{W}_1^f$, $\mathbf{W}_1^r$, $\mathbf{W}_1^v$ $\Gamma_1$ and $\Gamma \leftarrow$ Initialization
Procedure II: $\mathbf{W}_f \leftarrow$ Weights Learning $(\Gamma_l, \Gamma, \mathbf{T})$
Procedure III: $\mathbf{E}_1$ and $\mathbf{P}_1 \leftarrow$ Error Backpropagation $(\mathbf{T}, \Gamma_l, \mathbf{W}_1^v)$
• Stage 2: Iterative Learning Stage $(2 \leq i \leq M)$ % Iteratively increase the number of subspace
for each iteration $i$ do
    $\mathbf{W}_i^f \leftarrow$ Assign randomly % The entrance layer weight.
    $\mathrm{H}_{li} = g(\mathbf{W}_i^f \cdot \mathbf{X}_l + \mathbf{b}_f)$, $\mathrm{H}_i = g(\mathbf{W}_i^f \cdot \mathbf{X} + \mathbf{b}_f)$ % The entrance layer feature, $\mathrm{H}_{li}$ and $\mathrm{H}_i$ are the features of the labelled data and all the data, respectively.
    Procedure III: $\mathbf{W}_i^r \leftarrow$ Weights Learning $(\mathrm{H}_{li}, \mathrm{H}_i, \mathbf{P}_{i-1})$ % The refinement layer weight.
    $\Psi_{li} = g(\mathbf{W}_i^r \cdot \mathrm{H}_{li} + \mathbf{b}_r)$, $\Psi_i = g(\mathbf{W}_i^r \cdot \mathrm{H}_i + \mathbf{b}_r)$ % The refinement layer feature, $\Psi_{li}$ and $\Psi_i$ are the features of the labelled data and all the data, respectively.
    $\Gamma_l = \sum_j^i \Psi_{lj}$, $\Gamma = \sum_j^i \Psi_j$ % Update the global-level representation.
    Procedure III: $\mathbf{W}_i^v \leftarrow$ Weights Learning $(\Gamma_l, \Gamma, \mathbf{T})$
    Procedure II: $\mathbf{E}_i$ and $\mathbf{P}_i \leftarrow$ Error Backpropagation $(\mathbf{T}, \Gamma_l, \mathbf{W}_i^v)$
end for
$\mathbf{Y} = \Gamma \cdot \mathbf{W}_i^v$     % $\Gamma$ is the global-level representation, $\mathbf{Y}$ is the label.
```

```
Procedure I: Initialization
Inputs: NA
Outputs: $\mathbf{W}_1^f$, $\mathbf{W}_1^r$, $\mathbf{W}_1^v$, $\Gamma_l$ and $\Gamma$

$\mathbf{W}_1^f \leftarrow$ Assign randomly     % Assign randomly with $\mu$=0, $\sigma$=0.01 of Gaussian distribution
$\mathrm{H}_{l1} = g(\mathbf{W}_1^f \cdot \mathbf{X}_l + \mathbf{b}_1^f)$ and $\mathrm{H}_1 = g(\mathbf{W}_1^f \cdot \mathbf{X} + \mathbf{b}_1^f)$
$\mathbf{W}_1^r \leftarrow$ Assign randomly     % Assign randomly with $\mu$=0, $\sigma$=0.01 of Gaussian distribution
$\Psi_{l1} = g(\mathbf{W}_1^r \cdot \mathrm{H}_{l1} + \mathbf{b}_1^r)$ and $\Psi_1 = g(\mathbf{W}_1^r \cdot \mathrm{H}_1 + \mathbf{b}_1^r)$
$\Gamma_l = \Psi_{l1}$ and $\Gamma = \Psi_1$
Procedure III: $\mathbf{W}_1^v \leftarrow$ Weights Learning $(\Gamma_l, \Gamma, \mathbf{T})$
return $\mathbf{W}_1^f$, $\mathbf{W}_1^r$, $\mathbf{W}_1^v$, $\Gamma_l$ and $\Gamma$
```

```
Procedure II: Weights Learning $(\Gamma_l, \Gamma, \mathbf{T})$
Inputs: $\Gamma_l$, $\Psi = \{\Gamma_l; \Gamma_u\}$, $\mathbf{T}$
Outputs: $\mathbf{W}$

$\mathbf{L} \leftarrow$ Construct via $\Gamma$ % Construct the Laplacian Matrix with both labelled and unlabelled features.
$C$, $\lambda$ % Choose tradeoff parameters
$\mathbf{W} = (I + C\Gamma_l^T\Gamma_l + \lambda\Gamma^T\mathbf{L}\Gamma)^{-1}\Gamma_l^T C\mathbf{T}$
return $\mathbf{W}$;
```

```
Procedure III: Error Backpropagation $(\mathbf{T}, \Gamma_l, \mathbf{W}_f)$
Inputs: $\mathbf{T}$, $\Gamma_l$, $\mathbf{W}_f$
Outputs: $\mathbf{E}_i$, $\mathbf{P}_i$

$\mathbf{E}_i = \mathbf{T} - \Gamma_l \cdot \mathbf{W}^v$
$\mathbf{P}_i = g^{-1}\left(\mathbf{E}^i \cdot \left(I + C(\mathbf{W}^v)^T\mathbf{W}^v + \lambda(\mathbf{W}^v)^T L\mathbf{W}^v\right)^{-1}(\mathbf{W}^v)^T C\right)$ %Pulling back the error
return $\mathbf{E}_i$ and $\mathbf{P}_i$
```

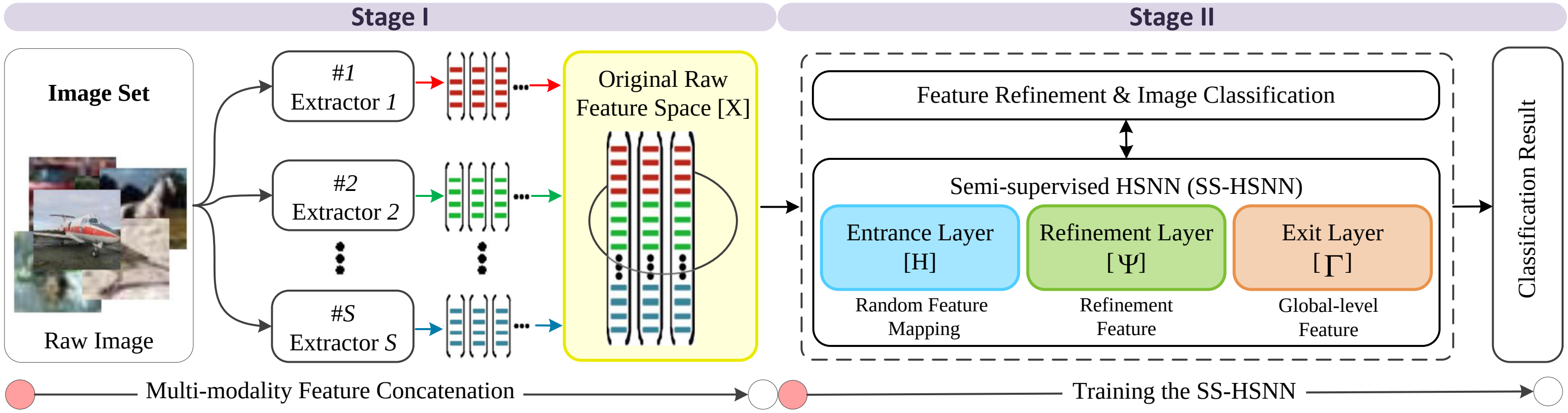


Figure 7.1 – The diagram of the proposed SS-HSNN. Similar to the Wi-HSNN, the SS-HSNN is composed of an input layer, an output layer, and three hidden layers. The first hidden layer is the entrance layer which contains L number of S-node. The second hidden layer called the refinement layer consists of L number of RS-node. The last hidden layer is the exit layer, which generates the global-level representation.

7.4.1.2 The Dataset

In this chapter, we performed experiments on 12 popular datasets, including image classification, text categorization, and vessel target detection. The details of each dataset are summarized in Table 7.2.

Table 7.2 – Summary of the datasets

Datasets	$\mathbf{X}_l$	$\mathbf{X}_u$	$\mathbf{X}_{tes}$	n
G50c	50	313	100	50
Coil-20	240	920	280	1,024
YaleB	418	1,520	476	1,024
Coil-100	1,200	4,600	1,400	1,024
Isolet	1,248	4,992	1,557	617
USPS	1,460	5,831	2,007	256
CIFAR-10	10,000	40,000	10,000	2,048
CIFAR-100	10,000	40,000	10,000	2,048
MNIST-1	1,000	9,000	60,000	784
MNIST	10,000	50,000	10,000	784
WeaRe	3,951	15,732	4,217	3,000
HFSWR-RDE	25	500	25	65,536

We employed six commonly used datasets, G50c, USPS, Coil-20, Coil-100, Isolet, and YaleB, and four complex datasets, CIFAR-10, CIFAR-100, MNIST-1, and MNIST, to evaluate the SS-HSNN. Most of the datasets are quite challenging for semi-supervised classification. For example, the Coil-100 dataset contains only 5,800 training samples but has 100 classification categories. It is worth noting that for the G50c, MNIST-1, and MNIST datasets, we followed the commonly used training settings [203, 205, 206]. As for the Coil-20, YaleB, Coil-100, Isolet, USPS, CIFAR-10, and CIFAR-100 datasets, 20% of training samples per category were collected as labelled samples, while the remaining 80% of the data were used as unlabelled training samples. Each of the datasets was split into three parts, two of which were used for training (denoted by $\mathbf{X}_l$ and $\mathbf{X}_u$), and the rest ($\mathbf{X}_{tes}$) were used for testing.

For the text categorization dataset, we utilized one weather report dataset, WeaRe[1], to fully verify the proposed method. The WeaRe dataset consists of 23,900 training

[1] The dataset can be download at here.

samples from 39 learning objects. The training set contains 19,683 patterns, while the testing set has 4,217 patterns. In this dataset, 20% of training texts are labelled, with the remaining training patterns being unlabelled. The embeddings (300-dimensional vectors) of each text pattern were obtained via fastText[2], a pretrained word embedding dictionary containing one million English words that was created by Facebook's AI Research (FAIR) lab. In this chapter, the number of word vectors in the sequence is set at 10. Thus, the dimension of each text vector is 3,000.

As for the vessel target detection dataset, we created a new dataset called HFSWR-RDE, which is an extension of HF-radar [148]. HF-radar is a vessel detection dataset collected from high-frequency surface-wave radar (HFSWR) located on the coast of Bohai Bay, China. In a shore-based HFSWR system, the receiving array detectors are lined up in an array spaced at a homogeneity interval. The sea waves generate resonance scattering that can easily cause interference with vessel signal transmissions. Thus, it is very difficult to recognize vessels against such a complex background. In addition, in practical HFSWR monitoring, not all of the images have accurate ground truth. In other words, the available labelled training samples are very limited. As a result, the traditional object detection algorithms easily run into an over-fitting problem. In this chapter, to verify the effectiveness of the proposed SS-HSNN, we extended the existing dataset with hundreds of unlabelled HFSWR samples, forming a new radar dataset (HFSWR-RDE) with both labelled and unlabelled radar images.

7.4.1.3 The Rival Methods

In this chapter, we will make comparisons with several state-of-the-art MR-based semi-supervised learning algorithms [205, 206, 208], ST-based semi-supervised classification strategies [198, 220, 214], subnet-based classification algorithms [1, 114, 23], and vessel detection algorithms for radar [154, 153, 148].

MP inverse-based MR algorithms These methods include the semi-supervised single-layer Moore-Penrose inverse-based algorithm (SS-SMP) [203], the semi-supervised hierarchical network (SS-HMP) [205], the semi-supervised broad learning system net-

[2] FastText can be download at here.

work (SS-BLS) [206], and the Laplacian-Hessian regularization-based hierarchical model (LHR-HMP) [208].

ST-based algorithms The self-training algorithms, such as strong constraint self-training (SCST) [198], decision tree-based self-training (DTST) [220], and confidence regularized self-training (CRST) [214], were compared with the proposed strategy.

Subnet-based algorithms The state-of-the-art subnet-based algorithms were used for comparison. These algorithms include the single-layer subnet-based neural network (SSNN) [1], the two-layer subnet-based neural network (TSNN) [114], and the hierarchical subnet-based neural network (HSNN) [23]. Notably, these methods are designed for supervised learning. For a fair comparison, the above-mentioned algorithms utilized the proposed MP inverse-based solution to optimize weights, and thus these strategies can be implemented for semi-supervised learning. This chapter uses the prefix "SS" (SS-SSNN, SS-TSNN, and SS-HSNN) to refer to these algorithms using this procedure.

To conduct comprehensive evaluations, raw images or high-level features extracted from an ImageNet pre-trained ResNet [88] were fed to the proposed SS-HSNN as well as all the comparison algorithms as the input. In particular, for the G50c, Coil-20, Coil-100, YaleB, Isolet, USPS, MNIST-1, and MNIST datasets, we loaded with raw images. For CIFAR-10 and CIFAR-100, we used the 2048-dimensional features extracted from the "fc1000" layer of ResNet as the input. Note that high-level features are the data extracted from the pre-trained ResNet without fine-tuning. These features, therefore, only contain innate priors regarding ImageNet.

7.4.1.4 Configurations of The Semi-supervised Methods

The detailed hyperparameter settings of the proposed work and the compared rival methods are summarized as follows.

For the MP inverse-based MR algorithms, the hidden layer nodes and the regularized parameters C and λ were optimized within the grid $\{1000, 2000, 3000, 5000\} \times \{10^{-3}, 10^{0}, 10^{3}\}$. The number of hidden layers was set to 3. The regularized parameters of the last classification layer were optimized within grid $\{10^{0}, 10^{3}, 10^{5}\}$.

The activation function of the enhancement layer in SS-BLS was sigmoid. For the ST-based algorithms, the total number of training epochs and the mini-batch size were both set to 100, and the initial learning rate was 0.01, with a 0.1 attenuation rate for every 20 epochs. In SCST, the threshold value T and t were set to 0.999 and 0.001, respectively. As for the subnet-based rival algorithms as well as the proposed SS-HSNN, the number of subnets in each algorithm was set to 5, and each subnet contains 500 hidden neurons. The C and λ were optimized within the grid $\{10^{-3}, 10^{0}, 10^{3}\}$, respectively.

7.4.1.5 Evaluation Metrics

In this chapter, for the semi-supervised classification and text categorization domains, we utilized the Top-1 testing accuracy, training time, and inference time to evaluate the performance of each algorithm. To comprehensively verify the performance of the proposed SS-HSNN, the recorded results shown in this chapter are the mean average of a minimum of three experiments. As for the radar image vessel target detection task, vessel target detection probability, i.e., true positive rate (P_d), false-alarm probability (P_f), missing probability (M_r), and error probability (E_r) were used. In particular, those evaluation indexes are defined by the following:

$$\begin{aligned} &P_d = \frac{TP}{(TP + FN)}, \quad P_f = \frac{FP}{(FP + TP)}, \\ &M_r = 1 - P_d, \text{ and } \quad E_r = P_f + M_r, \end{aligned} \tag{7.10}$$

where TP stands for true positive, FP refers to false positive, and FN is the false negative.

7.4.2 Model Settings

In this subsection, we have the sensitive analysis of the SS-HSNN on the dimension of each subnetwork d and the choice of iteration L.

As can be seen from Table 7.3, with the increase of neuron nodes in each subnet, the testing accuracy of the SS-HSNN shows a trend of first increasing and then fluctuating. The experimental results show that if the proposed SS-HSNN needs to maintain high generalization performance, at least 200 to 400 neuron nodes are required in each subnet. On the other hand, it seems intuitive that when the number of iterations is fixed, the greater the number of neurons in each subnet, the longer the training time and inference time for SS-HSNN to process one dataset. To be consistent and shorten the processing time, the number of nodes in each subnet d was set to 500.

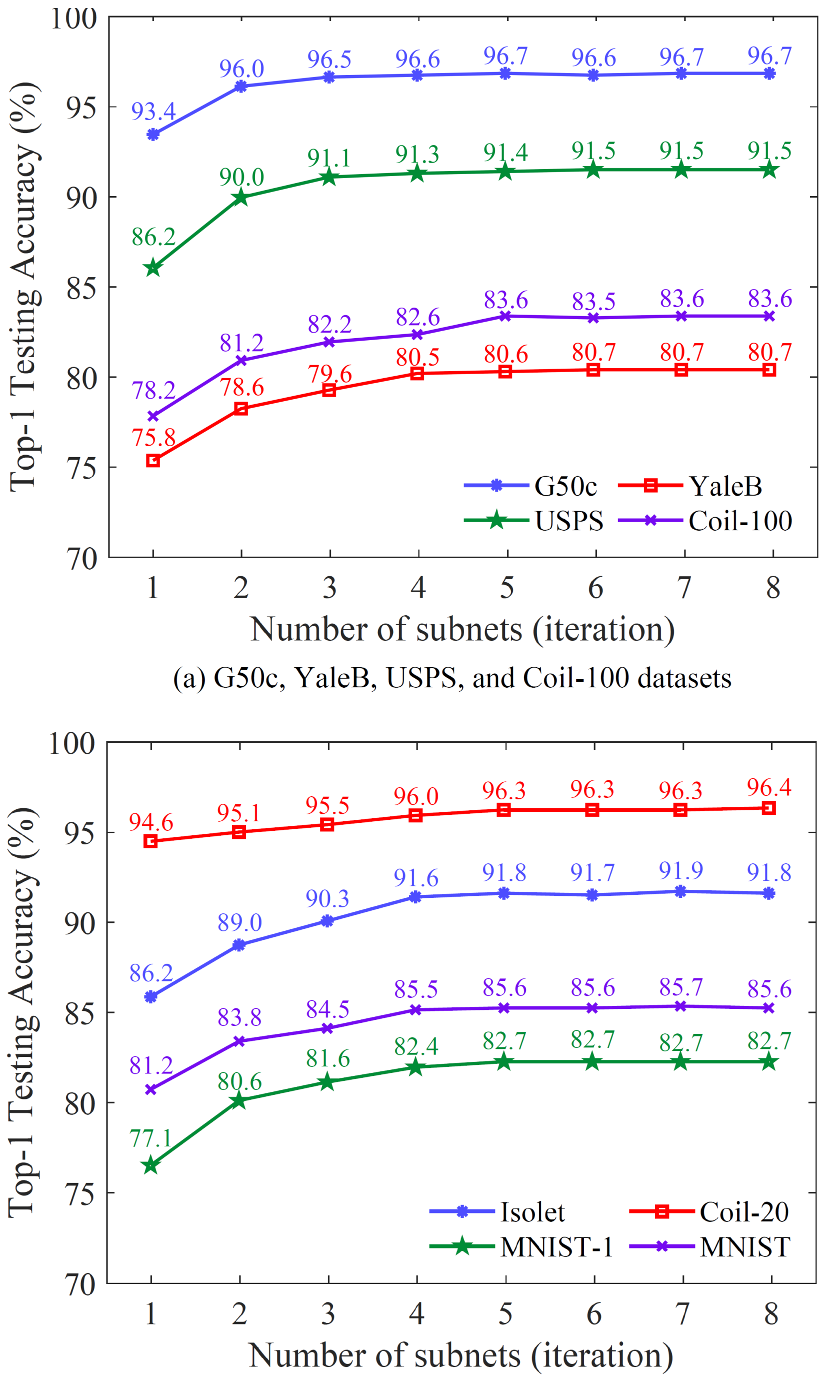
Table 7.3 – Top-1 testing accuracy of SS-HSNN with the increase of neurons in each subnet.

Dataset	100	300	400	500	600	800	1000
G50c	89.7	95.6	96.6	95.9	96.2	96.2	96.0
Coil-20	91.5	96.2	95.5	96.1	95.6	95.6	95.7
Coil-100	79.7	82.7	83.3	83.5	83.1	83.2	83.2
USPS	85.5	91.0	90.4	90.9	91.4	90.6	90.4

To find the effect of the number of subnets L, we carried out extensive experiments on the image classification datasets, and the results are shown in Fig. 7.2. The number of neurons in each subnet was set to 500. It is easily concluded that the testing accuracy of the proposed algorithm converges after $s = 4$. To ensure the structure is well-trained, it is recommended that L should be set to 5 by default.

7.4.3 Analysis on Image Classification Domain

Table 7.4 shows the comparison results of the proposed method and several state-of-the-art semi-supervised learning algorithms on the image classification task. The best result corresponding to each dataset is shown in boldface. We observed that SS-HSNN outperformed its competitors on 6 out of 8 datasets. In particular, the best results among the comparison algorithms on USPS, Coil-20, Isolet, YaleB, and MNIST are 91.2%, 95.5%, 90.4%, 80.1%, and 85.7%, respectively. The benefits of the proposed SS-HSNN are significant. The best-recorded results of the proposed SS-HSNN are 91.9%, 96.4%, 91.8%, 80.6%, and 87.3%, respectively, providing 0.4% to 1.5% improvement on these datasets. As for the overall mean performance, the proposed SS-HSNN exhibits robustness with an average overall performance of 88.8%. The overall improvements of the proposed model are 2.3%, 4.3%, 2.8%, 3.9%, 3.8%,



(a) G50c, YaleB, USPS, and Coil-100 datasets

(b) Isolet, Coil-20, MNIST-1, and MNIST datasets

Figure 7.2 – Top-1 testing accuracy with the increase of subnets.

Table 7.4 – Top-1 testing accuracy comparison of the SS-MSNN and other state-of-the-art semi-supervised learning methods.

Method	G50c (%)	USPS (%)	Coil-20 (%)	Coil-100 (%)	Isolet (%)	YaleB (%)	MNIST-1 (%)	MNIST (%)	Average (%)
The Comparison Algorithms									
SS-HMP [205]	95.9	89.5	94.5	80.9	90.2	79.4	79.0	83.6	86.6
LHR-HMP [208]	95.2	87.8	90.2	78.4	88.3	78.2	77.8	81.0	84.6
SS-BLS [206]	96.2	88.7	94.4	79.5	89.4	80.1	77.5	82.5	86.1
SCST [198]	93.8	89.2	92.2	78.1	89.3	79.2	76.1	82.4	85.0
DTST [220]	94.7	88.4	93.7	77.5	87.4	78.5	76.5	83.1	85.1
CRST [214]	96.5	90.4	94.7	**84.2**	89.7	79.6	80.7	84.8	87.6
SS-SSNN[1] [1]	95.5	87.8	92.3	79.4	86.9	76.3	77.1	83.5	84.9
SS-TSNN[1] [114]	96.3	89.2	94.8	80.3	90.1	79.6	78.5	85.0	86.7
SS-MSNN[1] [23]	**97.1**	91.2	95.5	80.9	90.4	79.1	80.2	85.7	87.5
The Proposed Algorithm									
SS-HSNN	96.6	**91.9**	**96.4**	83.6	**91.8**	**80.6**	**82.8**	**87.3**	**88.9**

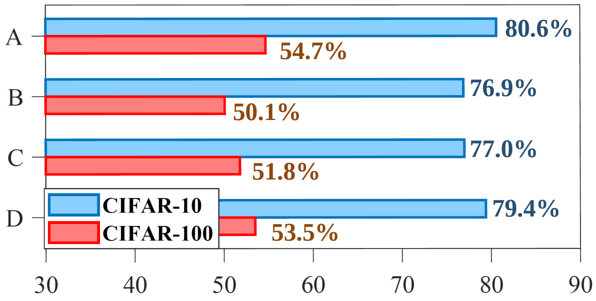
[1] The original SSNN [1], TSNN [114], and MSNN [23] are proposed for supervised learning. For fair comparison, the above-mentioned algorithms are conducted with semi-supervised solution proposed in this study. They are denoted as SS-SSNN, SS-TSNN, and SS-TSNN respectively.

1.3%, 4.0%, 2.2%, and 1.4% compared to SS-HMP [205], LHR-HMP [208], SS-BLS [206], SCST [198], DTST [220], CRST [214], SS-SSNN [1], SS-TSNN [114], and SS-MSNN [23], respectively. A possible reason for this better performance could be that the newly generated feature subspace is searched via the current error term in SS-HSNN, thus helping the structure to dynamically adjust the latent feature space and offset the error item.

In addition, Table 7.5 shows the comparison results of the semi-supervised algorithms on the CIFAR-10 and CIFAR-100 datasets. The results are consistent with the observations in the aforementioned experiments. As expected, SS-HSNN outperformed the SS-SMP on both datasets. In particular, the proposed method achieved both 1.2% increments of Top-1 testing accuracy over the existing hierarchical subnet-based framework on CIFAR-10 and CIFAR-100 datasets, while showing a 3.7% and 4.6% boost compared to SS-SMP. Therefore, the proposed algorithm is an effective technique for handling semi-supervised tasks.

Table 7.5 – Comparison with other semi-supervised algorithms on CIFAR-10 and CIFAR-100 (%).

Methods	Datasets	
	CIFAR-10	CIFAR-100
A: SS-SMP	76.9	50.1
B: SS-SSNN	77.0	51.8
C: SS-MSNN	79.4	53.5
D: SS-HSNN	**80.6**	**54.7**



7.4.4 Analysis on Extended Domains

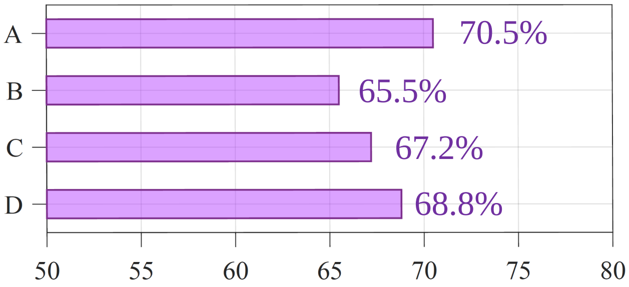
This chapter does not just evaluate the proposed SS-HSNN on the semi-supervised image classification domain; it also employs comparisons on two real-world application domains: text-pattern classification and HFSWR vessel target detection.

7.4.4.1 Text-pattern Categorization

Text categorization is the task of deciding whether a piece of text belongs to any of a set of pre-specified categories. Table 7.2 shows the details of the text dataset WeaRe. In this study, the embeddings of each sequence pattern were fed directly as input to verify the categorization capacity of each comparison algorithm. The classification performance of the WeaRe dataset is reported in Table 7.6. The proposed SS-HSNN achieved a testing accuracy of 70.5% on this dataset, which is 4.0%, 3.3%, and 1.7% better than those of SS-SMP, SS-SSNN, and SS-HSNN, respectively.

Table 7.6 – Comparison with other semi-supervised algorithms on WeaRe text categorization (%).

Methods	Top-1 accuracy
A: SS-SMP	65.5
B: SS-SSNN	67.2
C: SS-HSNN	68.8
D: SS-MSNN	**70.5**



7.4.4.2 HFSWR vessel target Detection

The traditional range-Doppler (RD) image contains the following components: dead zone, sea clutter, ionospheric clutter, background, and vessel targets. However, in HFSWR, the available samples are limited, and only a few RD images have accurate ground truth that can be used for training. HFSWR-RD [148] is an RD image dataset derived from an HFSWR located on the coast of Bohai Bay in China. It contains 50 RD images; half of the images were used for training while the rest were used for testing. In order to facilitate research on vessel target detection with radar and evaluate SS-HSNN against other semi-supervised algorithms in a practical application domain, we include an extra 500 unlabelled RD images into our original HF-radar dataset, thus providing a more challenging but comprehensive RD radar image dataset: HFSWR-RDE. The basic details of HFSWR-RDE are summarized in Table 7.2.

The state-of-the-art HFSWR ship-target detection algorithms include the least-square-based methods [148], wavelet transform (WT)-based methods [152, 153], and constant false-alarm rate (CFAR)-based methods [154, 155]. Rather than only making comparisons with traditional radar-processing algorithms, several semi-supervised MR frameworks including SS-MSNN [23] and SS-HMP [205] were also evaluated. To obtain fair experimental results, the pre-processing steps for SS-MSNN [23], SS-HMP [205], and the proposed SS-HSNN were the same as [148], which were extracted from a Haar-like descriptor. The detection results for the HFSWR-RDE dataset are shown in Table 7.7. As can be seen from Table 7.7, the proposed SS-HSNN provides superior performance, whereas the P_d of SS-HSNN exceeds that of SS-MSNN [23] by around 1%. Meanwhile, the P_f, M_r, and E_r of this study are all smaller or equal to those of the other methods.

Table 7.7 – RD image vessel target detection results.

Method	P_d (%)	P_f (%)	M_r (%)	E_r (%)
Regression method [148]	92.6	5.8	7.4	13.2
CFAR method [154]	85.4	13.4	14.6	28.0
Wavelet method [153]	90.3	8.3	9.7	18.0
SS-HMP [205]	92.8	6.5	7.2	13.7
SS-MSNN [23]	93.4	6.0	6.6	12.6
SS-HSNN	**94.4**	**5.6**	**5.6**	**11.2**

7.4.5 Timing Analysis

The average training time and inference time of SS-HSNN and other state-of-the-art comparison methods, including SS-HMP [205], SS-TSNN [114], and SS-MSNN [23], are tabulated in Table 7.8. The SS-HMP strategy takes the shortest amount of time for training, whereas the other subnet-based algorithms need more processing time, as the subnet structures need time to pull back the error. Surprisingly, SS-HSNN only requires about half of the training time of SS-TSNN and one-tenth of the computational time as of SS-HSNN. The reason for this is: the existing multilayer subnet-based algorithms, including SS-TSNN and SS-HSNN, still need to stack multiple learning blocks to find the discriminative encoding. For example, SS-TSNN constructs the multilayer structure with hundreds of two-layer subnets; similarly, the SS-HSNN framework uses hundreds of bottom layers to broaden the structure, and each bottom layer consists of a two-layer subnet structure. However, the proposed SS-HSNN serves as a one-step training structure that learns the latent space representations with a single model. Furthermore, the mean average testing time per frame is also recorded in Table 7.8. We found that the proposed SS-HSNN provides an excellent inference speed. Thus, the efficiency of the proposed model is verified.

Table 7.8 – Comparison of processing time: Tr. (s) the training time in second, Te. (ms) is the mean average testing time per frame in millisecond.

Datasets	SS-HMP [205]		SS-TSNN [203]		SS-MSNN [23]		SS-HSNN	
	Tr.	Te.	Tr.	Te.	Tr.	Te.	Tr.	Te.
G50c	**0.5**	1.4	3.5	1.6	19.4	2.4	2.0	**1.4**
Coil-20	**1.4**	1.4	22.8	1.7	128.5	2.8	14.6	**1.4**
Coil-100	**31.8**	1.4	452.1	1.5	2694.1	2.7	284.2	**1.4**
USPS	**45.6**	1.4	624.4	1.6	3958.2	2.8	371.2	**1.5**

7.4.6 Effectiveness Analysis of SS-HSNN via VC Dimension

In this subsection, the effectiveness of the proposed SS-HSNN over the existing MP inverse-based networks is verified based on Vapnik–Chervonenkis (VC) dimension.

In statistical learning theory, the true risk (ϵ_{true}) of the model has a relationship with the network's performance on a known set of training data (ϵ_{emp}) and the distribution of the data (Φ), which is termed as Eq. (7.11).

$$\epsilon_{true} = \epsilon_{emp} + \Phi(h, t), \tag{7.11}$$

where h is the VC-dimension of the learning model, and t is the total number of training samples.

In most cases, the ϵ_{true} is not computable as the true risk requires the joint probability distribution function (Φ) for the input and output of the learning model, which is almost impossible to accurately obtain. On the other hand, the learning process of one model is driven by the optimization of the average of the error computed over all the training samples, i.e., ϵ_{emp}. In fact, the $\Phi(h, t)$ in Eq. (7.11) is the VC-confidence [221], which can be expressed by Eq. (7.12).

$$\Phi(h, t) = \sqrt{\frac{h\ln\big((2t/h) + 1\big) - \ln(\eta/4)}{t}}. \tag{7.12}$$

The Eq. (7.12) establishes, with probability $1 - \eta$, how close the ϵ_{emp} is to the ϵ_{true}. It is a probable measure of the confidence in the generalization capabilities of a model trained to perform a certain task [222]. Further, VC compression (h/t) is a ratio that relates to the expressive power of one learning structure. The smaller the VC compression (h/t), the better the theoretical guarantee of generalization. In other words, given several learning networks and a sufficiently small η, it is best to choose the structure that minimizes the VC dimension h and $\Phi(h, t)$, as this network gives the lowest upper bound on the true error ϵ_{true}.

It has been shown that the lower bound on the VC dimension of a neural network is in the order $h = \Omega(\mathbf{W}^2)$, where $\mathbf{W}$ is the total number of parameters in one model [223]. Further, Koiran et al. [224] mentioned that with continuous activation function, the VC-dimension of one multilayer neural network is proportional to the square of the number of the weights $\mathbf{W}$. Compared to the other MP inverse-based MR algorithms, the proposed SS-HSNN has fewer trainable parameters. Take the Coil-20 dataset as an example. Following the hyperparameter settings mentioned in Table 7.2, the number of neurons in each subnetwork is 500, and we completed 5 iterations to refine the latent space feature. The total number of trainable weights in SS-HSNN $= 1024 \times 500 \times 5 + 500 \times 20 \approx 2.6M$. For a SS-HMP structure, the smallest number of parameters (best case) in this structure is [1024-1000-1000-1000-1], and the total number of trainable weights in this case $= 1024 \times 2000 + 1000 \times 1000 + 1000 \times 1000 + 2000 \times 20 \approx 4.1M$. Thus, our model requires fewer weights. Thus, we have $\Phi_{SS-HMP}(h, t) \leq \Phi_{SS-HSNN}(h, t)$. Furthermore, the true risk of SS-HSNN and SS-HMP has the following conclusion:

$$\epsilon_{SS-HSNN} \leq \epsilon_{SS-HMP}. \tag{7.13}$$

7.5 Conclusion

In this chapter, we propose a new semi-supervised multilayer subnet-based neural network (SS-HSNN) to effectively handle image classification tasks. Specifically, this study makes the following contributions: First, it proposes a subnet-based algorithm with a new connection topology for semi-supervised classification. Second, it introduces a one-step training strategy that combines global-level representation learning and final pattern recognition to boost the network's training efficiency. Third, a new semi-supervised dataset HFSWR-RDE was collected. The cross-domain experiments on ten semi-supervised classification datasets, one text categorization dataset, and one radar signal-processing dataset show that the SS-HSNN can accomplish semi-supervised classification with higher accuracy and efficiency than existing state-of-the-art semi-supervised algorithms.

For future research, it would be worthwhile to develop a novel subnet-based algorithm with self-training capabilities for semi-supervised learning.

Chapter 8

Multi-Model Moore-Penrose Inverse-based Recomputation Frameworks for Large Data Analysis

Most multilayer MP inverse-based neural networks, such as deep random vector functional link (RVFL), are structured with two separate stages: unsupervised feature encoding and supervised pattern classification. Once the unsupervised learning is finished, the latent encoding would be fixed without supervised fine-tuning. However, in complex tasks such as handling the ImageNet dataset, there are often many more clues that can be directly encoded, while unsupervised learning, by definition, cannot know exactly what is useful for a certain task. There is a need to retrain the latent space representations in the supervised pattern classification stage to learn some clues that unsupervised learning has not yet been learned. In particular, the error matrix from the output layer is pulled back to each hidden layer, and the parameters of the hidden layer are recalculated with MP inverse for more generalized representations. In this chapter, a recomputation-based multilayer network using MP inverse (RML-MP) is developed. A sparse RML-MP (SRML-MP) model to boost the performance of RML-MP is then proposed. The experimental results with varying training samples (from $3\,K$ to $1.8\,M$) show that the proposed models provide better generalization performance than most representation learning algorithms.

8.1 Introduction

The AE-based RL structures have been widely investigated [149, 23]. The AE is an unsupervised learning algorithm that generally applies iterative learning strategies such as SGD as a cornerstone of their training, aiming to learn the reduced encoding by reproducing the input patterns at the output layer. The AE was firstly introduced in [225] as a way of pretraining in ANNs. Then, the structure is employed for representation learning tasks, such as sparse AE (SAE) [150], denoise AE (DAE) [101], and weight-decay regularization-based AE (WD-AE) [149].

Not just learnt with iterative learning strategies, AEs trained with LS-based MP inverse techniques were also investigated in the past decade. The ELM [181], which adopts random hidden layer neurons and generates the output weights with the LS-based MP inverse, is a single layer neural network with fast training speed and excellent generalization accuracy. Following that, the multilayer ELM (M-ELM) [42], which is stacked with ℓ_2 penalty-based ELM autoencoders (ELM-AEs), has been proposed. Then, authors in [98] developed a hierarchical framework structured with ℓ_1 penalty-based ELM-AEs to explore the sparse representations. In recent years, researchers in various real-world applications have made significant contributions to broaden the field scope of ELM [148, 226, 227], and explosive developments on multilayer ELM-AE based RL algorithms have been witnessed [25, 173, 23, 113, 172].

However, the existing ELM-AE-based RL algorithms cannot be directly adopted in multiview big data analysis. They have several limitations: Firstly, they cannot obtain satisfactory results on high-dimensional datasets with a large number of training samples. Most ELM-AE-based RL algorithms only focused on processing small-scale and medium-scale datasets. The performance of these models on big datasets has rarely been exploited.

Secondly, most of the ELM-AE-based RLs generate loosely-connected representations in processing large-scale datasets. The MP inverse-based AE belongs to the unsupervised dimension reduction technique since no labels are included in the training. In most cases, MP inverse-based AE is an intermediate model toward the final objects. For example, Katuwal et al. [228] proposed an MP inverse-based deep network using multiple MP inverse-based AEs for feature extraction. Then, the weights of the final layer are analytically calculated on the low-dimensional representation using MP inverse. In contrast to BP-based AEs for which the parameters of each AE would be fine-tuned according to the label information, the multilayer MP inverse-based networks do not contain fine-tuning. Therefore, the parameters of these AEs are fixed once the feature is extracted. Such a training process makes MP inverse-based networks achieve fast training speed and comparable performance on small-scale datasets over some BP-based deep networks. However, lacking supervision, some important clues may be filtered before training the final layer, which affects the final performance [229]. This issue is more obvious when handling large-scale sets with high complexity.

To address the above-mentioned limitations, this chapter proposes a novel MP inverse-based algorithm called RML-MP for big data analysis. First, the MP inverse-based AEs using the ℓ_2 penalty are stacked to exploit the effective encoding from complex input data, and the final classification layer is calculated with the LS scheme. Then, the MP inverse strategy is applied to pull back the error from the output layer to each hidden layer one by one, generating the desired output $\mathbf{P}$ for each layer. Finally, based on the input data and desired output, the MP inverse technique is utilized to recompute weights in each layer. By doing so, the bond between hidden layer representations and labels is strengthened, and robust representations can be obtained. Meanwhile, the effective $\ell_{1/2}$ penalty-based learning framework is adopted in the retraining stage of RML-MP, leading to a sparse algorithm SRML-MP. The $\ell_{1/2}$ penalty in retraining helps SRML-MP to obtain sparse weights. Furthermore, as stated in [8, 230], for most applications, multi-model feature learning is more practical than single-model learning. It is thus of great interest to exploit multi-modal big data analysis for higher performance purposes.

This chapter makes three contributions to extant knowledge. First, an RML-MP framework is proposed to perform multi-modal big data encoding. Second, an SRML-MP is further developed to improve the performance of RML-MP. The flowchart of RML-MP and SRML-MP is depicted as Fig. 8.1. Third, a comprehensive comparison is conducted to validate the effectiveness of the proposed models over the other representation learning approaches on different datasets, such as Places-365.

8.2 The proposed algorithms

8.2.1 The Proposed RML-MP

Table 8.1 describes the notations used in this chapter.

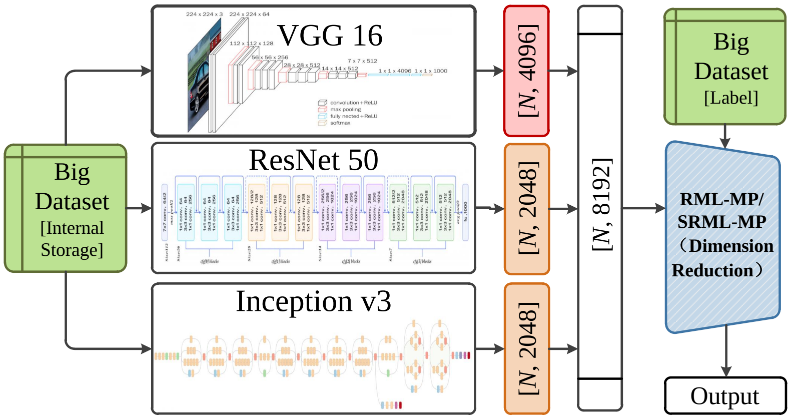
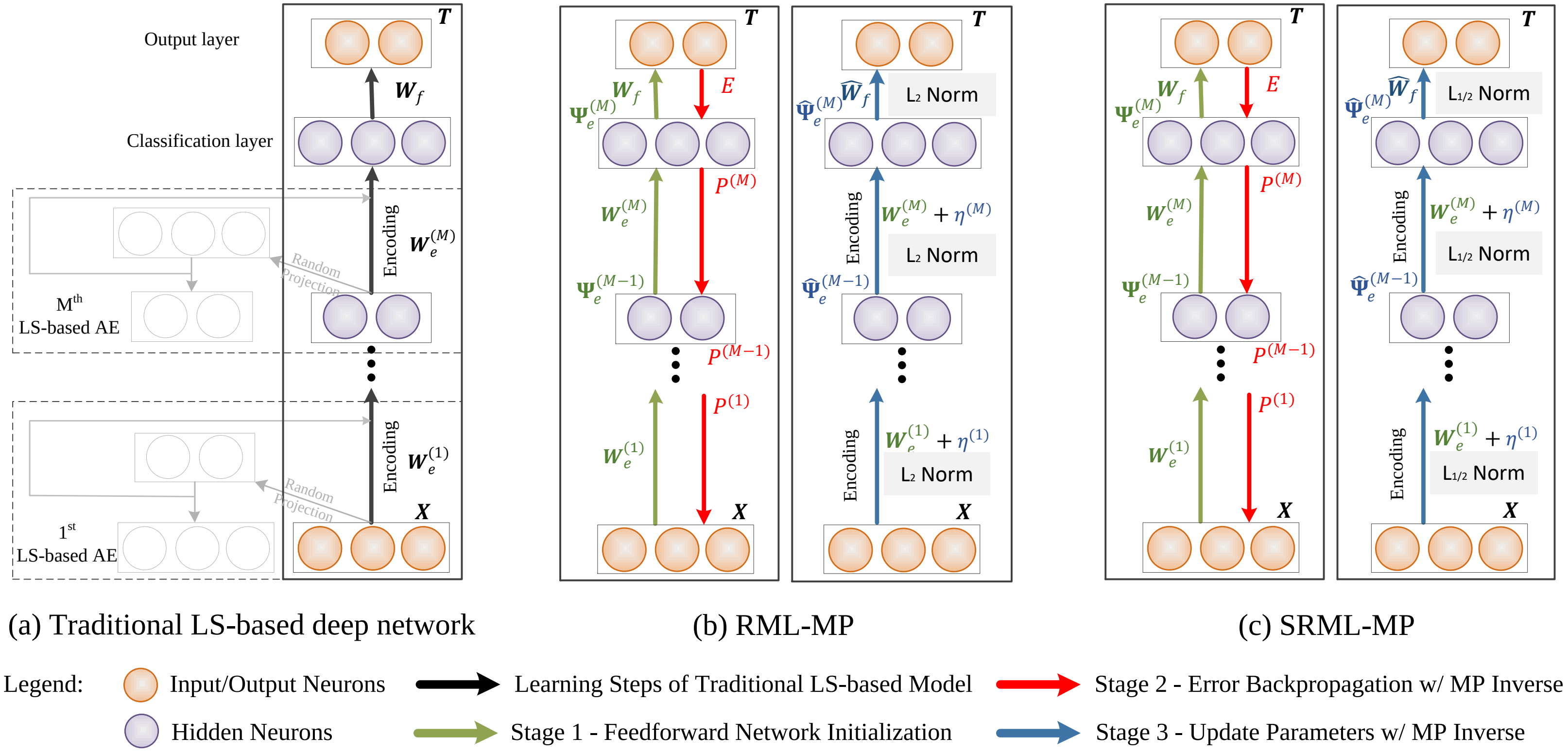


Figure 8.1 – Flowchart of the RML-MP and SRML-MP for multi-model RL.

Table 8.1 – Notations to be used in this chapter

Notation	Meaning
$\boldsymbol{\eta}^{(i)}$	offset weights of the i-th encoding layer
$\sigma(\cdot)$	the activation function
$\Psi_e^{(i)}$	the encoding of the i-th layer
$\hat{\Psi}_e^{(i)}$	the updated encoding of the i-th layer
C	regularization term C
$\mathbf{E}$	error matrix from the output layer
M	the number of AEs in RML-MP / SRML-MP
m	the dimensionality of the output
N	the number of samples
n	the dimensionality of each sample
λ	learning rate
$\mathbf{P}^{(i)}$	the i-th layer error feedback data
$\mathbf{T}$	the target, $\mathbf{T} \in R^{N \times d}$
$\mathbf{W}_e^{(i)}$	parameters of the i-th encoding layer
$\hat{\mathbf{W}}_e^{(i)}$	updated parameters of the i-th encoding layer
$\mathbf{W}_f$	parameters of the last classification layer
$\hat{\mathbf{W}}_f$	updated parameters of the last classification layer
$\mathbf{X}$	the input, $\mathbf{X} \in R^{N \times n}$
$\mathbf{Y}$	the actual output, $\mathbf{Y} \in R^{N \times m}$



(a) Traditional LS-based deep network

(b) RML-MP

(c) SRML-MP

Figure 8.2 – Comparison of frameworks of the (a) traditional MP inverse-based deep network [4], (b) the proposed RML-MP, and (c) the proposed SRML-MP. The difference of RML-MP and SRML-MP lies in Stage 3, the RML-MP use ℓ_2 penalty to recalculate the parameters while the SRML-MP adopt $\ell_{1/2}$ penalty to update the weights.

Figure 8.2 shows the diagrams of the traditional MP inverse-based model, the proposed RML-MP, and the SRML-MP for comparison. The main difference between the proposed models and other traditional MP inverse-based neural networks [4, 228] lies in the model training procedures. The proposed RML-MP contains three successive learning stages: Stage 1 - feedforward network initialization, Stage 2 - error backpropagation with MP inverse, and Stage 3 - update parameters with MP inverse.

8.2.1.1 Stage 1 - Feedforward Network Initialization

The first stage aims to develop a traditional MP inverse-based deep neural network. Given a training dataset with N number of training samples, $\mathbf{X} = [\mathbf{x}_1^T, \mathbf{x}_2^T, \cdots, \mathbf{x}_N^T]^T$, $\mathbf{x}_i \in \mathcal{R}^n$ is the input vector, $\mathbf{T} = [\mathbf{t}_1^T, \mathbf{t}_2^T, \cdots, \mathbf{t}_N^T]^T$, $\mathbf{t}_i \in \mathcal{R}^m$ is its associated target. The MP inverse-based AE [4, 228, 42] tries to encode the input data through setting the input as the target. In particular, the optimal output layer parameters of MP inverse-based AE are calculated with the MP inverse technique. With randomly assigned hidden layer weights $\mathbf{W}$ and b, the ℓ_2 norm-based AE is optimized with the following minimizing problem:

$$\begin{aligned} &\min\ J = \frac{C}{2}||\Psi_e \mathbf{W}_e - \mathbf{X}||_F^2 + \frac{1}{2}||\mathbf{W}_e||_F^2, \\ &\text{s.t. } \Psi = \sigma(\mathbf{X}, \mathbf{W}, b), \text{ and} \\ &\quad \mathbf{W}^T\mathbf{W} = I,\ b^Tb = 1, \end{aligned} \tag{8.1}$$

where $\sigma(\cdot)$ is the activation function (sine or sigmoid), C is the regularization term, Ψ_e is the hidden space encoding, and $\mathbf{W}_e$ refers to the output layer weight, which is calculated with MP inverse: $\mathbf{W}_e = \Psi_e^\dagger \mathbf{X}$. In this chapter, with the identity matrix I, the LS fit is utilized:

$$\mathbf{W}_e = \Psi_e^\dagger \mathbf{X} = (\frac{I}{C} + \Psi_e^T \Psi_e)^{-1} \Psi_e^T \mathbf{X}. \tag{8.2}$$

The encoding Ψ_e of the MP inverse-based AE is

$$\Psi_e = \sigma\left(\mathbf{X} \cdot \mathbf{W}_e^T\right). \tag{8.3}$$

Supposing that an multilayer MP inverse-based network has M hidden layers. Mathematically, the objective function J to learn this deep network can be described as:

$$\begin{aligned} &\min\ J = \frac{C}{2}||\Psi_e^{(M)}\mathbf{W}_f - \mathbf{T}||_F^2 + \frac{1}{2}||\mathbf{W}_f||_F^2, \\ &\text{s.t. } \Psi_e^{(i)} = \sigma\left(\Psi_e^{(i-1)} \cdot \left(\mathbf{W}_e^{(i-1)}\right)^T\right), 1 \leq i \leq M, \end{aligned} \tag{8.4}$$

where $\Psi_e^{(i)}$ is the i-th layer output matrix, $\mathbf{X}$ can be considered as the 0-th layer feature matrix ($\Psi_e^{(i)}$ where i equal to zero), $\mathbf{W}_f$ refers to the weights of the classification layer which is generated through MP inverse, and $\mathbf{W}_e^{(i)}$ is the i-th layer weights that would be fixed once determined.

The output layer weights $\mathbf{W}_f$ are calculated as Eq. (8.5).

$$\begin{aligned} &\mathbf{W}_f = \left(\frac{I}{C} + (\Psi_e^{(M)})^T\Psi_e^{(M)}\right)^{-1}(\Psi_e^{(M)})^T\mathbf{T}, \\ &\mathbf{Y} = \Psi_e^{(M)} \cdot \mathbf{W}_f. \end{aligned} \tag{8.5}$$

8.2.1.2 Stage 2 - Error Backpropagation with MP Inverse

For the classification layer, the target $\mathbf{T}$ and the network output $\mathbf{Y}$ are obtained. Next, the network error $\mathbf{E}$ is pulled back from the output layer to each hidden layer. The proposed retraining strategy adjusts the hidden layer representations by attempting to offset the feedback error. First, the error of the output layer can be described by

$$\mathbf{E} = \mathbf{T} - \Psi_e^{(M)} \cdot \mathbf{W}_f. \tag{8.6}$$

Then, we pull the error back across the last classification layer. By having $\mathbf{E}$ and $\mathbf{W}_f$, the desired change of the last hidden layer is

$$\mathbf{P}^{(M)} = \mathbf{E} \cdot \left(\frac{I}{C} + {\mathbf{W}_f}^T\mathbf{W}_f\right)^{-1}\mathbf{W}_f^{\mathbf{T}}, \tag{8.7}$$

where $(I/C + \mathbf{W}_f^T\mathbf{W}_f)^{-1}\mathbf{W}_f^{\mathbf{T}}$ is the MP inverse of $\mathbf{W}_f$. The target change of other hidden layer is calculated by

$$\mathbf{P}^{(i-1)} = \sigma^{-1}\left(\mathbf{P}^{(i)} \cdot \left(\frac{I}{C} + \left(\mathbf{W}_e^{(i)}\right)^T \mathbf{W}_e^{(i)}\right)^{-1} \cdot \left(\mathbf{W}_e^{(i)}\right)^T\right), \tag{8.8}$$

where $2 \leq i \leq M$, $\sigma^{-1}(\cdot)$ is the inverse of activation function, and $\mathbf{P}^{(i)}$ is the i-th layer target offset.

8.2.1.3 Stage 3: Update Parameters with MP Inverse

In this chapter, we hypothesize that the error $\mathbf{E}$ and feedback data $\mathbf{P}^{(i)}$ contain some information clues that AEs have not learned, and the weights optimized by the target offset $\mathbf{P}^{(i)}$ and encoded feature $\Psi_e^{(i)}$ can boost the feature encoding capacity and improve the generalization performance. After error backpropagation, at Stage 3, both the output layer and hidden layer weights of RML-MP are updated.

For the i-th hidden layer, an error-based update weight $\boldsymbol{\eta}^{(i)}$ is calculated, satisfying

$$\hat{\Psi}_e^{(i-1)}(\mathbf{W}_e^{(i)} + \lambda \cdot \boldsymbol{\eta}^{(i)}) = \Psi_e^{(i)} + \mathbf{P}^{(i)},\ 1 \leq i \leq M \tag{8.9}$$

where λ stands for learning rate, $\hat{\Psi}_e^{(i-1)}$ is the updated $(i-1)$-th hidden layer representations. The weight $\boldsymbol{\eta}^{(i)}$ aims to offset the error $\mathbf{P}^{(i)}$. Hence, we have $\hat{\Psi}_e^{(i-1)} \cdot \boldsymbol{\eta}^{(i)} = \mathbf{P}^{(i)} \rightarrow \boldsymbol{\eta}^{(i)} = (\hat{\Psi}_e^{(i-1)})^{\dagger}\mathbf{P}^{(i)}$. By using MP inverse, $\boldsymbol{\eta}^{(i)}$ is calculated by

$$\boldsymbol{\eta}^{(i)} = \left(\frac{I}{C} + (\hat{\Psi}_e^{(i-1)})^T \hat{\Psi}_e^{(i-1)}\right)^{-1} (\hat{\Psi}_e^{(i-1)})^T \mathbf{P}^{(i)}, \tag{8.10}$$

where $\hat{\Psi}_e^{(i-1)}$ is the updated encoding calculated by Eq. (8.11).

$$\hat{\Psi}_e^{(i-1)} = \begin{cases} \mathbf{X}, & i = 1 \\ \sigma\left(\hat{\Psi}_e^{(i-2)} \cdot \hat{\mathbf{W}}_e^{(i-1)}\right), & 2 \leq i \leq M, \end{cases} \tag{8.11}$$

In Eq. (8.11), $\hat{\mathbf{W}}_e^{(i)}$ is the updated weight. When $i = 1$, the weights of 1-st hidden layer are updated. The input $\mathbf{X}$ is considered as the 0-th layer feature $\hat{\Psi}_e^{(0)}$. Thus, the i-th hidden layer weight can be updated by

$$\begin{aligned} \hat{\mathbf{W}}_e^{(i)} &= \mathbf{W}_e^{(i)} + \lambda \cdot \boldsymbol{\eta}^{(i)} \\ &= \mathbf{W}_e^{(i)} + \lambda \cdot \left(\frac{I}{C} + (\hat{\Psi}_e^{(i-1)})^T \hat{\Psi}_e^{(i-1)}\right)^{-1} (\hat{\Psi}_e^{(i-1)})^T \mathbf{P}^{(i)}, \end{aligned} \tag{8.12}$$

Then, the weights of the output layer are calculated by

$$\hat{\mathbf{W}}_f = \left(\frac{I}{C} + (\hat{\Psi}_e^{(M)})^T \hat{\Psi}_e^{(M)}\right)^{-1} (\hat{\Psi}_e^{(M)})^T \cdot \mathbf{T}, \tag{8.13}$$

8.2.1.4 The Learning Steps of RML-MP

The proposed RML-MP can be summarized as follows.

- **Step 1**: Given input features and the corresponding labels $\{\mathbf{X},\ \mathbf{T}\}$, the learning rate λ, and regularization term C.
- **Step 2**: Train a multilayer MP inverse-based network with M AEs, analytically calculate the output layer parameters $\mathbf{W}_f$ with Eq. (8.5).
- **Step 3**: Pull back error term $\mathbf{E}$ from the classification layer to the first hidden layer by Eq. (8.7) and Eq. (8.8) one-by-one.
- **Step 4**: Recalculate weights $\hat{\mathbf{W}}_e^{(i)}$ and $\hat{\Psi}_e^{(i)}$ from the 1-st hidden layer to the M-th hidden layer via Eq. (8.10) and Eq. (8.12) sequentially.
- **Step 5**: Update classification layer weights $\hat{\mathbf{W}}_f$ by Eq. (8.13).

8.2.2 The Proposed SRML-MP

Essentially, the proposed RML-MP utilizes a ℓ_2 norm to recompute the parameters. However, the updated weight with such a regularization penalty tends to be dense and may further lead to over-fitting problems. Thus, in this chapter, a sparse solution calculated with $\ell_{1/2}$ penalty is then developed.

8.2.2.1 Sparse Learning

SAE [150] is an AE whose training criterion contains sparse penalty. One can build a model's loss function via penalizing activations of hidden layers so that only a limited number of neurons are activated. The most commonly used SAE is developed through the use of the ℓ_1 regularizer. Experimental results [231] have already shown the superior performance of sparse learning on some small-scale datasets. Thus, it is intuitive to propose a retraining algorithm using sparse learning for better generalization performance. Recently, authors in [232] have mentioned that $\ell_{1/2}$ norm can produce sparser solutions than ℓ_1 regularizer. Thus, a structure of combining RML-MP with $\ell_{1/2}$ norm (SRML-MP) is further proposed.

In SRML-MP, the updated weight $\boldsymbol{\eta}^{(i)}$ for the i-th encoding layer is calculated as

$$\min J = \frac{1}{2}||\hat{\Psi}_e^{(i-1)}\boldsymbol{\eta}^{(i)} - \mathbf{P}^{(i)}||_2^2 + C||\boldsymbol{\eta}^{(i)}||_{1/2}^{1/2}, \tag{8.14}$$

where $||\boldsymbol{\eta}^{(i)}||_{1/2}^{1/2} = \sum_{j=1}^{N} |\boldsymbol{\eta}_j^{(i)}|^{1/2}$. A fast iterative jumping thresholding (IJT) [233] algorithm is adopted to solve Eq. (8.14). Specifically, with IJT [233], the proximity operator $pro_{\mu,C|\cdot|^q}$ of $\ell_q(0 < q < 1)$ regularization can be described as

$$pro_{\mu,C|\cdot|^q}(z) = \begin{cases} \left(\cdot + \frac{C}{2}\mu q \text{sign}(\cdot)^{q-1}\right)^{-1}(z), & |z| \geq \tau_{\mu,q} \\ 0, & |z| \leq \tau_{\mu,q}, \end{cases} \tag{8.15}$$

for any $z \in R$, where

$$\begin{aligned} \tau_{\mu,q} &= \frac{2-q}{2-2q}\left(C\mu(1-q)\right)^{\frac{1}{2-q}}, \\ \psi_{\mu,q} &= \left(C\mu(1-q)\right)^{\frac{1}{2-q}}. \end{aligned} \tag{8.16}$$

The range of $pro_{\mu,C|\cdot|^q}$ is $\{0\} \cup [\psi_{\mu,q}, \infty)$. For $q = 1/2$, with proximity operator, the i-th encoding layer weights $\boldsymbol{\eta}^{(i)}$ can be expressed analytically [234]:

$$\boldsymbol{\eta}^{(i)} = U_{\mathbf{P}^{(i)}} \cdot \text{diag}\{\sigma_{\mathbf{P}^{(i)}} - \sqrt{C}P_{\ell 1}(\frac{\sigma_{\mathbf{P}^{(i)}}}{\sqrt{C}})\} \cdot V_{\mathbf{P}^{(i)}}^T, \tag{8.17}$$

where $U_{\mathbf{P}}$, $V_{\mathbf{P}}$, and $\sigma_{\mathbf{P}^{(i)}}$ stand for the left unitary matrix of SVD of target output $\mathbf{P}^{(i)}$, the right unitary matrix of SVD of target output $\mathbf{P}^{(i)}$, and singular values of $\mathbf{P}^{(i)}$, respectively. $P_{\ell 1}(\cdot)$ is the orthogonal projection of one vector onto the ℓ_1 unit ball.

8.2.2.2 The Learning Steps of SRML-MP

The proposed SRML-MP is developed with four steps.

- **Step 1**: Train a multilayer MP inverse-based network with M AEs, calculate output layer weights $\mathbf{W}_f$ via Eq. (8.5).

- **Step 2**: Pull back error $\mathbf{E}$ via Eq. (8.7) and Eq. (8.8).
- **Step 3**: Calculate the i-th layer updated weight $\boldsymbol{\eta}^i$ with $\ell_{1/2}$ penalty by IJT algorithm by Eq. (8.17), recompute the hidden layer weights via Eq. (8.12).
- **Step 4**: Update the output layer weights $\hat{\mathbf{W}}_f$ with $\ell_{1/2}$ norm.

8.3 Experimental Results

To evaluate the performance of the proposed RML-MP and SRML-MP, a set of experiments was conducted with other representation learning algorithms on the image classification domain. Additionally, to comprehensively evaluate the practical benefits of the proposed strategies, extensive discussion and experiments were carried out on the food categorization domain.

8.3.1 Experimental Setup

In this subsection, the experimental setup and the dataset description are presented.

8.3.1.1 The Environment

All of the experiments conducted in this chapter were performed in the Matlab R2019b environment, running in a workstation with Intel Core $E5-2650$ CPU and 256 GB memory. The high-level features extracted from DCNNs were carried out on the NVIDIA $1080Ti$ GPU.

8.3.1.2 The Datasets

In this chapter, seven image classification datasets were used for evaluation, Caltech-101, ImageNet-1/2, Places-365-1/2/3, and Places-365. Further, the proposed methods were evaluated on one food categorization dataset (Food-251 [131]). The dataset specifications are shown in Table 8.2. Details of each dataset are as follows.

Image classification datasets: The Places-365 and ImageNet datasets are considered the largest datasets in the image classification area. The ImageNet contains more

Table 8.2 – Summary of the datasets.

Modes	Datasets	Classes	Training	Testing
	Caltech-101	102	3060	6084
	ImageNet-1	1,000	200,000	50,000
	ImageNet-2	1,000	500,000	50,000
Image Classification	Places-365-1	365	182,500	36,500
	Places-365-2	365	365,000	36,500
	Places-365-3	365	547,500	36,500
	Places-365	365	1,803,460	36,500
Food Image Classification	Food-251	251	118,475	11,994

than 1.2 million patterns. In this chapter, two mini datasets are utilized. In particular, the ImageNet-1 and ImageNet-2 datasets are generated by randomly selecting 200 and 500 images per category from the dataset, respectively. The Places-365 dataset is composed of 365 categories, containing more than 1.8M images. 500, 1,000, and 1,500 images per category from Places-365 are selected to generate the Places-365-1, Places-365-2, and Places-365-3 datasets. For all of these datasets (ImageNet-1/2, Places-365-1/2/3), the validation set is utilized in the testing stage. Furthermore, the original Places-365 dataset is used to validate the proposed algorithms, where the training set is applied for training and the validation set is to test the model. Besides, Caltech-101 is a widely used dataset. Following the training settings in the previous works [23], we take 30 patterns per class for training and use the rest for testing.

Food categorization dataset: The recently proposed food dataset Food-251 is used for experimental analysis. The Food-251 dataset contains 118,475 training images and 11,994 validation images. In this study, the whole training set is used for training. The validation set is adopted to test the accuracy of the algorithms. The following methods were compared: low-level feature-based methods [235], high-level CNN-based algorithms [137], and feature fusion-based frameworks [115].

8.3.1.3 The Rival Methods

In this chapter, several autoencoder-based multilayer models are tested, which can be divided into the following three families.

Representation learning algorithms with SGD technique This family includes WD-AE [149], denoise autoencoder with Gaussian (DAE^G) [101] and binary (DAE^B) [101] making noise, and SAE [150]. After the optimal encoding is learned via each algorithm, a softmax classifier is used to find the final classification result. For the above-mentioned algorithms, fine-tuning is utilized to adjust the weights of the trained model to boost the network's generalization performance.

Representation learning algorithms with MP inverse technique These methods include hierarchical RVFL-based neural network (H-RVFL) [4], multilayer MP inverse-based framework (MLS) [42], subnet-based structure (SNN) [236], multilayer subnet framework (MSNN) [23], and Wi-HSNN [173]. Note that the weights of hidden layers are fixed after the unsupervised encoding is finished. In other words, these comparison algorithms do not contain any fine-tuning.

Furthermore, to fully validate the effectiveness of the proposed methods, we compared RML-MP and SRML-MP with the strategy that does not contain retraining. In other words, this strategy develops its model using Eq. (8.1) to Eq. (8.5) with no error backpropagation (Stage 2) and weights updating (Stage 3). We use a multilayer network with MP inverse (ML-MP) to denote this baseline.

8.3.1.4 Configurations of The RL Methods

For BP-based strategies, the training epochs and size of each mini-batch are set to be 100. The initial learning rate is set as 0.01 with a 0.1 attenuation rate for every 10 training epochs. The input corruption rate of DAE^G is 0.5. The training epochs for fine-tuning is 20. For MP inverse-based algorithms, the number of hidden neurons and the regularization term in all AEs are optimized within the grid $\{500,\ 1000,\ 2000\} \times \{10^{-3},\ 10^0,\ 10^3\}$, while the regularization term for the last classification layer is searched from $\{10^0,\ 10^2,\ 10^4\}$. The number of subnets is set as 5, and the number of neurons in each subnet is defined as 1000. Also, the regularization term in subnet-based models is optimized within the grid $\{10^0,\ 10^2,\ 10^4\}$. As for the proposed RML-MP and SRML-MP, the number of stacked AEs is 2, and the number of hidden neurons for each MP inverse-based AE is 1000. The sigmoid function is chosen as the non-linear activation function. The offset term C is 4.

8.3.1.5 Configurations of The Input Features

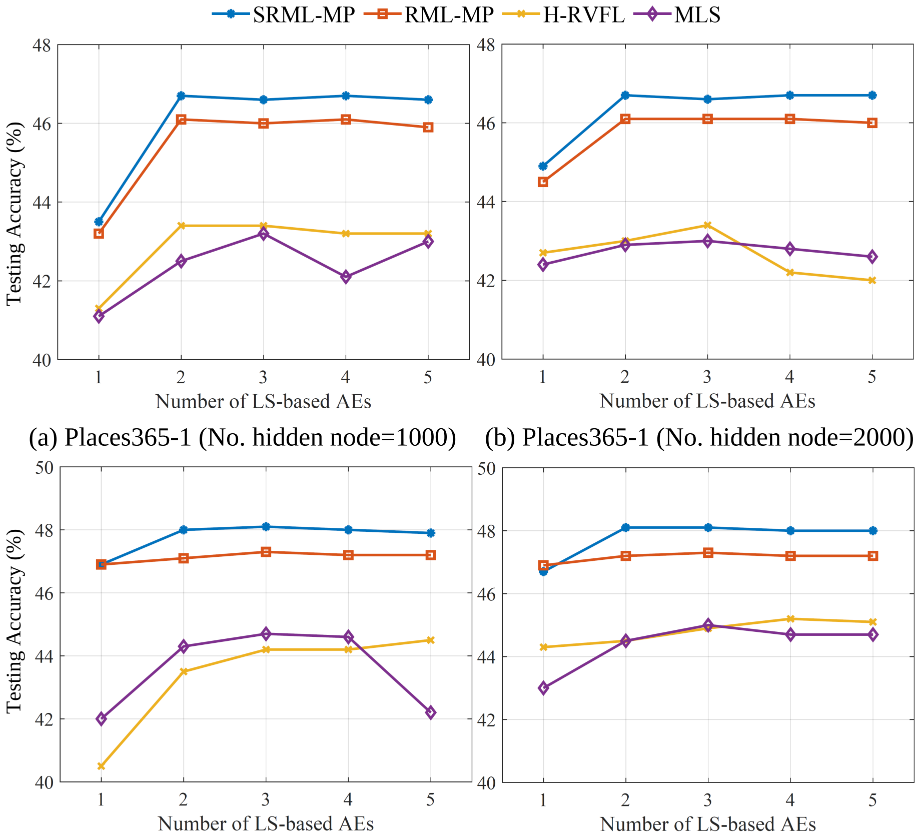
Multi-model algorithms fuse different sources of features that are complementary to each other to achieve superior recognition performance. Three pre-trained DCNNs, namely VGG-16 [87], ResNet-50 [88], and InceptionNet-v3 [237], are adopted as feature extractors. These DCNNs are pre-trained on the ImageNet set, and the final softmax layer contains 1,000 neurons. In this chapter, the high-level features that are extracted from the top layer of DCNNs are loaded as the raw feature. As for high-level feature extraction, the original top layer is decapitated and replaced with a new classification layer with the same number of classes as the target dataset. After the DCNN is fine-tuned, the high-level features are captured from the penultimate layer of each DCNN. Thus, the data extracted from VGG-16, ResNet-50, and InceptionNet-v3 are 4,096-, 2,048-, and 2,048-dimensional vectors, respectively. The dimensionality of the concatenated feature is 8,192.

8.3.1.6 Evaluation Matrics

For the image classification and food categorization, the Top-1 testing accuracy is adopted to compare the accuracy of the proposed algorithms and other state-of-the-art methods. Besides, the training time and the inference time are compared. For each algorithm, at least three trials were conducted for each dataset.

8.3.2 Model Settings

The proposed retraining algorithms have several hyperparameters, such as the number of AEs used in SRML-MP/RML-MP and the number of neurons in each AE. In this subsection, we empirically verify the recommendations of these two hyperparameters. Figure 8.3 shows the Top-1 testing accuracy of the proposed methods, MLS, and H-RVFL on Places-365-1/2 datasets as the number of AEs increased. In this evaluation, although each algorithm's learning procedures differ, the offset term C, the number of hidden neurons in each AE remain the same (1,000 or 2,000) for consistency and fair comparison. Fig. 8.3 (a) and (c) depict the results when the number of hidden neurons in each AE equals 1,000, while Fig. 8.3 (b) and (d) show the comparison results when the number of latent space nodes in each AE is 2,000. It is observed that i) the performance of RML-MP and SRML-MP converge after the number of MP inverse-based AEs equals 2, ii) the proposed models achieve competent performance when the number of neurons in each AE is 1,000, and iii) with the same hyperparameters, the Top-1 testing accuracy of the proposed models is improved by 2% when compared to the existing multilayer MP inverse-based algorithms, such as H-RVFL and MLS. Thus, in this chapter, for the proposed RML-MP and SRML-MP, the number of stacked AEs is set to 2, and the number of hidden neurons in each AE is 1,000.



(a) Places365-1 (No. hidden node=1000) (b) Places365-1 (No. hidden node=2000)

(c) Places365-2 (No. hidden node=1000) (d) Places365-2 (No. hidden node=2000)

Figure 8.3 – Comparison of different algorithms with Inception-v3 features. (a) and (b) are the results on Places-365-1 dataset, (c) and (d) are the results on Places-365-2 dataset.

8.3.3 Analysis on Image Classification Domain

In this subsection, the testing accuracy of the proposed RML-MP, SRML-MP, and other comparison algorithms on several commonly used image classification datasets is reported.

To verify the effectiveness of multi-modal learning, a sanity check with various combinations of high-level features is conducted as reported in Table 8.3. Different combinations from single-model to multi-model features are loaded as input. The experimental results show that the fusion of multi-modal features provides a remarkable performance boost over models trained with single-model features. For example, the RML-MP with the concatenated features (ResNet-50 + InceptionNet-v3 + VGG-16) provides 48.9%, 51.8%, and 82.5% testing accuracy on the Places-365-1, Places-365-3, and ImageNet-2 datasets, respectively, having 2.2%, 4.6%, and 16.7% higher accuracy than the single-model VGG-16 feature.

Table 8.3 – Effectiveness analysis of feature fusion strategy.

Methods	Features	Places-365-1	Places-365-3	ImageNet-2
RML-MP	VGG	46.7	47.2	65.8
	ResNet50	45.2	45.9	79.3
	Inception-v3	46.2	48.6	81.4
	ResNet50, Inception-v3	47.2	49.5	82.1
	VGG, ResNet50, Inception-v3	48.9	51.8	**82.5**
SRML-MP	VGG	46.8	47.6	66.2
	Re.	45.3	46.8	79.3
	Inception-v3	46.8	49.0	81.5
	ResNet50, Inception-v3	47.8	50.7	82.2
	VGG, ResNet50, Inception-v3	**49.9**	**52.0**	82.4

The overall comparisons of the existing multilayer LS-based RL algorithms and the proposed methods on the image classification datasets are provided in Table 8.4. Along with the Top-1 testing accuracy, the mean average performance among all datasets is shown as well. Through comparison, the following conclusions can be drawn:

Table 8.4 – Top-1 testing accuracy comparison among different non-iterative RL methods. Values in BLUE are the best results with Inception-v3 features. The ones in RED are the best results with concatenated features (Inc. - Inception-v3 features)

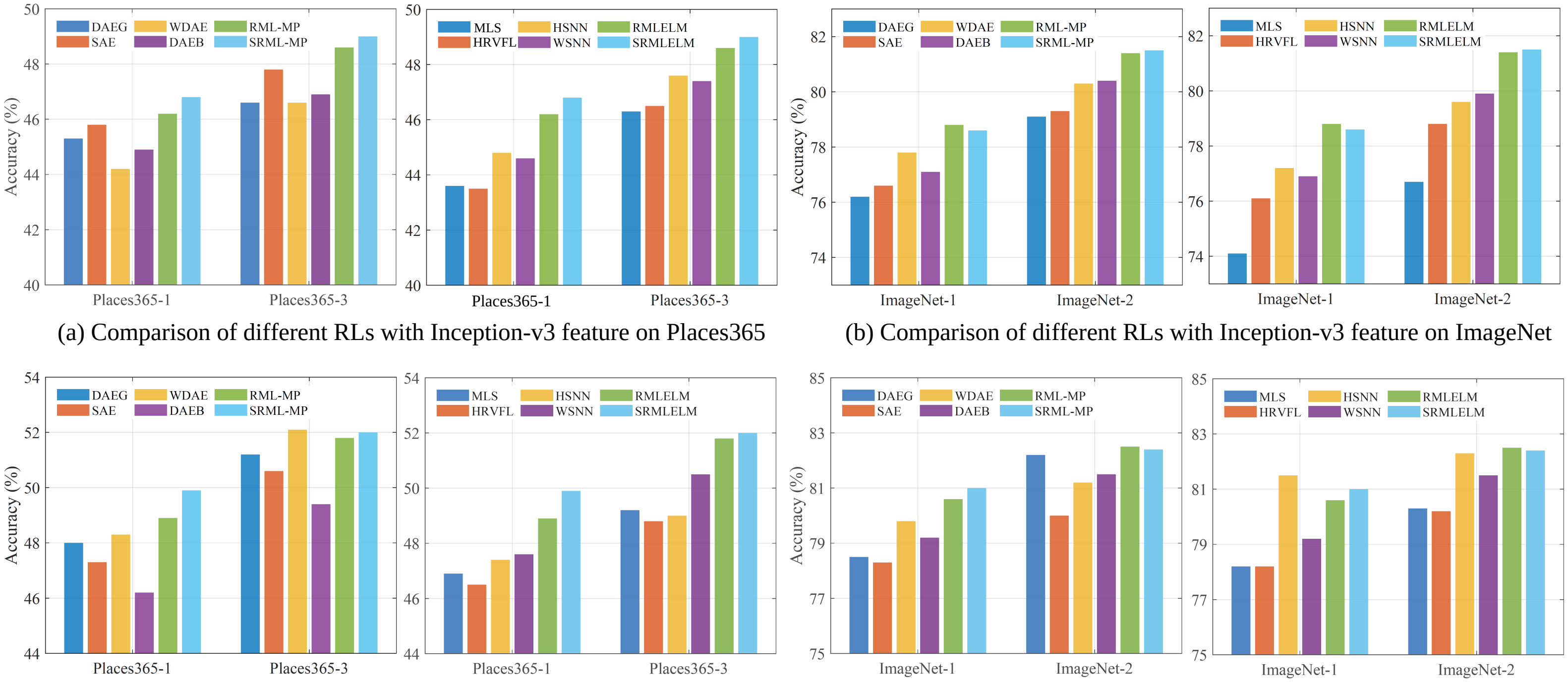
Methods	Caltech-101	ImageNet-1	ImageNet-2	Places-365-1	Places-365-2	Places-365-3	Places-365	Average
Single-model: Inception-v3 features								
Inc.+H-RVFL [4]	90.2	76.1	78.8	43.5	45.7	46.5	48.8	61.4
Inc.+MSNN [23]	89.4	77.2	79.6	44.8	45.4	47.6	48.9	61.8
Inc.+MLS [42]	89.7	74.1	76.7	43.6	45.2	46.3	46.6	60.3
Inc.+SNN [236]	88.6	76.2	79.2	42.7	45.5	47.1	48.4	61.1
Inc.+Wi-HSNN [173]	89.6	76.9	79.9	44.6	46.7	47.4	49.8	62.1
Inc.+ML-MP	89.6	74.0	76.9	42.9	45.7	45.8	46.9	60.2
Inc.+RML-MP	91.7	78.8	81.4	46.2	47.1	48.6	52.9	63.8
Inc.+SRML-MP	91.8	78.6	81.5	46.8	48.2	49.0	52.5	64.1
Multi-model: concatenated features								
All+H-RVFL [4]	91.4	78.2	80.2	46.5	48.9	48.8	51.7	63.7
All+MSNN [23]	92.2	81.5	82.3	47.4	48.8	49.0	51.4	64.7
All+MLS [42]	91.7	78.2	80.3	46.9	47.7	49.2	50.3	63.5
All+SNN [236]	91.8	78.9	80.1	45.7	47.4	47.7	51.5	63.3
All+Wi-HSNN [173]	92.1	79.2	81.5	47.6	49.4	50.5	52.6	64.7
All+ML-MP	91.5	78.7	80.4	46.7	47.9	49.3	50.5	63.6
All+RML-MP	93.2	80.6	82.5	48.9	51.1	51.8	55.7	66.3
All+SRML-MP	93.4	81.0	82.4	49.9	51.6	52.0	55.0	66.5

- The proposed RML-MP and SRML-MP with single-model feature provide superior performance compared to the comparison algorithms. For instance, the SRML-MP shows a valuable increment over the ML-MP, including 3.2%, 4.6% and 5.6% on Places-365-3, ImageNet-2 and Places-365, respectively.
- The RML-MP and SRML-MP with the concatenated features have competent performance over other RL frameworks. For example, the mean average accuracy of RML-MP and SRML-MP is promoted by 2.8% and 3.0% than that of the MLS, respectively.
- Compared to SRML-MP, RML-MP offers benefits in processing big and complex datasets (with more than 1 million samples). For example, when only considering multi-model concatenated feature, RML-MP achieves 0.7% better accuracy than the SRML-MP on Places-365 dataset.

Furthermore, the comparison results with all state-of-the-art autoencoder-based representation learning algorithms, including DAE^G [101], SAE [150], WD-AE [149], MLS [42], H-RVFL [4], MSNN [23], and Wi-HSNN [173], are shown in Fig. 8.4. As can be observed, the proposed RML-MP and SRML-MP generally outperform the rest of the autoencoder-based algorithms. Therefore, the benefits of the proposed RML-MP and SRML-MP on big data analysis are verified.

8.3.4 Timing Analysis

Table 8.5 tabulates the training and inference time complexities of the existing MP inverse-based methods and the proposed RML-MP and SRML-MP. All of the results recorded in Table 8.5 are counted in minutes. For a fair comparison, the DCNN training time and feature pre-processing time are ignored. Only the complexities of network encoding and pattern classification are recorded. One can readily see from the table that the MLS needs the shortest training time, while the proposed RML-MP and SRML-MP take around two to three times longer for training. The reason for the longer training time is that, on one dataset, the proposed learning pipelines search the better representations by pulling back the error matrix from the final classification layer to each hidden layer respectively.



(a) Comparison of different RLs with Inception-v3 feature on Places365

(b) Comparison of different RLs with Inception-v3 feature on ImageNet

(c) Comparison of different RLs with concatenated features on Places365

(d) Comparison of different RLs with concatenated features on ImageNet

Figure 8.4 – Comparison of different autoencoder-based algorithms on Places-365 and ImageNet datasets. (a) and (b) are the comparisons with Inception-v3 feature, (c) and (d) are the results with all concatenation feature.

Table 8.5 – Processing time w/ Inception features in MINUTE

Methods	Places-365-1	Places-365-2	Places-365-3	ImageNet-2
		Training time comparison		
MLS	1.7	3.0	5.5	4.9
H-RVFL	2.5	4.4	9.2	8.5
SNN	17.7	27.3	38.7	35.1
MSNN	18.4	38.2	49.8	48.3
RML-MP	4.9	10.5	20.6	18.1
SRML-MP	6.6	11.9	18.4	16.9
		Testing time comparison		
MLS	2.2	2.3	2.2	3.8
H-RVFL	3.3	3.3	3.3	5.1
SNN	4.9	4.8	4.8	8.2
MSNN	6.2	6.5	6.4	9.8
RML-MP	2.1	1.9	2.0	3.3
SRML-MP	2.0	2.0	2.0	3.1

Although the proposed MRL-MP and SMPL-MP need longer training time, they are more effective as they provide much better generalization performance than the existing multilayer MP inverse-based methods. As for the inference time, the RML-MP and SRML-MP provide similar inference time as that of MLS. Also, they require less time compared to the rest algorithms. For example, on the ImageNet-2 dataset, the proposed RML-MP and SRML-MP speed up the inference by around 1.5, 2.5 and 3.0 times when compared to the H-RVFL, SNN and MSNN, respectively.

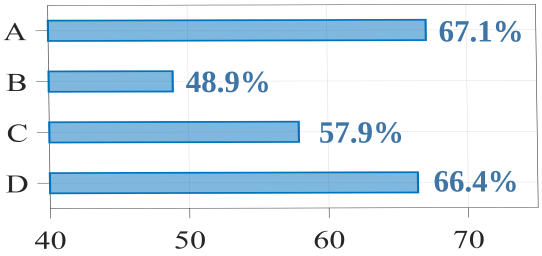
8.3.5 Analysis on Food Image Classification Domain

To further verify the effectiveness of the proposed RML-MP and SRML-MP algorithms, experiments were further conducted on the food image classification dataset. Here, several state-of-the-art food image classification algorithms [137, 139, 115] are compared, and the results are described in Table 8.6. For consistency and fair comparison, the same pre-processing steps and input features as those in [115] are utilized.

As shown in Table 8.6, compared with the state-of-the-art food image categorization model (width model [115]), the proposed RML-MP algorithm still provides better performance with a 0.7% of improvement. Besides, the proposed feature encoding structure show superior performance over the AlexNet-based model [137] and fusion method [139] by 18.2% and 9.2% of boost, respectively.

Table 8.6 – Comparison with other algorithms on food categorization (%).

Methods	Food-251
B: AlexNet-based [137]	48.9
C: Fusion of DCNN [139]	57.9
D: Width model [115]	66.4
A: RML-MP	**67.1**



8.3.6 Qualitative Analysis

T-SNE-based visual inception is presented by comparing the encoded features by different algorithms in Fig. 8.5. Due to space constraints, we only show the qualitative comparison with MLS, RML-MP, and SRML-MP on one dataset, i.e., Caltech101. It can be concluded from Fig. 8.5 (b), (c), and (d) that RML-MP and SRML-MP enhance the outer distance and reduce the inner distance of each category. The above result suggests that the proposed multilayer MP inverse-based retraining strategies can explain the loaded data in a more discriminative way. The reason is that the encoding from MP inverse-based AEs only contains unsupervised details; however, some important clues may be missing without the help of supervised learning. The proposed retraining strategies provide the network with the opportunity to distill the representations after unsupervised encoding, thus leading to better generalization performance.

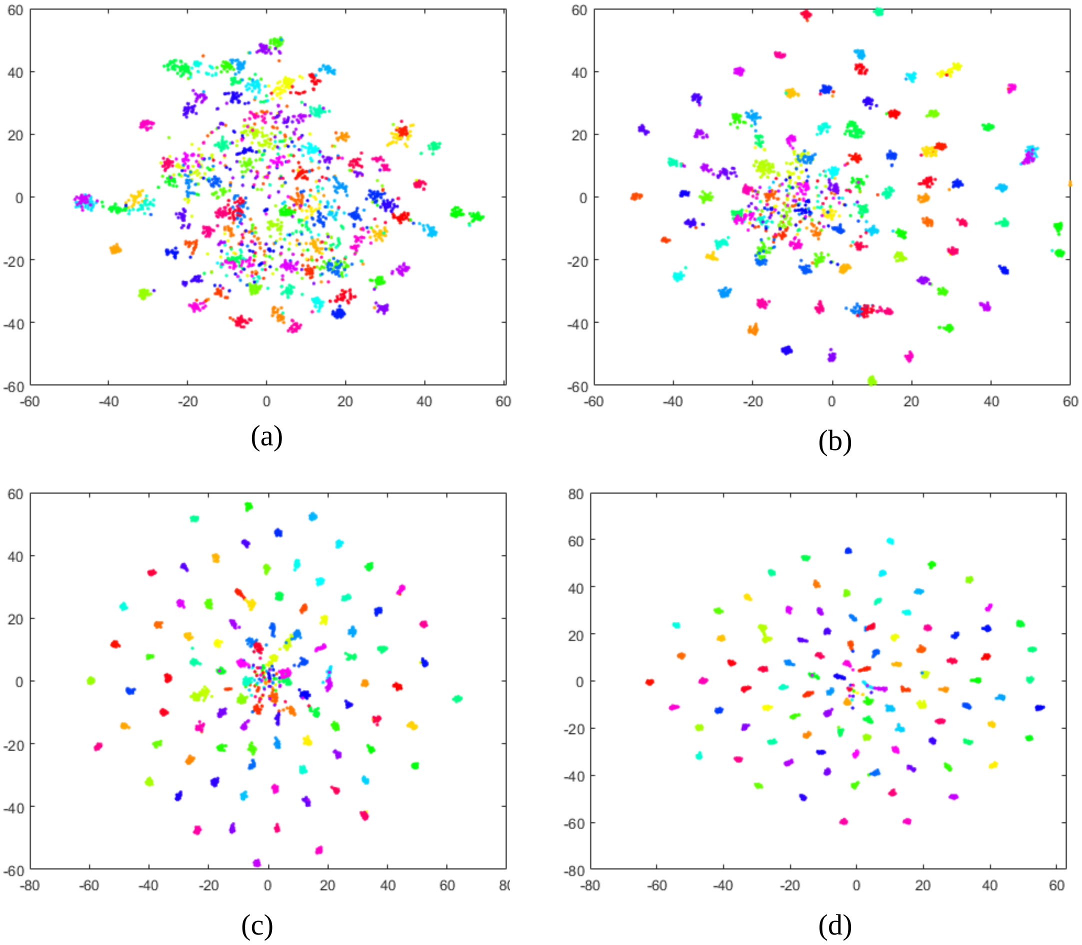


Figure 8.5 – The visualized t-SNE plot on Caltech-101 dataset. The t-SNE plots visualize features quality under four situations: (a) the concatenated raw feature, (b) the features refined by MLS, (c) the features refined by the proposed RML-MP, and (d) the features reinforced by the proposed SRML-MP.

8.3.7 Limitations

Although the proposed RML-MP and SRML-MP show superior performance among existing multilayer MP inverse-based algorithms on large-scale datasets, some limitations remain. First, the models in this chapter are only structured with multiple AEs. Without convolutional layers, the feature learning capacity of RML-MP and SRML-MP is limited. In other words, when handling big datasets, they need feature extraction algorithms, such as DCNNs, to get the raw features. These models, therefore, can be considered as feature reinforcement models instead of feature extractors. Second, the existing multilayer MP inverse-based networks, even the proposed RML-MP and SRML-MP, learn their models with two separate steps: unsupervised learning that is applied as network pre-training and supervised learning for final pattern classification. However, in complex tasks, it is not efficient. One practical way is to combine supervised with unsupervised learning in one deep network simultaneously, such as a semi-supervised ladder network [36].

8.4 Conclusion

The chapter proposes two multilayer neural networks for multi-model large data analysis. The RML-MP and SRML-MP are developed to enhance the generalization capability of the traditional multilayer MP inverse-based structures. The contributions of this chapter are as follows: First, representations learned from traditional least squares-based autoencoders may be biased and inadequate for solving complex tasks (such as ImageNet). In this chapter, the retraining strategy is proposed to enhance the representation capacity of one model. Second, the proposed RML-MP and SRML-MP handle big data efficiently. These models process large-scale datasets such as Places-365 containing more than 1.8 million samples with competitive performance and affordable training time. The experiments on 8 datasets ranging from image classification and food categorization show that the RML-MP and SRML-MP provide superior performance to the existing multilayer MP inverse-based representation learning algorithms.

Chapter 9

Fast Domain Transfer Learning for Application Towards Efficient Pattern Recognition

Domain transfer learning is a technique that exploits a pre-trained model from a source domain for a related task in a target domain to achieve better performance when the target domain lacks labelled data. Recently, MP inverse-based parameter fine-tuning of pre-trained fully-connected (FC) layers has emerged in transfer learning. However, due to the stringent computational requirements, such an approach is yet to gain much traction for practical applications. In this chapter, we address this issue through a novel fast retraining strategy, which greatly enhances the applicability of the MP inverse-based retraining of DCNNs. Specifically, in each retraining epoch, we employ a random layer freezing strategy to control the number of layers to be fine-tuned, and an MP inverse-based batch-by-batch refinement of dense layer parameters. This strategy greatly enhances the convergence time required for retraining the DCNNs. The experimental results show that the proposed retraining pipeline achieves competitive performance with quicker model convergence compared to the conventional transfer learning approaches. For instance, the proposed approach converges nearly 1.5 times faster when retraining an ImageNet pre-trained ResNet-50 on the Places-365 dataset.

9.1 Introduction

Transfer learning has been a very effective strategy in machine learning (ML) for various applications, such as image classification [238, 14, 173, 239], nature language processing [47, 240], image captioning [48], and object detection [49, 241]. Taking a pre-trained DCNN, and fine-tuning its layers is one of the widely adopted transfer learning approaches. It can be done in two ways [47, 242, 243]: i) retraining only a few selected layers, particularly the top FC layers with replacement of a custom output layer, and ii) full model end-to-end fine-tuning using target domain labelled samples. In end-to-end fine-tuning, all layer parameters are trained, resulting in accurate predictions, but demanding more time and large-scale datasets to avoid the overfitting issue. Therefore, ML solution developers take advantage of the first approach that retrains only a few selected layers. In this direction, recent research has utilized the MP inverse-based technique to retrain the densely-connected few layers of DCNNs with any standard optimizers such as stochastic gradient descend with momentum (SGDM) [38], root mean square propagation (RMSProp) [244], and adaptive momentum estimation (Adam) [39] to achieve better generalization performance.

Mathematically, the parameter optimization of the FC layers can be done through LS estimation as in any linear system, whereby the optimal parameters of these layers correspond to the LS solutions. The MP inverse is the most widely used technique to find the unique solution to any LS problems with matrix representation [211, 245, 246, 247]. The earliest work utilizing LS strategy in an artificial neural network (ANN) can be traced back to 1992, where Schmidt *et al.* [26] proved that the output layer weights can be calculated using the MP inverse technique, and the information encoded through this process called Fishier vector. Following this, an increased focus has been placed on MP inverse-based hierarchical networks [173, 248]. To the best of our knowledge, Yang *et al.* [27] is the state-of-the-art algorithm that utilizes the MP inverse in transfer training. In each training epoch, a pre-trained DCNN, such as AlexNet [50], GoogLeNet [87] or DenseNet [176], is first fine-tuned with a standard optimizer on the target domain; then, the parameters of the FC layers are refined through MP inverse computation. By doing so, the fine-tuned DCNN achieves high-level generalization performance. Despite its advantage, it is not as widespread as it could be. The limitation is caused due to two reasons: stringent computational requirement and inefficiency of all layers fine-tuning.

1) Stringent demand for computational resources: One of the drawbacks in Yang *et al.* [27] is that it significantly increases the computational workload. In fact, this model needs high computational resources, such as a workstation with 256 GB main memory to process the big datasets like Places-365 [35] which consists of more than 1.8 million samples for visual classification. The reason is that the traditional MP inverse used in Yang *et al.* [27] is essentially a one-batch learning algorithm that requires processing the input data at once [249, 250]. When utilizing ImageNet pre-trained VGG-16 [46], Yang *et al.* [27] requires around 200 GB main memory to perform MP inverse for the FC layers, which is infeasible for regular commercial computers and laptops.

2) Inefficiency of all layers fine-tuning: Yang *et al.* [27] approaches the model transfer as an end-to-end all layer retraining. Judging from some successful techniques [135, 251], we hypothesize in this chapter that it is not necessary to involve the entire convolutional layers in each retraining epoch. Thus, we randomly select a few layers in the retraining process. It will help us to overcome the over-fitting issues that generally happen when training DCNNs on small-scale datasets [135]. Some works [251] accelerate the retraining process through complex hyper-parameter adjustment in the optimizers. Although it accelerates the retraining process, it faces a huge performance degradation.

To address the afore-said shortcomings, this work proposes a novel retraining pipeline called MP inverse-based fast retraining (MPFR), which is shown in Fig. 9.1. The proposed MPFR has notable advantages in terms of learning efficiency and applicability. It is built upon two strategies: Strategy 1 - random layer freezing, and Strategy 2 - batch-by-batch FC layer feature refinement. The first strategy employs a fast retraining strategy to speed up the fine-tuning process of the DCNN. Here, we propose a mechanism that randomly activates an r_a portion of the DCNN layers in every retraining epoch. The rate r_a is a preset value and progressively decreased. In the first several retraining epochs, r_a is set to 1 to fully retrain the network, i.e., all of the parameters of the DCNN are fine-tuned. Then, r_a is gradually decreased to 0.2 at the last retraining epoch, i.e., 80% of the DCNN layers do not require parameter updates. The second strategy is the MP inverse-based batch-by-batch learning algorithm. Essentially, the batch-by-batch algorithm processes dense layer refinement sequentially, instead of the memory-hungry refinement models like in [27] that process the entire input data at once. Thus, the proposed MPFR framework does not necessarily need high-end hardware resources.

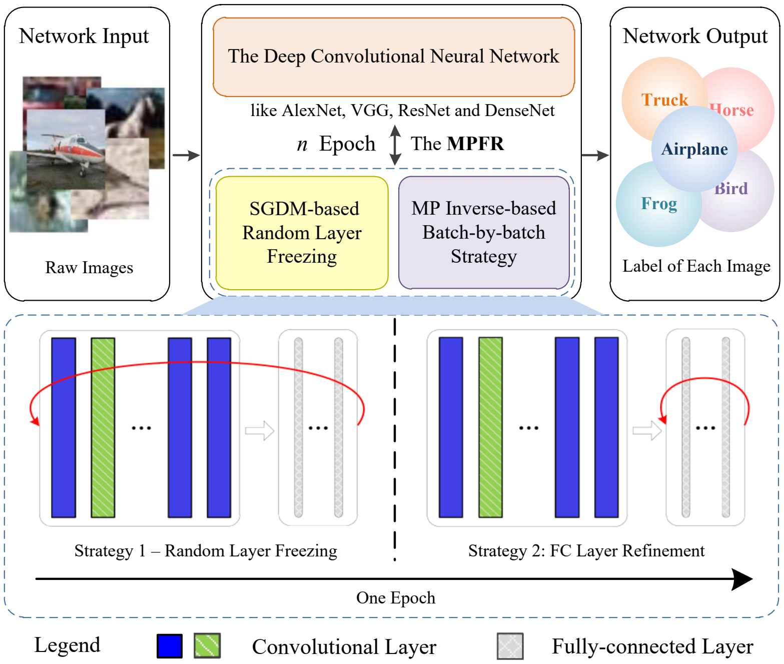

Network Input
Raw Images
The Deep Convolutional Neural Network
like AlexNet, VGG, ResNet and DenseNet
n Epoch
The MPFR
SGDM-based Random Layer Freezing
MP Inverse-based Batch-by-batch Strategy
Network Output
Truck
Horse
Airplane
Frog
Bird
Label of Each Image
Strategy 1 – Random Layer Freezing
Strategy 2: FC Layer Refinement
One Epoch
Legend
Convolutional Layer
Fully-connected Layer

Figure 9.1 – Abstract data flow diagram of the proposed MPFR with one possible optimizer in Strategy 1: SGDM.

In the extensive experiments, several ImageNet pre-trained DCNNs are utilized to verify the effectiveness of the proposed MPFR pipeline. The contributions of this chapter are threefold:

- **Applicability:** The batch-by-batch algorithm is utilized to ensure that the proposed model can support any hardware environment.
- **Efficiency:** The random layer freezing is utilized to decrease fine-tuning workloads of the MP inverse-based refinement strategy and to avoid over-fitting.
- **Extensive experimental analysis:** We conduct through experimental study using six benchmark datasets, including the Places-365 to show the robustness of the proposed pipeline.

9.2 Related Works on Domain Transfer Learning

In general, the conventional domain transfer learning approaches are either task-driven or data-driven [238]. Hence, these approaches can be categorized into three groups of techniques according to Pan *et al.* [252], such as transductive transfer learning (TTL), unsupervised transfer learning (UTL) and inductive transfer learning (ITL).

In TTL, both source and target tasks are similar but the application domains are different. The labelled data are available only in the source domain, but not in the target domain; thus, the framework can be conceived as semi-supervised learning. Adversarial training is one of the well-known examples of TTL. For instance, Kamnitsas *et al.* [253] utilized a domain adaptation method for image segmentation by using adversarial training of two 3D neural networks. The domain-invariant features are learned using 3D models, and a multi-connected domain discriminator is applied to segment the input image. Moreover, Zhang *et al.* [254] studied an adversarial domain adaptation from whole slide images (WSI) to microscopy images. In particular, a novel deep microscopy adaptation network (DMAN) is proposed to conduct a WSI-trained structure to predict microscopy images.

As for UTL, the label information is unknown for both the source and the target domains. For example, Chen *et al.* [255] proposed the discriminative mapping transform (DMP) for unsupervised linear adaptation, where the constrained maximum likelihood linear regression (CMLLR) is operated in the feature space to estimate the model parameters. Similarly, Lee *et al.* [256] introduced a self-learning algorithm for unsupervised domain adaptation for spoken document summarization. A structured pseudo model is used to generate the initial summaries for the target documents. Then, these generated summaries serve as extra training examples for learning a new summarizer.

9.3 The Proposed Algorithm

The goal of this chapter is to develop an efficient domain transfer learning. It exploits two key strategies: Strategy 1 - random layer freezing, and Strategy 2 - batch-by-batch FC layer feature refinement.

9.3.1 MP inverse-based Dense Layer Refinement

In this subsection, we review the learning algorithm proposed in [27]. Based on this strategy, we proposed our developed method.

In [27], the authors introduced a dense layer retraining algorithm for deep neural network fine-tuning. In each epoch, they retrain all the parameters of the DCNN using a standard optimizer, such as SGDM and Adam. Then, the conventional MP inverse is used to feedback the residual error $\boldsymbol{E}^n$ from the last FC layer before the final classifier layer to reestimate the weights of all the FC layers. The recomputation process can be divided into two steps, Step 1 - updating the last FC layer's weight $\hat{\boldsymbol{\alpha}}^n$ and Step 2 - recomputing the earlier FC layers' weight $\hat{\boldsymbol{\alpha}}^i$ (Notations are made available in Table 9.1).

Step 1 - Updating the last FC layer's weight $\hat{\boldsymbol{\alpha}}^n$: Firstly, the compensation weight $\boldsymbol{\eta}^n$ is computed in Eq. (9.1).

$$\boldsymbol{\eta}^n = (\boldsymbol{H}^n)^\dagger \boldsymbol{E}^n = \left((\boldsymbol{H}^n)^T(\boldsymbol{H}^n) + \frac{I}{C}\right)^{-1} (\boldsymbol{H}^n)^T \cdot \boldsymbol{E}^n, \tag{9.1}$$

where $\left((\boldsymbol{H}^n)^T(\boldsymbol{H}^n) + I/C\right)^{-1} (\boldsymbol{H}^n)^T$ is the MP inverse of $\boldsymbol{H}^n$. The weights of the last FC layer are updated by Eq. (9.2).

$$\begin{aligned}\hat{\boldsymbol{\alpha}}^n &= \boldsymbol{\alpha}^n + \gamma \cdot \boldsymbol{\eta}^n \\ &= \boldsymbol{\alpha}^n + \gamma \cdot \left(\left((\boldsymbol{H}^n)^T(\boldsymbol{H}^n) + \frac{I}{C}\right)^{-1} (\boldsymbol{H}^n)^T \cdot \boldsymbol{E}^n\right),\end{aligned} \tag{9.2}$$

where C is the regularization term, $\boldsymbol{\alpha}^n$ is the weights of n-th FC layer, $\hat{\boldsymbol{\alpha}}^n$ is the updated weights.

Step 2 - Recomputing the earlier FC layers' weight $\hat{\boldsymbol{\alpha}}^i$: After updating the weights of the last FC layer, we should recalculate the weights of the earlier FC layers. It is worth noting that before updating the weights of one FC layer, we need to calculate this layer's error. Take the i-th FC layer as an example, we have:

$$\boldsymbol{E}^i \cdot \hat{\boldsymbol{\alpha}}^{i+1} = \boldsymbol{E}^{i+1} \,|\, 1 \leq i < n. \tag{9.3}$$

Now using MP inverse, the error $\boldsymbol{E}^i$ can be computed as in Eq. (9.4).

$$\boldsymbol{E}^i = \boldsymbol{E}^{i+1} \left((\hat{\boldsymbol{\alpha}}^{i+1})^T(\hat{\boldsymbol{\alpha}}^{i+1}) + \frac{I}{C}\right)^{-1} (\hat{\boldsymbol{\alpha}}^{i+1})^T \tag{9.4}$$

Table 9.1 – Notations used in this chapter

Notation	Meaning
$\dagger$	the Moore-Penrose inverse
$\boldsymbol{\alpha}^i$	i-th FC layer weights (before MP inverse-based retraining)
$\hat{\boldsymbol{\alpha}}^i$	i-th FC layer weights (after MP inverse-based retraining)
γ	the retraining rate in Strategy 2
$\boldsymbol{\eta}_p^i$	compensation weights calculated with the first p batches of data in the i-th FC layer, $\boldsymbol{\eta}_p^i \in \mathcal{R}^{d^{i-1} \times d^i}$
C	the regularization term
d^i	the number of neurons in the i-th FC layer
$\boldsymbol{E}^i$	the i-th FC layer error feedback data, $\boldsymbol{E}^i \in \mathcal{R}^{N \times d^i}$
$\mathcal{F}$	the dropout operation
$\boldsymbol{E}_p$	the p-th batch of error data
$\boldsymbol{H}^i$	the input features of the i-th FC layer, $\boldsymbol{H}^i \in \mathcal{R}^{N \times d^{i-1}}$
$\boldsymbol{H}_p$	the p-th batch of input feature
I	the identity matrix
L_a	the number of activated convolutional layers
L_c	the total number of convolutional layers
L_i	the number of activated convolutional layers
M	the total number of mini-batches
N	the total number of training samples
N_p	the number of training samples in p-th data batch
n	the number of FC layer in one DCNN
r_a	the activation rate in Strategy 1
$\boldsymbol{T}^i$	the target of the i-th FC layer

Although Yang *et al.* [27] provide excellent performance on image classification, it has the following two limitations.

△ It can only be employed in a workstation as the parameters calculated by Eq. (9.2) demand huge computational resources. For example, when using VGG-16 on the Places-365 dataset, the method in [27] requires all the samples in the dataset to be processed once. In other words, we have to store all the samples' high dimensional feature maps, such as $\boldsymbol{H}^n$ and $\boldsymbol{E}^n$ in the main memory. For instance, the last dense layer's feature map $\boldsymbol{H}^n \in \mathcal{R}^{1,803,460 \times 4,096}$, and the peak memory usage in processing this feature map demands 218.9 GB.

△ Furthermore, Yang *et al.* [27]'s retraining algorithm is inefficient because it is a full-model end-to-end fine-tuning approach. It requires more retraining time and faces an overfitting issue on small-scale datasets.

9.3.2 Strategy 1 - Random Layer Freezing

It uses a simple random layer retraining algorithm as depicted in Fig. 9.2a to both avoid network over-fitting and speed up the fine-tuning. Specifically, the proposed method randomly freezes some hidden layers in each epoch using a preset activation rate r_a. Initially, the activation rate r_a is set to 1 in the first several fine-tuning epochs for warming up the pretraining model, whereby all the trainable parameters are updated in every backward pass. After the warm-up stage, the first several layers are capable of extracting low-level features that can be used by the following layers to build high-level features, and they are reliable enough to represent the target domain data. Thus, r_a is decreased, resulting in an increment of the number of inactivated layers in the subsequent epochs. Suppose that a DCNN architecture contains L_c total number of layers (including convolutional and fully-connected layers), the total number of activated (L_a) and inactivated (L_i) layers in each epoch are:

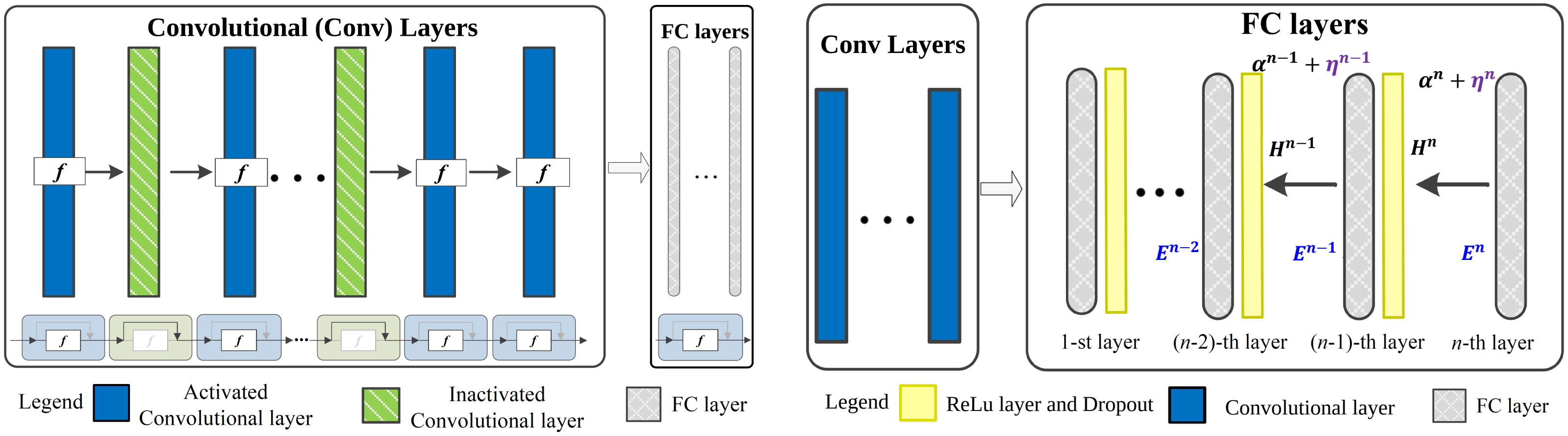
$$\begin{aligned} &L_a = r_a \times L_c, \text{ and} \\ &L_i = (1 - r_a) \times L_c. \end{aligned} \tag{9.5}$$

9.3.3 Strategy 2 - Batch-by-batch FC Layer Refinement

Suppose $\boldsymbol{H}^i$, $\boldsymbol{\alpha}^i$, $\boldsymbol{T}^n$, and $f_i(\cdot)$ are the input feature, weights of the i-th FC layer, the target of the last FC layer, and the i-th FC layer activation function, respectively. The objective function of the MP inverse-based FC layer retraining is defined as:

$$\begin{aligned} &\text{minimize} \quad J = \frac{1}{2}||\boldsymbol{E}^n||^2 = \frac{1}{2}||\boldsymbol{T}^n - f(\mathbf{H}, \boldsymbol{\alpha})||^2 \\ &f(\mathbf{H}, \boldsymbol{\alpha}) = f_n\left(\cdots f_2\left(f_1\left(\mathbf{H}^1, \boldsymbol{\alpha}^1\right), \boldsymbol{\alpha}^2\right) \cdots, \boldsymbol{\alpha}^n\right), \end{aligned} \tag{9.6}$$

where $f_i(\cdot)$ is a linear function. The MP inverse-based retraining aims to find the optimal weight $\hat{\boldsymbol{\alpha}}^i$ so that the loss function reaches the minimum. With the retraining rate γ, the weight $\boldsymbol{\eta}^i$ is used to compensate the network error, i.e., $\hat{\boldsymbol{\alpha}}^i = \boldsymbol{\alpha}^i + \gamma \cdot \boldsymbol{\eta}^i$.



Strategy 1 - Random layer freezing. In each epoch, users randomly activate L_a number of layers, while freezing the rest L_i number of layers from backward pass. L_a and L_i are determined by a hyperparameter r_a.

Strategy 2 - Batch-by-batch fully-connected layer feature refinement. $\boldsymbol{\eta}^n$ and $\boldsymbol{\eta}^{n-1}$ are obtained by Procedure I, and $\boldsymbol{E}^{n-1}$ and $\boldsymbol{E}^{n-2}$ are received via Procedure II. The detailed learning steps of Procedures I and II can be found in Algorithm 9.1.

Figure 9.2 – The proposed MPFR strategy. The DCNN is trained with several epochs consisting of two successive strategies: Strategy 1 is the random layer freezing, and Strategy 2 refers to the FC layer feature refinement.

For the i-th FC layer, the error-based compensation weight $\boldsymbol{\eta}^i$ satisfies:

$$\boldsymbol{H}^i(\boldsymbol{\alpha}^i + \gamma \cdot \boldsymbol{\eta}^i) = \boldsymbol{T}^i \,|\, 1 \le i \le n, \tag{9.7}$$

where $\gamma \in (0, 1]$. Based on Eq. (9.7), we have the following equation.

$$\boldsymbol{H}^i \cdot \boldsymbol{\eta}^i = \boldsymbol{E}^i \,|\, 1 \le i \le n, \tag{9.8}$$

where $\boldsymbol{E}^i$ is the matrix evaluating the gap between target $\boldsymbol{T}^i$ and the actual i-th layer output. The $\boldsymbol{\eta}^i$ is calculated by $\boldsymbol{\eta}^i = (\boldsymbol{H}^i)^\dagger \boldsymbol{E}^i$, where $(\boldsymbol{H}^i)^\dagger$ is the MP inverse of $\boldsymbol{H}^i$. However, the traditional MP inverse requires huge main memory in training because it generate the optimal FC layer weights on the entirety of the data.

To reduce the demand for computational resources, the batch-by-batch method proposed in Chapter 5 is utilized. By doing so, the $\boldsymbol{H}^i$ and $\boldsymbol{E}^i$ are processed chunk-by-chunk with M pieces. In other words, we use a sequential learning algorithm to handle the memory requirement in Eq. (9.2). First, the initial batch $\boldsymbol{H}_1$ and $\boldsymbol{E}_1$ is given, and the weights $\boldsymbol{\eta}_1$ is calculated by the traditional one-batch learning method. Then, the weights $\boldsymbol{\eta}_p$ are updated through $\boldsymbol{H}_p$, $\boldsymbol{E}_p$, and $\boldsymbol{\eta}_{p-1}$ in an iterative way. The iterative learning steps for $\boldsymbol{\eta}_p$ are expressed in Eq. (9.10).

$$\boldsymbol{\eta}_p = \begin{cases} 0, & p = 0 \\ U_p \boldsymbol{\eta}_{p-1} + W_p^{-1} \boldsymbol{H}_p^T \boldsymbol{E}_p, & 1 \le p \le M \end{cases} \tag{9.9}$$

where W_p^{-1}, U_p and N_p are obtained by

$$\begin{aligned} W_p^{-1} &= \begin{cases} \left[\frac{I}{C} + \boldsymbol{H}_1^T \boldsymbol{H}_1\right]^{-1}, & p = 1 \\ U_p W_{p-1}^{-1}, & 2 \le p \le M \end{cases} \\ U_p &= \begin{cases} 0, & p = 1 \\ I - N_p, & 2 \le p \le M \end{cases} \\ N_p &= W_{p-1}^{-1} \boldsymbol{H}_p^T \left(\boldsymbol{H}_p W_{p-1}^{-1} \boldsymbol{H}_p^T + I \right)^{-1} \boldsymbol{H}_p, \end{aligned} \tag{9.10}$$

Based on Eq. (9.10), for a specific FC layer, the weights are initialized using the first batch of data ($\boldsymbol{H}_1$) as in traditional MP inverse-based refinement approach. Then, they are updated using the remaining batches ($\boldsymbol{H}_p$, $1 < p \le M$) sequentially.

After the network is retrained with M batches of data, the parameters $\boldsymbol{\alpha}^n$ in the last (n-th FC) layer can be updated as in Eq. (9.11):

$$\hat{\boldsymbol{\alpha}}^n = \boldsymbol{\alpha}^n + \gamma \cdot \boldsymbol{\eta}_M^n, \tag{9.11}$$

where γ is the retraining rate. Similarly, with dropout $\mathcal{F}$ and activation function ReLU, the parameters $\boldsymbol{\alpha}^i$ in the earlier i-th FC layer with M batches data can be updated via Eq. (9.12):

$$\hat{\boldsymbol{\alpha}}^i = \mathcal{F}\left(\mathrm{ReLU}(0, \boldsymbol{\alpha}^i + \gamma \cdot \boldsymbol{\eta}_M^i)\right) \tag{9.12}$$

The proposed domain transfer learning pipeline is summarized in Algorithm 9.1: Lines 2-5 define the random layer freezing (Strategy 1). Here, we use the SGDM as an example to fine-tune the model. Lines 6-16 define the batch-by-batch FC layer feature refinement (Strategy 2). Strategy 2 depends on two call back procedures: Procedure I, batch-by-batch learning, and Procedure II, error backpropagation to refine the dense layer features and feedback the error, respectively.

9.4 Experimental Results

9.4.1 Experimental Setup

9.4.1.1 The Environment

The experiments were conducted in MATLAB 2019b on a workstation with a 256 GB memory and an E5-2650 processor, and all of the DCNNs were trained on NVIDIA 1080Ti GPU. Furthermore, the mean average Top-1 testing accuracies were computed from a minimum of three trials.

9.4.1.2 The Datasets

The following visual and food image classification datasets, including Caltech-101/256, Cifar-100, Places-365, Places-365-1, and Food-251 are used for performance evaluation of the proposed pipeline.

Algorithm 9.1 The proposed MPFR

Inputs: Given a DCNN, maximum number of retraining epoch D, input dataset $\mathbf{X}$, the last (n-th) FC layer target $\boldsymbol{T}^n$

Outputs: A well-trained DCNN

```
for $(j = 1, j <= D, j++)$ do
    • Strategy 1: Random Layer Freezing
    Randomly select $L_i$ layers form the DCNN.
    Inactivate layers. %Freeze layers.
    Train with SGDM for one epoch. % Train the DCNN with SGDM optimizor.
    • Strategy 2: Batch-by-batch FC Layer Feature Refinement
    $\boldsymbol{H}^n$, $\boldsymbol{\alpha}^n$ % Extract from the last ($n$-th) FC layer.
    $\boldsymbol{E}^n = \boldsymbol{T}^n - \boldsymbol{H}^n \cdot \boldsymbol{\alpha}^n$
    Procedure I: $\boldsymbol{\eta}^n \leftarrow$ Batch-by-batch learning $(\boldsymbol{H}^n, \boldsymbol{E}^n)$
    $\hat{\boldsymbol{\alpha}}^n = \boldsymbol{\alpha}^n + \gamma \cdot \boldsymbol{\eta}^n$ % Update the last FC layer.
    for $(i = n-1, i >= 1, i--)$ do
        $\boldsymbol{H}^i$, $\boldsymbol{\alpha}^i$ % Extract from the $i$-th FC layer.
        Procedure II: $\boldsymbol{E}^i \leftarrow$ Error Backpropagation $(\boldsymbol{E}^{i+1}, \hat{\boldsymbol{\alpha}}^{i+1})$
        Procedure I: $\boldsymbol{\eta}^i \leftarrow$ Batch-by-batch learning $(\boldsymbol{H}^i, \boldsymbol{E}^i)$
        $\hat{\boldsymbol{\alpha}}^i = \mathcal{F}\left(\text{ReLU}(0, \boldsymbol{\alpha}^i + \gamma \cdot \boldsymbol{\eta}^i_M)\right)$ % Update the parameters in the $i$-th FC layer, $\mathcal{F}$ is the dropout operation.
    end for
end for
```

Procedure I: *Batch-by-batch Learning* $(\boldsymbol{H}, \boldsymbol{E})$

Inputs: $\boldsymbol{H} = [\boldsymbol{H}_1^T, \boldsymbol{H}_2^T, \cdots, \boldsymbol{H}_M^T]^T$, $\boldsymbol{E} = [\boldsymbol{E}_1^T, \boldsymbol{E}_2^T, \cdots, \mathbf{E}_M^T]^T$

Outputs: $\boldsymbol{\eta}_M$

```
$W_1^{-1} = [\frac{I}{C} + \boldsymbol{H}_1^T \boldsymbol{H}_1]^{-1}$, $U_1 = 0$
$\boldsymbol{\eta}_1 = W_1^{-1} \boldsymbol{H}_1^T \boldsymbol{E}_1$ % Weights updated with the first data batch.
for ($p$=2, $p < M$, $p++$) do
    $N_p = W_{p-1}^{-1} \boldsymbol{H}_p^T (\boldsymbol{H}_p W_{p-1}^{-1} \boldsymbol{H}_p^T + I)^{-1} \boldsymbol{H}_p$
    $U_p = I - N_p$
    $W_p^{-1} = U_p W_{p-1}^{-1}$
    $\boldsymbol{\eta}_p = U_p \boldsymbol{\eta}_{p-1} + W_p^{-1} \boldsymbol{H}_p^T \boldsymbol{E}_p$ % Sequential learning.
end for
```

Procedure II: *Error Backpropagation* $(\boldsymbol{E}^i, \boldsymbol{\alpha}^i)$

Inputs: $\boldsymbol{E}^i = [(\boldsymbol{E}_1^i)^T, (\boldsymbol{E}_2^i)^T, \cdots, (\boldsymbol{E}_M^i)^T]^T$, $\boldsymbol{\alpha}^i$

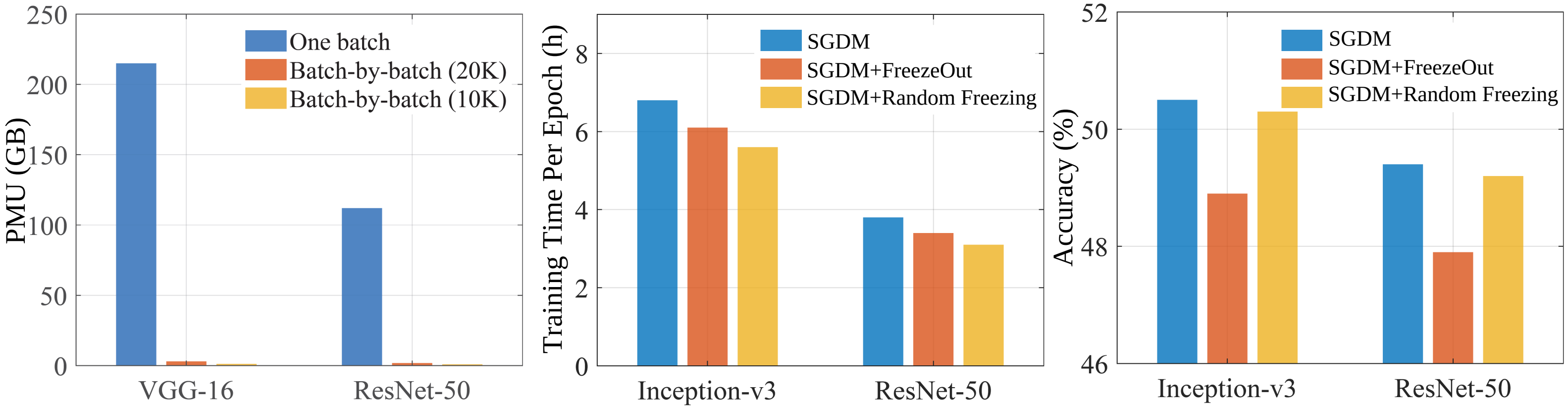
Outputs: $\boldsymbol{E}^{i-1}$

```
for $(p = 1, p <= M, p++)$ do
    $\boldsymbol{E}_p^{i-1} = \max\left(0, \boldsymbol{E}_p^i \cdot \left(\frac{I}{C} + (\boldsymbol{\alpha}^i)^T \boldsymbol{\alpha}^i\right)^{-1} \cdot (\boldsymbol{\alpha}^i)^T\right)$ % Error matrix back-propagation using MP inverse.
end for
$\boldsymbol{E}^{i-1} = [(\boldsymbol{E}_1^{i-1})^T, (\boldsymbol{E}_2^{i-1})^T, \cdots, (\boldsymbol{E}_M^{i-1})^T]^T$;
```

Table 9.2 – Hyperparameter settings of the proposed method and the compared rival algorithms

Methods	Parameters
Bottou *et al.* (SGDM) [38]	BS = 32, ILR = grid searched within $\{1.0^{-2}, 1.0^{-3}, 1.0^{-4}\}$, TE = 8, DR = 0.1 per 3 epochs, and the momentum = 0.9.
Kingma *et al.* (Adam) [39]	BS = 32, ILR = grid searched within $\{1.0^{-2}, 1.0^{-3}, 1.0^{-4}\}$, TE = 8, DR = 0.1 per 3 epochs, gradient decay factor $\beta_1 = 0.9$ and $\beta_2 = 0.999$.
Kurbiel *et al.* (RMSprop) [244]	BS = 32, ILR = grid searched within $\{1.0^{-2}, 1.0^{-3}, 1.0^{-4}\}$, TE = 8, DR = 0.1 per 3 epochs, and the exponential decay rate for squared gradient moving average = 0.9.
Brock *et al.* (FreezeOut) [251]	BS = 32, ILR = grid searched within $\{1.0^{-2}, 1.0^{-3}, 1.0^{-4}\}$, TE = 8, DR = 0.1 per 3 epochs, freezing time $t_0 = 0.5$, and momentum = 0.9.
Yang *et al.* [27]	BS = 32, ILR = grid searched within $\{1.0^{-2}, 1.0^{-3}, 1.0^{-4}\}$, TE = 8, DR = 0.1 per 3 epochs, momentum = 0.9, $\beta_1 = 0.9$ and $\beta_2 = 0.999$, C = grid searched within $\{10^{-3}, 10^{0}, 10^{3}\}$, and $\gamma = 0.2$.
Ours (MPFR)	BS = 32, ILR = grid searched within $\{1.0^{-2}, 1.0^{-3}, 1.0^{-4}\}$, TE = 8, DR = 0.1 per 3 epochs, initial r_a is 1, and it is set to 0.75, 0.5, and 0.25 at 25%, 50%, and 75% of TE, respectively. In Strategy 2, C = grid searched within $\{10^{-3}, 10^{0}, 10^{3}\}$, $\gamma = 0.2$, and batch size in MP inverse = 20 K.

Notations: BS - batch size, ILR - initial learning rate, TE - total number of retraining epochs, DR - decay rate.



(a) Peak memory usage on Places-365 (b) Training time per epoch on Places-365-1 (c) Top-1 testing accuracy on Places-365-1

Figure 9.3 – Experimental results of the proposed batch-by-batch FC layer refinement strategy and fast retraining. (a) is the ablation results of the proposed batch-by-batch algorithm. The analysis of this part is elaborated in Section 9.4.2.1. (b) (c) are the ablation results of the proposed random layer freezing schedule. The details of experiments is explained in Section 9.4.2.2

9.4.1.3 The Rival Methods

In this chapter, the following state-of-the-art domain transfer learning algorithms, such as Bottou *et al.* (SGDM) [38], Kingma *et al.* (Adam) [39], Kurbiel *et al.* (RMSProp) [244], FreezeOut [251], and Yang *et al.* [27] are applied as the baselines for comparison.

The MPFR is validated on several state-of-the-art ImageNet pre-trained DCNNs, including AlexNet [50], VGG-16 [46], Inception-V3 [237], ResNet-50 [88] and DenseNet-201 [176].

9.4.1.4 Configurations of The Rival Methods

The detailed hyperparameter settings of the proposed work and the compared rival methods are summarized in Table 9.2 for redroducibility. It is worth noting that we do not adopt any data augmentations in all our experiments.

9.4.2 Step-by-step Quantitative Analysis

9.4.2.1 Analysis of Batch-by-Batch FC Layer Refinement

To verify the effectiveness of applying an MP inverse-based batch-by-batch scheme in FC layer retraining, a sanity check is conducted to compare the proposed sequential learning strategy and the one-batch schedule [27] as tabulated in Fig. 9.3a and Table 9.3. This experiment is only intended to focus on validating the proposed batch-by-batch strategy. Thus, the r_a in each epoch (cf. Strategy 1 in Fig. 9.2a) is kept fixed ($r_a = 1$) while conducting different learning strategy in Strategy 2 (one-batch/batch-by-batch) listed in Table 9.3. Along with the Top-1 testing accuracy and the average training time of MP inverse in each epoch, peak memory usage (PMU) in fine-tuning is also used to evaluate the performance of different schemes. The investigation reveals that i) the batch-by-batch strategy significantly reduces the memory usage in retraining, and ii) the one-batch and batch-by-batch learning strategy show similar testing accuracy across all the datasets while having around 10% more processing time. Therefore, we can summarize the ***first conclusion***: The provided batch-by-batch method significantly reduces the computational burden while main-

Table 9.3 – Performance comparison of the proposed batch-by-batch method on VGG-16: Tr. (h) - average retraining time of MP inverse per epoch, Acc. (%) - Top-1 testing accuracy, PMU (GB) - peak memory usage, and N/batch - the batch-by-batch proposed pipeline with N samples per batch

Datasets	Type	Tr. (h)	Acc. (%)	PMU (GB)
Places-365-1	One-batch	1.2	50.68	87.2
	$20K$/batch	1.3	50.42	3.2
	$10K$/batch	1.4	50.40	1.6
Places-365	One-batch	3.2	55.21	218.9
	$20K$/batch	3.5	55.38	3.2
	$10K$/batch	3.7	55.32	1.6

Table 9.4 – Performance comparison of the proposed random layer freezing on ResNet-50: Tr. (h) - average retraining time per epoch in hours, Acc. (%) - Top-1 testing accuracy

Methods	Places-365-1		Places-365	
	Tr. (h)	Acc. (%)	Tr. (h)	Acc. (%)
SGDM	3.7	49.3	11.9	53.3
SGDM + FreezeOut	3.3	47.5	10.6	50.3
SGDM + Random freezing	2.8	49.1	9.1	52.4
Adam	4.4	49.6	13.4	53.6
Adam + FreezeOut	3.6	47.5	11.9	51.2
Adam + Random freezing	3.1	49.2	10.7	52.9

taining competitive performance on different datasets. Hence, this method allows the dense layer retraining to be utilized in any environment, which overcomes the main drawback of Yang *et al.* [27].

9.4.2.2 Analysis of Random Layer Freezing

To validate the effectiveness of random layer freezing, more ablation experiments are conducted on the Places-365-1 dataset. The experiments are conducted and trained under three different configurations: DCNN trained with i) the SGDM baseline, ii) the SGDM with FreezeOut [251] scheme, and iii) the SGDM with the proposed method.

Table 9.5 – Top-1 testing accuracy comparison of various domain transfer learning approaches

Architecture	Model	Caltech-101	Caltech-256	Cifar-100	Food-251	Places-365-1	Places-365	Average
VGG-16	Bottou *et al.* (SGDM) [38]	89.2	69.3	77.4	52.7	49.9	54.0	65.4
	Kingma *et al.* (Adam) [39]	90.8	72.0	78.2	54.1	49.0	55.0	66.5
	Kurbiel *et al.* (RMSProp) [244]	88.8	69.5	77.6	53.5	48.4	53.8	65.3
	FreezeOut [251] + SGDM[a]	87.0	68.5	77.0	51.8	48.2	52.6	64.2
	Yang *et al.* [27] + SGDM[b]	90.1	73.0	**79.5**	56.4	**50.6**	**55.5**	67.7
	Ours + SGDM[c]	**91.5**	**73.7**	79.2	**57.4**	50.5	55.4	**67.8**
ResNet-50	Bottou *et al.* (SGDM) [38]	89.6	75.4	82.3	59.4	49.3	53.3	68.3
	Kingma *et al.* (Adam) [39]	88.5	78.5	81.4	60.3	49.6	53.6	68.6
	Kurbiel *et al.* (RMSProp) [244]	89.1	75.5	84.2	59.2	49.1	53.2	68.4
	FreezeOut [251] + SGDM[a]	88.2	74.7	83.1	57.1	47.5	50.3	66.8
	Yang *et al.* [27] + SGDM[b]	91.0	79.5	83.6	**61.8**	**50.5**	**54.6**	70.1
	Ours + SGDM[c]	**91.5**	**80.3**	**83.7**	61.6	50.4	**54.6**	**70.4**
Inception-V3	Bottou *et al.* (SGDM) [38]	89.7	75.3	83.8	58.3	50.5	53.2	68.6
	Kingma *et al.* (Adam) [39]	89.1	81.0	82.9	60.4	49.2	53.7	69.4
	Kurbiel *et al.* (RMSProp) [244]	89.3	75.2	83.4	59.8	50.5	53.5	68.6
	FreezeOut [251] + SGDM[a]	87.6	74.9	83.0	59.3	49.1	52.4	67.7
	Yang *et al.* [27] + SGDM[b]	91.3	81.4	84.2	61.3	**51.3**	**54.7**	**70.9**
	Ours + SGDM[c]	**91.6**	**81.9**	**84.3**	**61.8**	51.1	54.4	70.8
DenseNet-201	Bottou *et al.* (SGDM) [38]	**92.5**	79.7	**86.1**	61.6	49.7	54.7	70.7
	Kingma *et al.* (Adam) [39]	91.2	81.0	84.2	61.9	49.5	53.2	70.2
	Kurbiel *et al.* (RMSProp) [244]	91.6	80.3	85.5	60.7	49.6	53.9	70.3
	FreezeOut [251] + SGDM[a]	89.9	77.5	84.1	59.8	47.2	51.3	68.3
	Yang *et al.* [27] + SGDM[b]	91.0	81.2	85.4	62.4	50.6	**55.4**	**71.1**
	Ours + SGDM[c]	91.7	**81.6**	85.2	**62.5**	**50.7**	55.0	**71.1**

[a] FreezeOut [251] + SGDM: The SGDM is used for fine-tuning, and the FreezeOut [251] is utilized to speed up training.

[b] Yang *et al.* [27] + SGDM: The SGDM is used for fine-tuning, and the Yang *et al.* [27]'s method is utilized to refine the FC layer features.

[c] Ours + SGDM: The SGDM is used for fine-tuning, and our proposed strategy (MPFR) is utilized to refine the FC layer features.

This analysis is intended to focus on evaluating the performance of the proposed random layer freezing. Hence, in each epoch, random layer freezing (strategy 1) is employed while the retraining strategy (strategy 2) is excluded from the training. Fig. 9.3b, 9.3c and Table 9.4 compare the average retraining time per epoch and the Top-1 testing accuracy. Through analysis, we reach the ***second conclusion***: The retraining time of the proposed random layer freezing speeds up the training by more than 30%, and it has a mild impact on the testing performance.

9.4.2.3 Comparison of Transfer Learning

Taking the above outcomes as the foundation, more experiments are carried out to compare the MPFR and the other transfer learning algorithms. The experimental results of all the datasets are tabulated in Table 9.5. Through observation, one can conclude the following: i) Compared to Yang *et al.* [27] + SGDM pipeline, the Top-1 testing accuracy of the proposed algorithm (Ours + SGDM) is improved by 0.3% to 1.4% on the small-scale datasets while showing a competitive performance on the large-scale datasets. ii) Compared to the existing transfer learning method, the proposed algorithm (Ours + SGDM) has a remarkable improvement in terms of the Top-1 testing accuracy. For example, for the VGG-16, the MPFR improves the Top-1 accuracy by 1.4% to 4.7% over Bottou *et al.* [38].

Fig. 9.4 shows the generalization performance of the MPFR with SGDM and Adam on Cifar-100 and Places-365-1 by a plot as the number of fine-tuning epochs increases. It is worth noting that the convergence point (CoP) of each method is shown as well. Hence, we can easily summarize that the proposed MPFR converges at 4 to 5 learning epochs, whereas the SGDM needs 7 to 8 epochs to reach its stable state. Further, Table 9.6 tabulates the model convergence time of the proposed MPFR, Yang *et al.* [27] and Bottou *et al.* [38] on Cifar-100, Places-365-1 and Places-365 datasets. The proposed MPFR speeds up the retraining process nearly by 10% better than of Yang *et al.* approach [27]. This amount of time-saving is contributed by the proposed random freezing (Strategy 1). Hence, from the results presented in Table 9.5 and Fig. 9.4, we reach the ***third conclusion*** that the proposed MPFR strategy outmatches the compared algorithms wrt the evaluation metrics including the complexity and applicability.

Furthermore, Fig. 9.5 draws the Top-1 testing accuracy wrt various initial learning rates: 1.0^{-2}, 1.0^{-3}, and 1.0^{-4} on Caltech-101 and Places-365-1 datasets. Note that for Yang *et al.* [27] and MPFR, the SGDM is utilized for fine-tuning. For completeness, we carried out additional experiments with basic two optimizers and different learning rates to analyze the proposed pipelines' performance as shown in Fig. 9.6. Where, the SGDM and Adam are utilized for DCNN fine-tuning, respectively. The experiments were conducted twenty times independently for a comprehensive comparison. In both experiments, the configurations, such as decay rate and batch size were kept fixed for consistency. Based on the experimental results shown in Fig. 9.5 and Fig. 9.6, we reach the ***fourth conclusion*** that the proposed MPFR algorithm consistently produce the best classification accuracy regardless of the initial learning rate.

9.5 Conclusion

In this chapter, a unified fast retraining procedure for domain transfer learning is proposed. This method achieves better generalization performance than the state-of-the-art transfer learning strategies. In particular, it provides i) a random freezing schedule to speed up the retraining process and prevent over-fitting, and ii) a batch-by-batch MP inverse-based strategy to optimize the parameters of the dense layers. The experimental results on benchmark datasets prove the effectiveness of the proposed algorithm. It has the following three main advantages: i) It alleviates overfitting problems on small-scale datasets. Also, it can provide competitive performance on large-scale datasets. ii) It doesn't require a high demand for computational resources. iii) It achieves a quicker retraining convergence time over the traditional approaches. The proposed strategy can be easily adapted in many industrial applications, such as power line fault detection in diagnosis and anomaly detection for Internet of Things (IoT) applications.

As for future direction, it would be worthwhile to use the proposed MP inverse strategy to train all the layers in DCNNs.

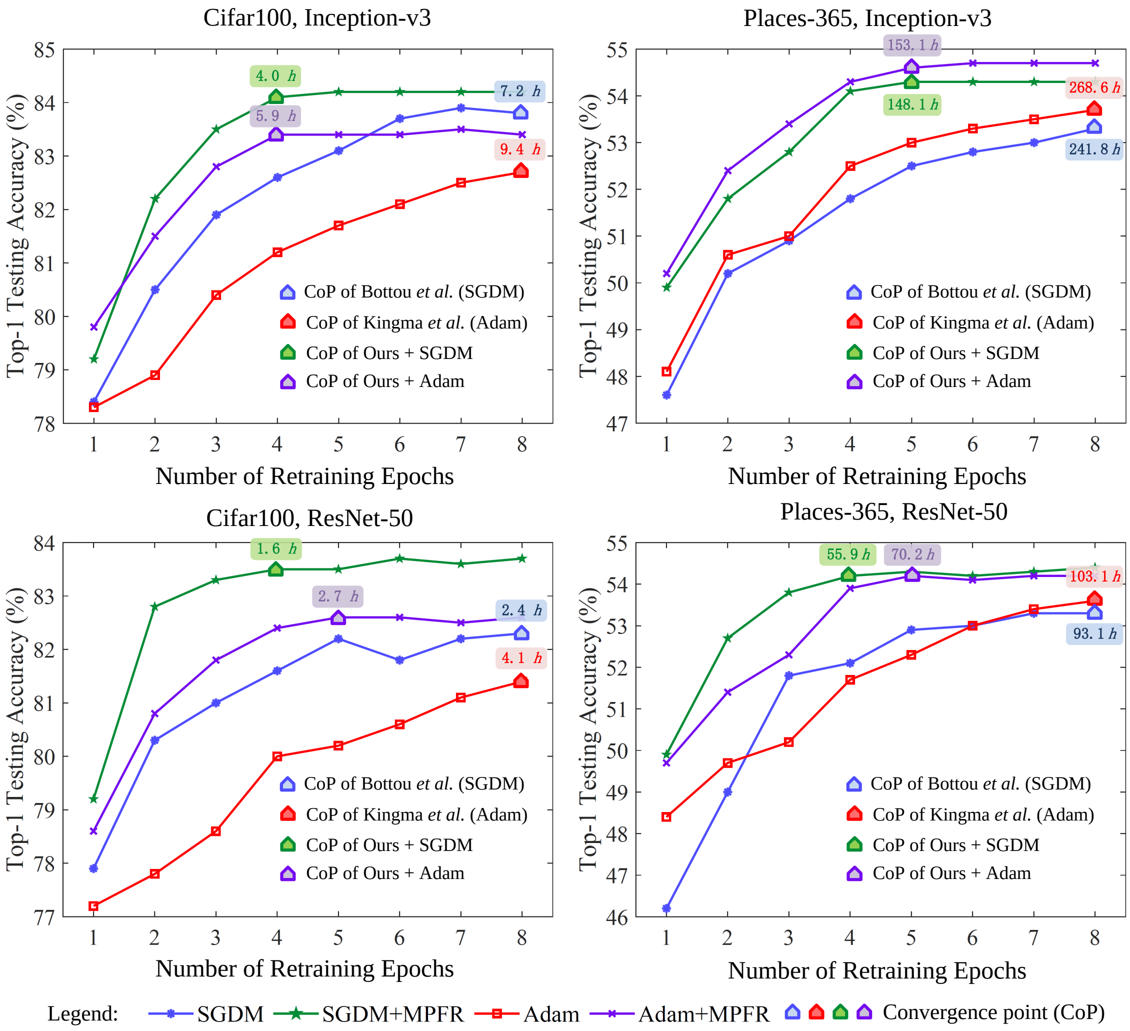
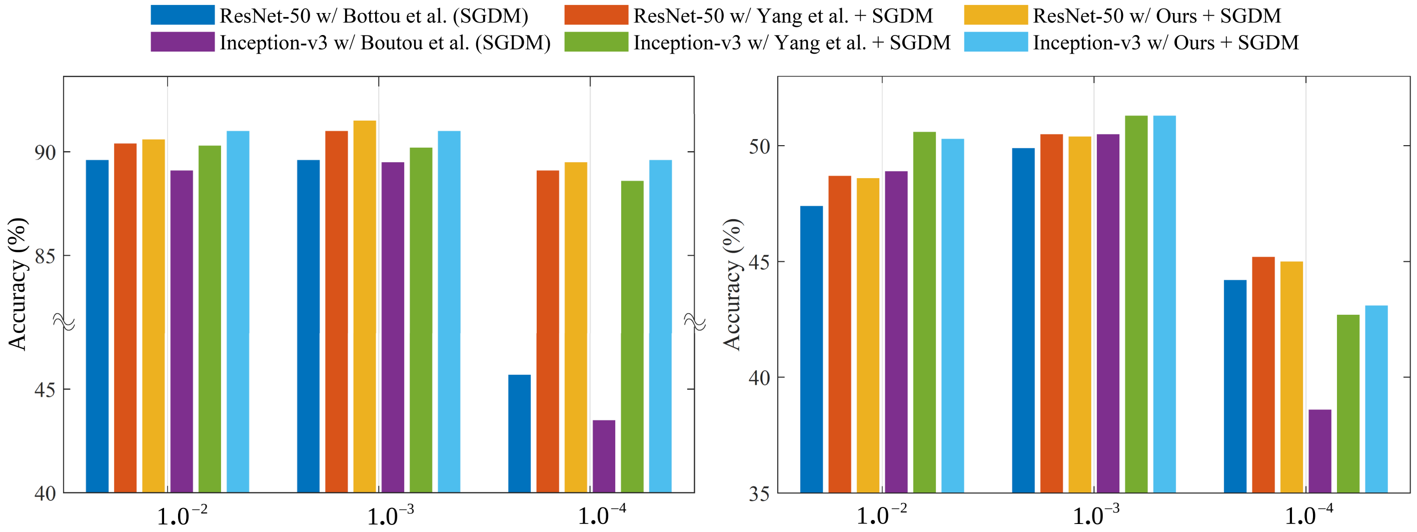


Figure 9.4 – Top-1 accuracy of various domain transfer learning approaches on Cifar-100 and Places-365 datasets. Note: the values shown in the highlighted boxes are the total convergence time in hours. For example, the proposed pipeline in this work takes 55.9 hours for retraining convergence for ResNet-50 with SGDM on Places-365 dataset.

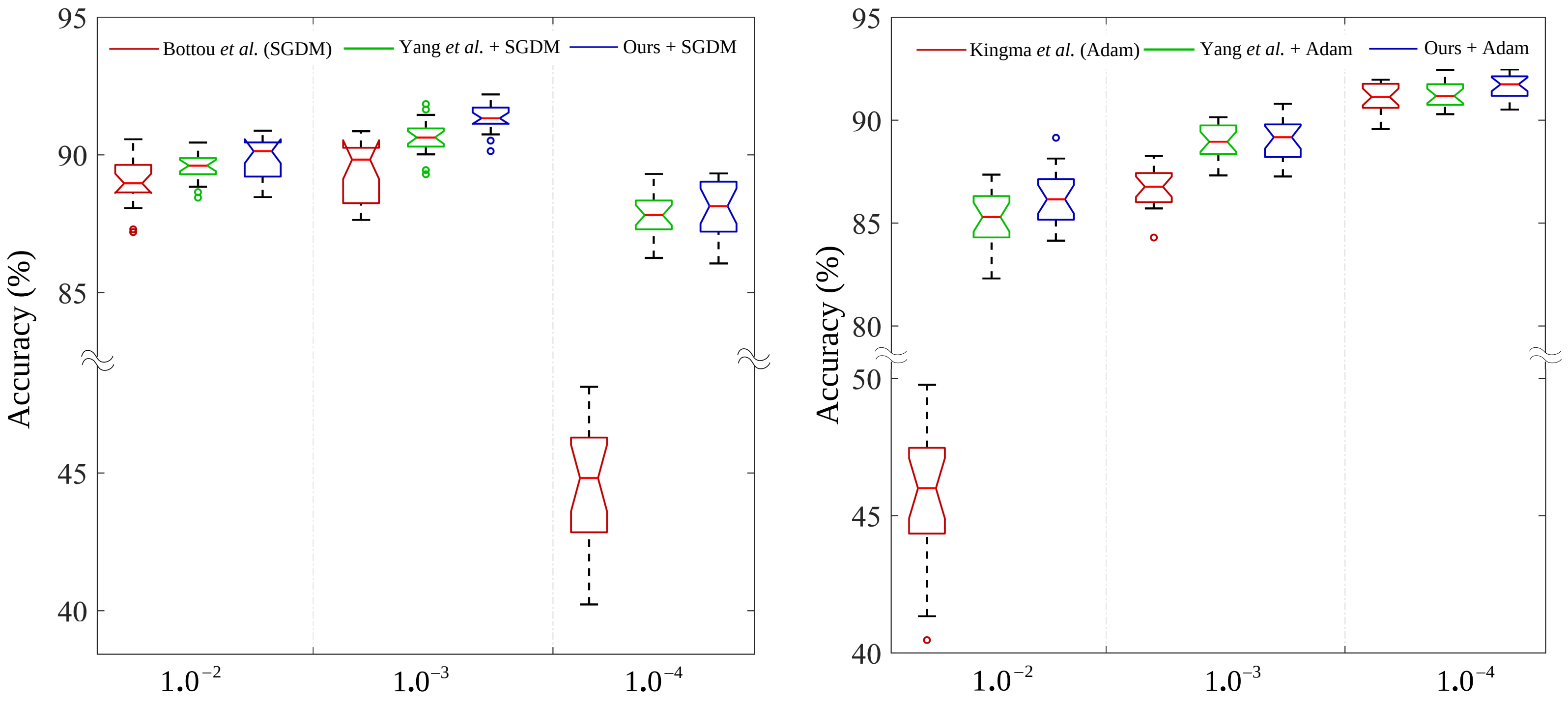
Table 9.6 – Comparison of the model convergence time in hours (h)

Methods	Cifar-100		Places-365-1		Places-365		Average
	ResNet-50 (h)	Inception-v3 (h)	ResNet-50 (h)	Inception-v3 (h)	ResNet-50 (h)	Inception-v3 (h)	
Bottou *et al.* (SGDM) [38]	2.4	7.2	28.4	81.4	93.1	241.8	77.6
Yang *et al.* [27] + SGDM	1.7	4.4	19.1	54.3	58.2	165.5	50.5
Ours + SGDM	**1.6**	**4.0**	**17.6**	**51.1**	**55.9**	**148.1**	**46.3**
Kingma *et al.* (Adam) [39]	4.1	9.4	33.2	90.3	103.1	268.6	84.8
Yang *et al.* [27] + Adam	2.9	6.4	21.8	58.6	75.3	170.4	56.0
Ours + Adam	**2.7**	**5.9**	**19.9**	**54.9**	**70.2**	**153.1**	**51.1**



(a) Caltech-101 (b) Places-365-1

Figure 9.5 – Top-1 testing accuracy of ResNet-50 and Inception-v3 wrt different initial learning rates.



(a) SGDM

(b) Adam

Figure 9.6 – Comparison of algorithms using ResNet-50 on Caltech-101 with two optimizers wrt different initial learning rates.

Chapter 10

Conclusion

10.1 Overview

The dissertation is an effort to show a detailed overview of the big data-based representation learning, then, several subnet-based representation learning algorithms using MP inverse strategy, such as Wi-HSNN, OS-HSNN, OC-HSNN, MCOC-HSNN and SS-HSNN, are proposed. Further, the RML-MP and SRML-MP are two advanced learning strategies for the multilayer MP inverse-based neural networks. At the end of this dissertation, a new optimization strategy for DCNN-based transfer learning is proposed. By doing so, it achieves a quicker network convergence speed.

In Chapter 1, the fundamental details of the representation learning are discussed. Also, the discussion of the motivations and contributions of this dissertation is elaborated. It is also important to highlight the proposed methodologies and the dissertation's expected outcomes. Chapter 2 builds a strong foundation by providing the background of the MP inverse-based models and some essential topics discussed in this dissertation. Then, Chapter 3 intends to provide a literature review on the existing state-of-the-art representation learning methods and advanced MP inverse-based networks, such as multilayer extreme learning machine and subnet neural network.

With the foundations and background ready, this dissertation elaborates the proposed subnet-based structures in Chapter 4, 5, 6, and 7. Following that, several novel network optimization algorithms by the combination of MP inverse strategy and advanced representation learning methods are proposed in Chapter 8 and 9. In the following section, the contributions of each chapter are summarized.

10.2 Contributions

The contributions of this dissertation can be split into three sections: the proposed subnet-based methods, i.e., Wi-HSNN, OS-HSNN, OC-HSNN, MCOC-HSNN and SS-HSNN; the deep learning methods, which are RML-MP, SRML-MP and MPFR; the newly gathered datasets, such as HFSWR-RD and CO-Mask. The key characteristics of each algorithm and dataset are listed in the following subsections.

10.2.1 The Proposed Subnet-based Methods

In 2016, a novel MP inverse-based architecture called SNN was developed for supervised learning [1]. It has the advantages of excellent generalization performance and a quicker learning curve. In this dissertation, we extend the idea of the subnet neural network to specific application domains. The variants of subnet neural networks include online learning, semi-supervised learning and one-class classification domains.

Chapter 4 presents a new supervised width-growth model (Wi-HSNN) based on subnet structure for feature refinement and classification. The contributions of this chapter are: 1) Most of the state-of-the-art multilayer networks architect their model in a layer-wise manner, however, the Wi-HSNN utilizes a width-growth strategy to iteratively search the optimal latent-space representations. Mathematical proof verifies the effectiveness of this model. 2) The Wi-HSNN is a one-step learning algorithm, learning the global-level encoding and classification simultaneously, which is capable of providing global-level representations. 3) A multi-model feature fusion strategy is proposed for better performance. The experimental results outperform all the competing MP inverse-based RL algorithms by 1% to 5% on the image classification domain. In addition, the proposed feature refinement strategy saves a large amount of training and inference time compared to the other subnet-based neural networks. Overall, compared to the other MP inverse-based structures, the proposed width-growth model is robust across various datasets in terms of generalization performance.

Chapter 5 derives a batch-by-batch learning strategy for the MP inverse technique. The traditional MP inverse method is a one-batch learning algorithm. However, in some real-world applications, the processing data is computationally infeasible to be trained at once. Thus, the recursive MP inverse is adopted and reformulated into a recursive learning scheme. By doing so, any sized large-scale datasets can be efficiently handled. Experimental results verify that the proposed strategy can process big data without the requirements of the high-performance computing device.

Chapter 6 proposed two subnet-based algorithms for one-class classification. The contributions of this chapter are: 1) A multilayer subnet-based one-class learning structure named OC-HSNN is proposed. 2) The MCOC-HSNN is an advanced one-class classification model using the maximum correntropy criterion for subspace latent optimization. 3) A new dataset named CO-Mask is gathered. In comparison to some advanced one-class learning algorithms, the proposed OC-SNN and MCOC-SNN have promising performance. Further, the visualized t-sne plot indicates that after several learning iterations, the inter-class distance of each category is reduced, while the intra-class counterpart is enhanced.

Chapter 7 extends the Wi-HSNN from the supervised learning domain to the semi-supervised learning domain. Specifically, the contributions of Chapter 7 are listed below. 1) A subnet-based semi-supervised learning method called SS-HSNN is proposed. Essentially, it is a manifold regularization algorithm that assumes the data with the same label are more likely to be close together. 2) A novel semi-supervised dataset for ship-target detection is formed. Comparisons with the other MP inverse-based semi-supervised methods show that the SS-HSNN achieves 0.5% to 2% improvement on image classification tasks. Also, the SS-HSNN achieves a real-time inference speed (1.2 ms per frame) with affordable training expenses. Therefore, the merits of the proposed MS-HSNN have been validated.

10.2.2 The Proposed Deep Learning Methods

Deep learning methods have become state-of-the-art algorithms both in the area of computer vision and big data analysis. Several advanced DCNN models, such as DenseNet and ResNet have achieved excellent performance on some 2D image datasets, including ImageNet and Place-365. By having the MP inverse strategy, the multilayer networks, such as multilayer ELM and DCNN, have shown superior performance in representation learning and some practical applications. Hence, this work utilizes the MP inverse-based retraining strategy to boost the performance of the existing multilayer network. It improves the performance through building novel optimization strategies in the following two aspects.

Chapter 8 harnesses the ability of multilayer ELM in handling large-scale datasets. It uses a multilayer framework to extract the useful features and to classify the input patterns. The core contribution of this chapter is that of the optimization of the hidden layer weights. Inspired by the subnet neural network that utilizes the pulled back error to search the weights of the new subspaces, the proposed models in Chapter 8 pull back the weights from the output layer to each hidden layer one by one. Then, the weights are updated according to the input and the feedback data. Further, the $\ell_{1/2}$-based loss function is used to search the sparse encoding. Experimental results show that the proposed optimization strategy boosts the performance by around 2% to 8% over the original multilayer ELM on image classification tasks.

Chapter 9 utilizes the batch-by-batch strategy to refine the FC layers of a DCNN. First, a random freeze learning strategy is adopted that uses one hyperparameter r to control the number of layers trained in each epoch to mitigate the overfitting problem. Second, the batch-by-batch strategy is engaged to refine the dense layer weights. The proposed method outmatches the compared optimization algorithms on some challenging transfer learning tasks concerning the evaluation metrics including the complexity and applicability. It achieves an average of 1.4% to 4.7% improvement compared to the SGD optimizer. In addition, compared to the original method, this strategy greatly enhances the convergence time (around 1.5 times quicker) required for retraining the DCNNs.

10.2.3 The Newly Gathered Datasets

In this dissertation, two new datasets called HFSWR-RD and CO-Mask are gathered.

In Chapter 6, we form a new misinformation detection dataset called CO-Mask. The information of this dataset is collected from the "big three" news agencies, which are Associated Press, Reuters, and Bloomberg. This dataset contains 4,907 training patterns and 3,725 testing patterns.

The HFSWR-RD is a semi-supervised learning dataset that is gathered from a high-frequency radar located on the east coast of China. Compared to the other sensors, the data processing of radar signals is more complex since various interference co-exists with the vessel signal. In addition, in practical ship-target monitoring, not all of the images have accurate ground truth. However, the current HFSWR dataset is dedicated to the supervised-learning task which only contains labelled samples. Therefore, we form a new radar dataset in Chapter 7 (HFSWR-RDE) with both labelled and unlabeled radar images for semi-supervised learning.

10.3 Applications

In this dissertation, the effectiveness of the proposed methods is verified in the following real-world applications.

- **Food Image Classification.** A healthy diet has crucial importance to human health, and food image classification plays a vital role in recording daily diets. Recent works on feature fusion have significantly boosted the generalization performance of food categorization tasks. The effectiveness of the proposed Wi-HSNN, OS-HSNN, OC-HSNN, MCOC-HSNN, RML-MP, SRML-MP and MPFR is validated on a challenging food dataset called Food-251.

- **Ship-target Detection.** High-frequency surface wave radar can be effectively used to detect ships in the exclusive economic zone. However, the ship signal is concealed and interfered with various clutter and background noise in the Doppler spectrum. Thus, the proposed OS-HSNN and SS-HSNN are utilized on HF-radar and HFSWR-RD datasets to efficiently detect ship-targets respectively.

- **Camera Model Identification.** This application domain aims to identify the camera that the image was taken with. It is quite a challenging task because the photos were taken from different cameras mostly differ in the texture details. A powerful and useful representation learning algorithm is what we urgently need for camera identification. Hence, we verified the proposed Wi-HSNN with state-of-the-art camera model identification models on the SPCUP dataset.

- **COVID-19-related Misinformation Detection.** From the general public perspective, social media is increasingly becoming a powerful tool for ordinary citizens to gather information and keep abreast of the developments of breaking news such as that related to COVID-19. However, misinformation exists that hinders the implementation of and compliance with public policies designed to flatten the curve and to go back to normal. We approached misinformation detection in digital platforms through one-class classification strategies, such as OC-HSNN and MCOC-HSNN.

10.4 Limitations

The limitations of the proposed methods are listed in this subsection as follows.

- For the Wi-HSNN, OS-HSNN, OC-HSNN, MCOC-HSNN, SS-HSNN, RML-MP, and SRML-MP, these methods have one key limitation: The structures are incapable of extracting high-level features because these models do not contain the convolutional layers. In other words, these algorithms could be feature refiners and classifiers, and they need advanced convolutional networks such as VGG, ResNet, and DenseNet for high-level feature extraction.

- For the proposed MPFR, it utilizes the MP inverse strategy to refine the dense layer weights. However, the training of the convolutional layers still needs the use of backpropagation-based optimization strategies, like SGDM and Adam. This hinders its applicability and popularity.

10.5 Scopes for Future Works

This section thoroughly presents the future works for boosting the proposed subnet neural networks and the deep learning models.

- The Google Brain research team proposed a novel deep learning architecture called MLP-Mixer [257]. This architecture is built based exclusively on multi-layer perceptrons. Hence, for future works, we can try to use the MP inverse strategy to boost the performance of the MLP-Mixer.

- The proposed subnet neural networks generate their global-level representations through the category information. Hence, it is worth investigating an SNN-based algorithm, which achieves a generalized feature space based on both feature properties and class-specific information.

- Self-supervised learning is a hot topic both in computer vision and machine learning. It will be promising if we extend the supervised subnet neural network into the self-supervised learning domain.

- The MPFR is proposed for DCNN retraining and transfer learning. It recalculates the dense layer weights in each retraining epoch. It would be worthwhile to use the proposed MP inverse strategy to train all the layers including convolutional layers in DCNNs.

Bibliography

[1] Y. Yang and Q. J. Wu, "Extreme learning machine with subnetwork hidden nodes for regression and classification," *IEEE transactions on cybernetics*, vol. 46, no. 12, pp. 2885–2898, 2016.

[2] T. Akilan, "Video foreground localization from traditional methods to deep learning," Ph.D. dissertation, University of Windsor (Canada), 2018.

[3] X. Song, S. Jiang, L. Herranz, Y. Kong, and K. Zheng, "Category co-occurrence modeling for large scale scene recognition," *Pattern Recognition*, vol. 59, pp. 98–111, 2016.

[4] R. Katuwal and P. N. Suganthan, "Stacked autoencoder based deep random vector functional link neural network for classification," *Applied Soft Computing*, vol. 85, p. 105854, 2019.

[5] A. A. MUSTAFA, "Selecting a representative image from a collection of images by solving a system of non-linear algebraic equations," *International Journal of Signal Processing*, vol. 1, pp. 177–185.

[6] J. Edwards, "Planet selfie: We're now posting a staggering 1.8 billion photos every day," *Business Insider*, 2014.

[7] L. Capital, "billion cameras by 2022 fuel business opportunities," *Tech reports*, 2017.

[8] L. Zhang, Q. Zhang, L. Zhang, D. Tao, X. Huang, and B. Du, "Ensemble manifold regularized sparse low-rank approximation for multiview feature embedding," *Pattern Recognition*, vol. 48, no. 10, pp. 3102–3112, 2015.

[9] C. Du, C. Du, L. Huang, and H. He, "Reconstructing perceived images from human brain activities with bayesian deep multiview learning," *IEEE transactions on neural networks and learning systems*, vol. 30, no. 8, pp. 2310–2323, 2018.

[10] T. Akilan, Q. J. Wu, and H. Zhang, "Effect of fusing features from multiple dcnn architectures in image classification," *IET Image Processing*, vol. 12, no. 7, pp. 1102–1110, 2018.

[11] F. De la Torre, "A least-squares framework for component analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 6, pp. 1041–1055, 2012.

[12] L. Zhao, T. Yang, J. Zhang, Z. Chen, Y. Yang, and Z. J. Wang, "Co-learning non-negative correlated and uncorrelated features for multi-view data," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[13] Z. Ding and Y. Fu, "Robust multiview data analysis through collective low-rank subspace," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 5, pp. 1986–1997, 2017.

[14] T. Akilan, Q. J. Wu, Y. Yang, and A. Safaei, "Fusion of transfer learning features and its application in image classification," in *2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE)*. IEEE, 2017, pp. 1–5.

[15] T. Akilan, Q. J. Wu, and W. Zhang, "Video foreground extraction using multi-view receptive field and encoder–decoder dcnn for traffic and surveillance applications," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 10, pp. 9478–9493, 2019.

[16] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *the Journal of machine Learning research*, vol. 3, pp. 993–1022, 2003.

[17] M. Welling, M. Rosen-Zvi, and G. E. Hinton, "Exponential family harmoniums with an application to information retrieval." in *Nips*, vol. 4. Citeseer, 2004, pp. 1481–1488.

[18] Y. Jia, M. Salzmann, T. Darrell *et al.*, "Factorized latent spaces with structured sparsity." in *NIPS*, vol. 10, 2010, pp. 982–990.

[19] P. L. Lai and C. Fyfe, “Kernel and nonlinear canonical correlation analysis,” *International Journal of Neural Systems*, vol. 10, no. 05, pp. 365–377, 2000.

[20] S. Akaho, “A kernel method for canonical correlation analysis,” *arXiv preprint cs/0609071*, 2006.

[21] J. Zheng, F. Cai, W. Chen, C. Feng, and H. Chen, “Hierarchical neural representation for document classification,” *Cognitive Computation*, vol. 11, no. 2, pp. 317–327, 2019.

[22] Z. Jiang, Z. Lin, and L. S. Davis, “Label consistent k-svd: Learning a discriminative dictionary for recognition,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 11, pp. 2651–2664, 2013.

[23] Y. Yang and Q. J. Wu, “Features combined from hundreds of midlayers: Hierarchical networks with subnetwork nodes,” *IEEE transactions on neural networks and learning systems*, vol. 30, no. 11, pp. 3313–3325, 2019.

[24] J. Tang, C. Deng, and G.-B. Huang, “Extreme learning machine for multilayer perceptron,” *IEEE transactions on neural networks and learning systems*, vol. 27, no. 4, pp. 809–821, 2016.

[25] H. Dai, J. Cao, T. Wang, M. Deng, and Z. Yang, “Multilayer one-class extreme learning machine,” *Neural Networks*, vol. 115, pp. 11–22, 2019.

[26] W. F. Schmidt, M. A. Kraaijveld, and R. P. 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.* IEEE, 1992, pp. 1–4.

[27] Y. Yang, J. Q. Wu, X. Feng, and A. Thangarajah, “Recomputation of dense layers for the performance improvement of dcnn,” *IEEE transactions on pattern analysis and machine intelligence*, 2019.

[28] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, “Extreme learning machine: theory and applications,” *Neurocomputing*, vol. 70, no. 1-3, pp. 489–501, 2006.

[29] D. Wang and M. Li, "Stochastic configuration networks: Fundamentals and algorithms," *IEEE transactions on cybernetics*, vol. 47, no. 10, pp. 3466–3479, 2017.

[30] C. P. Chen and Z. Liu, "Broad learning system: An effective and efficient incremental learning system without the need for deep architecture," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 1, pp. 10–24, 2017.

[31] Y. Yang, Q. J. Wu, W.-L. Zheng, and B.-L. Lu, "Eeg-based emotion recognition using hierarchical network with subnetwork nodes," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 10, no. 2, pp. 408–419, 2017.

[32] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[33] Y. LeCun, F. J. Huang, and L. Bottou, "Learning methods for generic object recognition with invariance to pose and lighting," in *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, vol. 2. IEEE, 2004, pp. II–104.

[34] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

[35] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, "Places: A 10 million image database for scene recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 6, pp. 1452–1464, 2017.

[36] A. Rasmus, M. Berglund, M. Honkala, H. Valpola, and T. Raiko, "Semi-supervised learning with ladder networks," in *Advances in Neural Information Processing Systems*, 2015, pp. 3546–3554.

[37] I. Golan and R. El-Yaniv, “Deep anomaly detection using geometric transformations,” in *Advances in Neural Information Processing Systems*, 2018, pp. 9758–9769.

[38] L. Bottou, “Large-scale machine learning with stochastic gradient descent,” in *Proceedings of COMPSTAT’2010*. Springer, 2010, pp. 177–186.

[39] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

[40] Y. Yang and Q. J. Wu, “Extreme learning machine with subnetwork hidden nodes for regression and classification,” *IEEE transactions on cybernetics*, vol. 46, no. 12, pp. 2885–2898, 2016.

[41] D. Husmeier, “Random vector functional link (rvfl) networks,” in *Neural Networks for Conditional Probability Estimation*. Springer, 1999, pp. 87–97.

[42] L. L. C. Kasun, H. Zhou, G.-B. Huang, and C. M. Vong, “Representational learning with extreme learning machine for big data,” *IEEE Intelligent Systems*, vol. 28, no. 6, pp. 31–34, 2013.

[43] W. Liu, P. P. Pokharel, and J. C. Príncipe, “Correntropy: Properties and applications in non-gaussian signal processing,” *IEEE Transactions on Signal Processing*, vol. 55, no. 11, pp. 5286–5298, 2007.

[44] B. Chen, X. Wang, Y. Li, and J. C. Principe, “Maximum correntropy criterion with variable center,” *IEEE Signal Processing Letters*, vol. 26, no. 8, pp. 1212–1216, 2019.

[45] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, “Backpropagation applied to handwritten zip code recognition,” *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[46] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.

[47] S. Ruder, M. E. Peters, S. Swayamdipta, and T. Wolf, "Transfer learning in natural language processing," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorials*, 2019, pp. 15–18.

[48] T. Akilan, A. Thiagarajan, B. Venkatesan, S. Thirumeni, and S. G. Chandrasekaran, "Quantifying the impact of complementary visual and textual cues under image captioning," in *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2020, pp. 389–394.

[49] P. Kaur, B. S. Khehra, and A. P. S. Pharwaha, "Deep transfer learning based multiway feature pyramid network for object detection in images," *Mathematical Problems in Engineering*, 2021.

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

[51] A. Severyn and A. Moschitti, "Twitter sentiment analysis with deep convolutional neural networks," in *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, 2015, pp. 959–962.

[52] G. Lin, J. Zhang, W. Luo, L. Pan, Y. Xiang, O. De Vel, and P. Montague, "Cross-project transfer representation learning for vulnerable function discovery," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 7, pp. 3289–3297, 2018.

[53] S. Roweis, "Em algorithms for pca and spca," *Advances in neural information processing systems*, pp. 626–632, 1998.

[54] R. M. Neal, "Connectionist learning of belief networks," *Artificial intelligence*, vol. 56, no. 1, pp. 71–113, 1992.

[55] I. J. Goodfellow, A. Courville, and Y. Bengio, "Spike-and-slab sparse coding for unsupervised feature discovery," *arXiv preprint arXiv:1201.3382*, 2012.

[56] I. T. Jolliffe and J. Cadima, "Principal component analysis: a review and recent developments," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 374, no. 2065, p. 20150202, 2016.

[57] K. 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*, vol. 2, no. 11, pp. 559–572, 1901.

[58] Y. Li, N. Wang, and R. J. Carroll, "Selecting the number of principal components in functional data," *Journal of the American Statistical Association*, vol. 108, no. 504, pp. 1284–1294, 2013.

[59] Q. Zhao, D. Meng, Z. Xu, W. Zuo, and L. Zhang, "Robust principal component analysis with complex noise," in *International conference on machine learning.* PMLR, 2014, pp. 55–63.

[60] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[61] P. Smolensky, "Chapter 6: information processing in dynamical systems: foundations of harmony theory," *Parallel distributed processing: explorations in the microstructure of cognition*, vol. 1.

[62] M. Ranzato and G. E. Hinton, "Modeling pixel means and covariances using factorized third-order boltzmann machines," in *2010 IEEE computer society conference on computer vision and pattern recognition.* IEEE, 2010, pp. 2551–2558.

[63] A.-r. Mohamed, G. E. Dahl, and G. Hinton, "Acoustic modeling using deep belief networks," *IEEE transactions on audio, speech, and language processing*, vol. 20, no. 1, pp. 14–22, 2011.

[64] A. MEİMANDİ, A. OryAn, S. HAddAdI, and A. B. SAdegH, "Histopathological and biomechanical evaluation of bone healing properties of dbm and dbm-g90

in a rabbit model," *Acta orthopaedica et traumatologica turcica*, vol. 49, no. 6, p. 683, 2014.

[65] G.-S. Xie, X.-B. Jin, X.-Y. Zhang, S.-F. Zang, C. Yang, Z. Wang, and J. Pu, "From class-specific to class-mixture: Cascaded feature representations via restricted boltzmann machine learning," *IEEE Access*, vol. 6, pp. 69 393–69 406, 2018.

[66] X. Lü, L. Meng, C. Chen, and P. Wang, "Fuzzy removing redundancy restricted boltzmann machine: improving learning speed and classification accuracy," *IEEE Transactions on Fuzzy Systems*, vol. 28, no. 10, pp. 2495–2509, 2019.

[67] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[68] G. Hinton and S. T. Roweis, "Stochastic neighbor embedding," in *NIPS*, vol. 15. Citeseer, 2002, pp. 833–840.

[69] L. Van Der Maaten, "Learning a parametric embedding by preserving local structure," in *Artificial Intelligence and Statistics*. PMLR, 2009, pp. 384–391.

[70] J. Weston, F. Ratle, H. Mobahi, and R. Collobert, "Deep learning via semi-supervised embedding," in *Neural networks: Tricks of the trade*. Springer, 2012, pp. 639–655.

[71] Y. Bengio, H. Larochelle, and P. Vincent, "Non-local manifold parzen windows," in *NIPS*, vol. 18, 2005, pp. 115–122.

[72] L. Seidenari, G. Serra, A. D. Bagdanov, and A. Del Bimbo, "Local pyramidal descriptors for image recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 36, no. 5, pp. 1033–1040, 2014.

[73] J. Sánchez, F. Perronnin, T. Mensink, and J. Verbeek, "Image classification with the fisher vector: Theory and practice," *International journal of computer vision*, vol. 105, no. 3, pp. 222–245, 2013.

[74] R. Kwitt, N. Vasconcelos, and N. Rasiwasia, “Scene recognition on the semantic manifold,” in *European Conference on Computer Vision.* Springer, 2012, pp. 359–372.

[75] L. Zhang, X. Zhen, and L. Shao, “Learning object-to-class kernels for scene classification,” *IEEE Transactions on image processing*, vol. 23, no. 8, pp. 3241–3253, 2014.

[76] X. He, D. Cai, S. Yan, and H.-J. Zhang, “Neighborhood preserving embedding,” in *Tenth IEEE International Conference on Computer Vision (ICCV’05) Volume 1*, vol. 2. IEEE, 2005, pp. 1208–1213.

[77] D. Cai, X. He, K. Zhou, J. Han, and H. Bao, “Locality sensitive discriminant analysis.” in *IJCAI*, vol. 2007, 2007, pp. 1713–1726.

[78] L. Bo, X. Ren, and D. Fox, “Multipath sparse coding using hierarchical matching pursuit,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 660–667.

[79] S. Lazebnik, C. Schmid, and J. Ponce, “Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories,” in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’06)*, vol. 2. IEEE, 2006, pp. 2169–2178.

[80] L. Liu, L. Wang, and X. Liu, “In defense of soft-assignment coding,” in *2011 International Conference on Computer Vision.* IEEE, 2011, pp. 2486–2493.

[81] J. C. Van Gemert, C. J. Veenman, A. W. Smeulders, and J.-M. Geusebroek, “Visual word ambiguity,” *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 7, pp. 1271–1283, 2009.

[82] J. Yang, K. Yu, Y. Gong, and T. Huang, “Linear spatial pyramid matching using sparse coding for image classification,” 2009.

[83] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong, “Locality-constrained linear coding for image classification,” in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on.* IEEE, 2010, pp. 3360–3367.

[84] N. Rasiwasia and N. Vasconcelos, "Holistic context models for visual recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 5, pp. 902–917, 2012.

[85] J. Weng, N. Ahuja, and T. S. Huang, "Cresceptron: a self-organizing neural network which grows adaptively," in *[Proceedings 1992] IJCNN International Joint Conference on Neural Networks*, vol. 1. IEEE, 1992, pp. 576–581.

[86] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504–507, 2006.

[87] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[88] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[89] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "Cnn features off-the-shelf: an astounding baseline for recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2014, pp. 806–813.

[90] S. Guo, W. Huang, L. Wang, and Y. Qiao, "Locally supervised deep hybrid model for scene recognition," *IEEE transactions on image processing*, vol. 26, no. 2, pp. 808–820, 2016.

[91] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[92] H. Bourlard and Y. Kamp, "Auto-association by multilayer perceptrons and singular value decomposition," *Biological cybernetics*, vol. 59, no. 4, pp. 291–294, 1988.

[93] M. Ranzato, C. Poultney, S. Chopra, Y. LeCun *et al.*, "Efficient learning of sparse representations with an energy-based model," *Advances in neural information processing systems*, vol. 19, p. 1137, 2007.

[94] H. Lee, C. Ekanadham, and A. Ng, "Sparse deep belief net model for visual area v2," *Advances in neural information processing systems*, vol. 20, pp. 873–880, 2007.

[95] I. Goodfellow, H. Lee, Q. Le, A. Saxe, and A. Ng, "Measuring invariances in deep networks," *Advances in neural information processing systems*, vol. 22, pp. 646–654, 2009.

[96] M. Ranzato, Y.-L. Boureau, Y. LeCun *et al.*, "Sparse feature learning for deep belief networks," *Advances in neural information processing systems*, vol. 20, pp. 1185–1192, 2007.

[97] W. Y. Zou, A. Y. Ng, and K. Yu, "Unsupervised learning of visual invariance with temporal coherence," in *NIPS 2011 workshop on deep learning and unsupervised feature learning*, vol. 3, 2011.

[98] J. Tang, C. Deng, and G.-B. Huang, "Extreme learning machine for multilayer perceptron," *IEEE transactions on neural networks and learning systems*, vol. 27, no. 4, pp. 809–821, 2015.

[99] B. A. Olshausen and D. J. Field, "Emergence of simple-cell receptive field properties by learning a sparse code for natural images," *Nature*, vol. 381, no. 6583, pp. 607–609, 1996.

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

[101] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *Journal of machine learning research*, vol. 11, no. Dec, pp. 3371–3408, 2010.

[102] P. Vincent, “A connection between score matching and denoising autoencoders,” *Neural computation*, vol. 23, no. 7, pp. 1661–1674, 2011.

[103] K. Swersky, M. Ranzato, D. Buchman, B. Marlin, and N. de Freitas, “On score matching for energy based models: Generalizing autoencoders and simplifying deep learning,” in *Proc. ICML*, 2011.

[104] G. Alain and Y. Bengio, “What regularized auto-encoders learn from the data-generating distribution,” *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 3563–3593, 2014.

[105] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio, “Contractive auto-encoders: Explicit invariance during feature extraction,” in *Icml*, 2011.

[106] J. Park and I. W. Sandberg, “Universal approximation using radial-basis-function networks,” *Neural computation*, vol. 3, no. 2, pp. 246–257, 1991.

[107] B. Igelnik and Y.-H. Pao, “Stochastic choice of basis functions in adaptive function approximation and the functional-link net,” *IEEE Transactions on Neural Networks*, vol. 6, no. 6, pp. 1320–1329, 1995.

[108] D. P. Mesquita, J. P. P. Gomes, and L. R. Rodrigues, “Artificial neural networks with random weights for incomplete datasets,” *Neural Processing Letters*, vol. 50, no. 3, pp. 2345–2372, 2019.

[109] C. Huang, Q. Huang, and D. Wang, “Stochastic configuration networks based adaptive storage replica management for power big data processing,” *IEEE Transactions on Industrial Informatics*, vol. 16, no. 1, pp. 373–383, 2019.

[110] Y. Ren, P. N. Suganthan, N. Srikanth, and G. Amaratunga, “Random vector functional link network for short-term electricity load demand forecasting,” *Information Sciences*, vol. 367, pp. 1078–1093, 2016.

[111] Y. Zhang, J. Wu, Z. Cai, B. Du, and S. Y. Philip, “An unsupervised parameter learning model for rvfl neural network,” *Neural Networks*, vol. 112, pp. 85–97, 2019.

[112] T. Wang, J. Cao, X. Lai, and B. Chen, "Deep weighted extreme learning machine," *Cognitive Computation*, vol. 10, no. 6, pp. 890–907, 2018.

[113] C. M. Wong, C. M. Vong, P. K. Wong, and J. Cao, "Kernel-based multilayer extreme learning machines for representation learning," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 3, pp. 757–762, 2016.

[114] Y. Yang and Q. J. Wu, "Multilayer extreme learning machine with subnetwork nodes for representation learning," *IEEE transactions on cybernetics*, vol. 46, no. 11, pp. 2570–2583, 2016.

[115] W. Zhang, J. Wu, and Y. Yang, "Wi-hsnn: A subnetwork-based encoding structure for dimension reduction and food classification via harnessing multi-cnn model high-level features," *Neurocomputing*, vol. 414, pp. 57–66, 2020.

[116] W. Wu, Q. J. Wu, W. Sun, Y. Yang, X. Yuan, W.-L. Zheng, and B.-L. Lu, "A regression method with subnetwork neurons for vigilance estimation using eog and eeg," *IEEE Transactions on Cognitive and Developmental Systems*, 2018.

[117] Y. Yang, Q. J. Wu, and Y. Wang, "Autoencoder with invertible functions for dimension reduction and image reconstruction," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 48, no. 7, pp. 1065–1079, 2016.

[118] B. Xiao and S. Yin, "A new disturbance attenuation control scheme for quadrotor unmanned aerial vehicles," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 6, pp. 2922–2932, 2017.

[119] H.-J. Ma, Y. Liu, T. Li, and G.-H. Yang, "Nonlinear high-gain observer-based diagnosis and compensation for actuator and sensor faults in a quadrotor unmanned aerial vehicle," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 1, pp. 550–562, 2018.

[120] M. S. Hossain, M. Al-Hammadi, and G. Muhammad, "Automatic fruit classification using deep learning for industrial applications," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 2, pp. 1027–1034, 2018.

[121] C. P. Chen, "A rapid supervised learning neural network for function interpolation and approximation," *IEEE Transactions on Neural Networks*, vol. 7, no. 5, pp. 1220–1230, 1996.

[122] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: a new learning scheme of feedforward neural networks," in *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, vol. 2. IEEE, 2004, pp. 985–990.

[123] W. F. Schmidt, M. A. Kraaijveld, R. P. Duin *et al.*, "Feed forward neural networks with random weights," in *International Conference on Pattern Recognition.* IEEE Computer Society Press, 1992, pp. 1–1.

[124] I. J. Goodfellow, Y. Bulatov, J. Ibarz, S. Arnoud, and V. Shet, "Multi-digit number recognition from street view imagery using deep convolutional neural networks," *arXiv preprint arXiv:1312.6082*, 2013.

[125] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[126] A. El-Yamany, H. Fouad, and Y. Raffat, "A generic approach cnn-based camera identification for manipulated images," in *2018 IEEE International Conference on Electro/Information Technology (EIT).* IEEE, 2018, pp. 0165–0169.

[127] B. Xu, X. Wang, X. Zhou, J. Xi, and S. Wang, "Source camera identification from image texture features," *Neurocomputing*, vol. 207, pp. 131–140, 2016.

[128] C. Chen and M. C. Stamm, "Camera model identification framework using an ensemble of demosaicing features," in *2015 IEEE International Workshop on Information Forensics and Security (WIFS).* IEEE, 2015, pp. 1–6.

[129] Y. Matsuda, H. Hoashi, and K. Yanai, "Recognition of multiple-food images by detecting candidate regions," in *2012 IEEE International Conference on Multimedia and Expo.* IEEE, 2012, pp. 25–30.

[130] L. Bossard, M. Guillaumin, and L. Van Gool, "Food-101–mining discriminative components with random forests," in *European Conference on Computer Vision*. Springer, 2014, pp. 446–461.

[131] P. Kaur, K. Sikka, W. Wang, S. Belongie, and A. Divakaran, "Foodx-251: A dataset for fine-grained food classification," *arXiv preprint arXiv:1907.06167*, 2019.

[132] Y. Kawano and K. Yanai, "Automatic expansion of a food image dataset leveraging existing categories with domain adaptation," in *European Conference on Computer Vision*. Springer, 2014, pp. 3–17.

[133] T. K. Ho, "Random decision forests," in *Proceedings of 3rd international conference on document analysis and recognition*, vol. 1. IEEE, 1995, pp. 278–282.

[134] C. M. Bishop, *Pattern recognition and machine learning*. springer-Verlag, 2006.

[135] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Weinberger, "Deep networks with stochastic depth," in *European Conference on Computer Vision*. Springer, 2016, pp. 646–661.

[136] S. Singh, A. Gupta, and A. A. Efros, "Unsupervised discovery of mid-level discriminative patches," in *European Conference on Computer Vision*. Springer, 2012, pp. 73–86.

[137] K. Yanai and Y. Kawano, "Food image recognition using deep convolutional network with pre-training and fine-tuning," in *2015 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*. IEEE, 2015, pp. 1–6.

[138] P. Pandey, A. Deepthi, B. Mandal, and N. B. Puhan, "Foodnet: Recognizing foods using ensemble of deep networks," *IEEE Signal Processing Letters*, vol. 24, no. 12, pp. 1758–1762, 2017.

[139] J. Zheng, L. Zou, and Z. J. Wang, "Mid-level deep food part mining for food image recognition," *IET Computer Vision*, vol. 12, no. 3, pp. 298–304, 2017.

[140] E. Aguilar, M. Bolaños, and P. Radeva, "Food recognition using fusion of classifiers based on cnns," in *International Conference on Image Analysis and Processing.* Springer, 2017, pp. 213–224.

[141] H. Hassannejad, G. Matrella, P. Ciampolini, I. De Munari, M. Mordonini, and S. Cagnoni, "Food image recognition using very deep convolutional networks," in *Proceedings of the 2nd International Workshop on Multimedia Assisted Dietary Management.* ACM, 2016, pp. 41–49.

[142] E. K. Chong and S. H. Zak, *An introduction to optimization.* John Wiley & Sons, 2004.

[143] G. H. Golub and C. F. Van Loan, *Matrix computations.* JHU press, 2012, vol. 3.

[144] L. Fei-Fei and P. Perona, "A bayesian hierarchical model for learning natural scene categories," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 2. IEEE, 2005, pp. 524–531.

[145] L. Fei-Fei, R. Fergus, and P. Perona, "Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories," *Computer vision and Image understanding*, vol. 106, no. 1, pp. 59–70, 2007.

[146] G. Griffin, A. Holub, and P. Perona, "Caltech-256 object category dataset," 2007.

[147] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, "Places: A 10 million image database for scene recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 6, pp. 1452–1464, 2018.

[148] W. Zhang, Q. Li, Q. J. Wu, Y. Yang, and M. Li, "A novel ship target detection algorithm based on error self-adjustment extreme learning machine and cascade classifier," *Cognitive Computation*, vol. 11, no. 1, pp. 110–124, 2019.

[149] D. Charte, F. Charte, S. García, M. J. del Jesus, and F. Herrera, "A practical tutorial on autoencoders for nonlinear feature fusion: Taxonomy, models, software and guidelines," *Information Fusion*, vol. 44, pp. 78–96, 2018.

[150] A. Ng *et al.*, "Sparse autoencoder," *CS294A Lecture notes*, vol. 72, no. 2011, pp. 1–19, 2011.

[151] F. Jangal, S. Saillant, and M. Helier, "Wavelets: a versatile tool for the high frequency surface wave radar," in *2007 IEEE Radar Conference*. IEEE, 2007, pp. 497–502.

[152] F. Jangal, S. Saillant, and M. Hélier, "Wavelet contribution to remote sensing of the sea and target detection for a high-frequency surface wave radar," *IEEE Geoscience and Remote Sensing Letters*, vol. 5, no. 3, pp. 552–556, 2008.

[153] Q. Li, W. Zhang, M. Li, J. Niu, and Q. J. Wu, "Automatic detection of ship targets based on wavelet transform for hf surface wavelet radar," *IEEE Geoscience and Remote Sensing Letters*, vol. 14, no. 5, pp. 714–718, 2017.

[154] J. O. Hinz, M. Holters, U. Zölzer, A. Gupta, and T. Fickenscher, "Presegmentation-based adaptive cfar detection for hfswr," in *2012 IEEE Radar Conference*. IEEE, 2012, pp. 0665–0670.

[155] T. Liu, G. A. Lampropoulos, and C. Fei, "Cfar ship detection system using polarimetric data," in *2008 IEEE Radar Conference*. IEEE, 2008, pp. 1–4.

[156] L. Zhang, W. You, Q. Wu, S. Qi, and Y. Ji, "Deep learning-based automatic clutter/interference detection for hfswr," *Remote Sensing*, vol. 10, no. 10, p. 1517, 2018.

[157] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7263–7271.

[158] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*. Springer, 2016, pp. 21–37.

[159] F. Gao, T. Huang, J. Sun, J. Wang, A. Hussain, and E. Yang, “A new algorithm for sar image target recognition based on an improved deep convolutional neural network,” *Cognitive Computation*, vol. 11, no. 6, pp. 809–824, 2019.

[160] Y. Xiao, H. Wang, and W. Xu, “Parameter selection of gaussian kernel for one-class svm,” *IEEE Transactions on Cybernetics*, vol. 45, no. 5, pp. 941–953, 2014.

[161] T. Wang, J. Cao, X. Lai, and Q. J. Wu, “Hierarchical one-class classifier with within-class scatter-based autoencoders,” *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[162] M. Nicolau, J. McDermott *et al.*, “Learning neural representations for network anomaly detection,” *IEEE Transactions on Cybernetics*, vol. 49, no. 8, pp. 3074–3087, 2018.

[163] F. Alaei, N. Girard, S. Barrat, and J.-Y. Ramel, “A new one-class classification method based on symbolic representation: Application to document classification,” in *2014 11th IAPR International Workshop on Document Analysis Systems*. IEEE, 2014, pp. 272–276.

[164] A. E. Fard, M. Mohammadi, Y. Chen, and B. Van de Walle, “Computational rumor detection without non-rumor: A one-class classification approach,” *IEEE Transactions on Computational Social Systems*, vol. 6, no. 5, pp. 830–846, 2019.

[165] E. Parzen, “On estimation of a probability density function and mode,” *The annals of mathematical statistics*, vol. 33, no. 3, pp. 1065–1076, 1962.

[166] E. Pekalska, D. M. Tax, and R. Duin, “One-class lp classifiers for dissimilarity representations,” in *Advances in Neural Information Processing Systems*, 2003, pp. 777–784.

[167] L. E. Ghaoui, M. I. Jordan, and G. R. Lanckriet, “Robust novelty detection with single-class mpm,” in *Advances in Neural Information Processing Systems*, 2003, pp. 929–936.

[168] P. Juszczak, D. M. Tax, E. Pe, R. P. Duin *et al.*, "Minimum spanning tree based one-class classifier," *Neurocomputing*, vol. 72, no. 7-9, pp. 1859–1869, 2009.

[169] Q. Leng, H. Qi, J. Miao, W. Zhu, and G. Su, "One-class classification with extreme learning machine," *Mathematical Problems in Engineering*, vol. 2015, 2015.

[170] J. Cao, H. Dai, B. Lei, C. Yin, H. Zeng, and A. Kummert, "Maximum correntropy criterion-based hierarchical one-class classification," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[171] X. Cui, J. Cao, T. Wang, and X. Lai, "Robust randomized autoencoder and correntropy criterion based one-class classification," *IEEE Transactions on Circuits and Systems II: Express Briefs*, 2020.

[172] T. Wang, X. Lai, J. Cao, C.-M. Vong, and B. Chen, "An enhanced hierarchical extreme learning machine with random sparse matrix based autoencoder," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* IEEE, 2019, pp. 3817–3821.

[173] W. Zhang, Q. J. Wu, Y. Yang, T. Akilan, and H. Zhang, "A width-growth model with subnetwork nodes and refinement structure for representation learning and image classification," *IEEE Transactions on Industrial Informatics*, 2020.

[174] W. Zhang, Q. J. Wu, Y. Yang, T. Akilan, and M. Li, "Hkpm: A hierarchical key-area perception model for hfswr maritime surveillance," *IEEE Transactions on Geoscience and Remote Sensing*, 2021.

[175] T. Akilan, Q. J. Wu, A. Safaei, and W. Jiang, "A late fusion approach for harnessing multi-cnn model high-level features," in *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC).* IEEE, 2017, pp. 566–571.

[176] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks." in *CVPR*, vol. 1, no. 2, 2017, p. 3.

[177] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Advances in Neural Information Processing Systems*, 2014, pp. 3320–3328.

[178] S. Li, M. Shao, and Y. Fu, "Locality linear fitting one-class svm with low-rank constraints for outlier detection," in *2014 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2014, pp. 676–683.

[179] S. Li, M. Shao, and Fu, "Multi-view low-rank analysis with applications to outlier detection," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 12, no. 3, pp. 1–22, 2018.

[180] J. Wang and A. Cherian, "Gods: Generalized one-class discriminative subspaces for anomaly detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 8201–8211.

[181] G.-B. Huang, L. Chen, C. K. Siew *et al.*, "Universal approximation using incremental constructive feedforward networks with random hidden nodes," *IEEE Trans. Neural Networks*, vol. 17, no. 4, pp. 879–892, 2006.

[182] A. J. Bell and T. J. Sejnowski, "The "independent components" of natural scenes are edge filters," *Vision Research*, vol. 37, no. 23, pp. 3327–3338, 1997.

[183] S. Gidaris, P. Singh, and N. Komodakis, "Unsupervised representation learning by predicting image rotations," *arXiv preprint arXiv:1803.07728*, 2018.

[184] W. Zhang, W. W. Zhao, D. Wu, and Y. Yang, "Predicting covid-19 trends in canada: a tale of four models," *Cognitive Computation and Systems*, vol. 2, no. 3, pp. 112–118, 2020.

[185] A. Zubiaga, A. Aker, K. Bontcheva, M. Liakata, and R. Procter, "Detection and resolution of rumours in social media: A survey," *ACM Computing Surveys (CSUR)*, vol. 51, no. 2, pp. 1–36, 2018.

[186] P. Faustini and T. F. Covões, "Fake news detection using one-class classification," in *2019 8th Brazilian Conference on Intelligent Systems (BRACIS)*. IEEE, 2019, pp. 592–597.

[187] L. Metz, B. Poole, D. Pfau, and J. Sohl-Dickstein, "Unrolled generative adversarial networks," *arXiv preprint arXiv:1611.02163*, 2016.

[188] L. Ruff, R. Vandermeulen, N. Goernitz, L. Deecke, S. A. Siddiqui, A. Binder, E. Müller, and M. Kloft, "Deep one-class classification," in *International conference on machine learning.* PMLR, 2018, pp. 4393–4402.

[189] K. Sohn, C.-L. Li, J. Yoon, M. Jin, and T. Pfister, "Learning and evaluating representations for deep one-class classification," *arXiv preprint arXiv:2011.02578*, 2020.

[190] J. Zhang, Y. Han, J. Tang, Q. Hu, and J. Jiang, "Semi-supervised image-to-video adaptation for video action recognition," *IEEE transactions on cybernetics*, vol. 47, no. 4, pp. 960–973, 2016.

[191] S. Wang, Z. Ma, Y. Yang, X. Li, C. Pang, and A. G. Hauptmann, "Semi-supervised multiple feature analysis for action recognition," *IEEE transactions on multimedia*, vol. 16, no. 2, pp. 289–298, 2013.

[192] M. R. Keyvanpour and M. B. Imani, "Semi-supervised text categorization: Exploiting unlabeled data using ensemble learning algorithms," *Intelligent Data Analysis*, vol. 17, no. 3, pp. 367–385, 2013.

[193] D. Tuia and G. Camps-Valls, "Semisupervised remote sensing image classification with cluster kernels," *IEEE Geoscience and Remote Sensing Letters*, vol. 6, no. 2, pp. 224–228, 2009.

[194] J. Muñoz-Marí, F. Bovolo, L. Gómez-Chova, L. Bruzzone, and G. Camp-Valls, "Semisupervised one-class support vector machines for classification of remote sensing data," *IEEE transactions on geoscience and remote sensing*, vol. 48, no. 8, pp. 3188–3197, 2010.

[195] Z. Yu, Y. Zhang, J. You, C. P. Chen, H.-S. Wong, G. Han, and J. Zhang, "Adaptive semi-supervised classifier ensemble for high dimensional data classification," *IEEE transactions on cybernetics*, vol. 49, no. 2, pp. 366–379, 2019.

[196] A. Fujino, N. Ueda, and K. Saito, “A hybrid generative/discriminative approach to semi-supervised classifier design,” in *AAAI*, 2005, pp. 764–769.

[197] X. Zhu and J. Lafferty, “Harmonic mixtures: combining mixture models and graph-based methods for inductive and scalable semi-supervised learning,” in *Proceedings of the 22nd international conference on Machine learning*, 2005, pp. 1052–1059.

[198] Z. Yuan and C. Lin, “Research on strong constraint self-training algorithm and applied to remote sensing image classification,” in *2021 IEEE International Conference on Power Electronics, Computer Applications (ICPECA).* IEEE, 2021, pp. 981–985.

[199] D. Wu, M. Shang, G. Wang, and L. Li, “A self-training semi-supervised classification algorithm based on density peaks of data and differential evolution,” in *2018 IEEE 15th international conference on networking, Sensing and Control (ICNSC).* IEEE, 2018, pp. 1–6.

[200] A. Blum and T. Mitchell, “Combining labeled and unlabeled data with co-training,” in *Proceedings of the eleventh annual conference on Computational learning theory*, 1998, pp. 92–100.

[201] Z.-H. Zhou and M. Li, “Semisupervised regression with cotraining-style algorithms,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 11, pp. 1479–1493, 2007.

[202] M. Belkin, P. Niyogi, and V. Sindhwani, “Manifold regularization: A geometric framework for learning from labeled and unlabeled examples,” *Journal of machine learning research*, vol. 7, no. Nov, pp. 2399–2434, 2006.

[203] G. Huang, S. Song, J. N. Gupta, and C. Wu, “Semi-supervised and unsupervised extreme learning machines,” *IEEE transactions on cybernetics*, vol. 44, no. 12, pp. 2405–2417, 2014.

[204] B. Lecouat, C.-S. Foo, H. Zenati, and V. Chandrasekhar, "Manifold regularization with gans for semi-supervised learning," *arXiv preprint arXiv:1807.04307*, 2018.

[205] Q. She, B. Hu, Z. Luo, T. Nguyen, and Y. Zhang, "A hierarchical semi-supervised extreme learning machine method for eeg recognition," *Medical & biological engineering & computing*, vol. 57, no. 1, pp. 147–157, 2019.

[206] H. Zhao, J. Zheng, W. Deng, and Y. Song, "Semi-supervised broad learning system based on manifold regularization and broad network," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67, no. 3, pp. 983–994, 2020.

[207] G. Krishnasamy and R. Paramesran, "Hessian semi-supervised extreme learning machine," *Neurocomputing*, vol. 207, pp. 560–567, 2016.

[208] Y. Lei, L. Cen, X. Chen, and Y. Xie, "A hybrid regularization semi-supervised extreme learning machine method and its application," *IEEE Access*, vol. 7, pp. 30 102–30 111, 2019.

[209] A. Iosifidis, A. Tefas, and I. Pitas, "Regularized extreme learning machine for multi-view semi-supervised action recognition," *Neurocomputing*, vol. 145, pp. 250–262, 2014.

[210] 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 (Cybernetics)*, vol. 42, no. 2, pp. 513–529, 2011.

[211] W. Zhang, Q. J. Wu, Y. Yang, and T. Akilan, "Multimodel feature reinforcement framework using moore-penrose inverse for big data analysis," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[212] G. Druck, C. Pal, A. McCallum, and X. Zhu, "Semi-supervised classification with hybrid generative/discriminative methods," in *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2007, pp. 280–289.

[213] Z. Ju and H. Gu, "Predicting pupylation sites in prokaryotic proteins using semi-supervised self-training support vector machine algorithm," *Analytical biochemistry*, vol. 507, pp. 1–6, 2016.

[214] Y. Zou, Z. Yu, X. Liu, B. Kumar, and J. Wang, "Confidence regularized self-training," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 5982–5991.

[215] R. Sheikhpour, M. A. Sarram, S. Gharaghani, and M. A. Z. Chahooki, "A survey on semi-supervised feature selection methods," *Pattern Recognition*, vol. 64, pp. 141–158, 2017.

[216] Z.-H. Zhou and M. Li, "Semi-supervised regression with co-training." in *IJCAI*, vol. 5, 2005, pp. 908–913.

[217] S. Qiao, W. Shen, Z. Zhang, B. Wang, and A. Yuille, "Deep co-training for semi-supervised image recognition," in *Proceedings of the european conference on computer vision (eccv)*, 2018, pp. 135–152.

[218] V. J. Prakash and D. L. Nithya, "A survey on semi-supervised learning techniques," *arXiv preprint arXiv:1402.4645*, 2014.

[219] V. Garla, C. Taylor, and C. Brandt, "Semi-supervised clinical text classification with laplacian svms: an application to cancer case management," *Journal of biomedical informatics*, vol. 46, no. 5, pp. 869–875, 2013.

[220] J. Tanha, M. van Someren, and H. Afsarmanesh, "Semi-supervised self-training for decision tree classifiers," *International Journal of Machine Learning and Cybernetics*, vol. 8, no. 1, pp. 355–370, 2017.

[221] C. J. Burges, "A tutorial on support vector machines for pattern recognition," *Data mining and knowledge discovery*, vol. 2, no. 2, pp. 121–167, 1998.

[222] L. Szymanski and B. McCane, "Deep networks are effective encoders of periodicity," *IEEE transactions on neural networks and learning systems*, vol. 25, no. 10, pp. 1816–1827, 2014.

[223] P. W. Goldberg and M. R. Jerrum, “Bounding the vapnik-chervonenkis dimension of concept classes parameterized by real numbers,” *Machine Learning*, vol. 18, no. 2-3, pp. 131–148, 1995.

[224] P. Koiran and E. D. Sontag, “Neural networks with quadratic vc dimension,” *journal of computer and system sciences*, vol. 54, no. 1, pp. 190–198, 1997.

[225] D. H. Ballard, “Modular learning in neural networks.” in *AAAI*, 1987, pp. 279–284.

[226] F. Lu, J. Wu, J. Huang, and X. Qiu, “Restricted-boltzmann-based extreme learning machine for gas path fault diagnosis of turbofan engine,” *IEEE Transactions on Industrial Informatics*, vol. 16, no. 2, pp. 959–968, 2019.

[227] T. Wu, W. Xue, H. Wang, C. Chung, G. Wang, J. Peng, and Q. Yang, “Extreme learning machine-based state reconstruction for automatic attack filtering in cyber physical power system,” *IEEE Transactions on Industrial Informatics*, 2020.

[228] R. Katuwal, P. N. Suganthan, and M. Tanveer, “Random vector functional link neural network based ensemble deep learning,” *arXiv preprint arXiv:1907.00350*, 2019.

[229] G. Chao, Y. Luo, and W. Ding, “Recent advances in supervised dimension reduction: A survey,” *Machine learning and knowledge extraction*, vol. 1, no. 1, pp. 341–358, 2019.

[230] Y. Wang, W. Zhang, L. Wu, X. Lin, and X. Zhao, “Unsupervised metric fusion over multiview data by graph random walk-based cross-view diffusion,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 1, pp. 57–70, 2015.

[231] J. Xu, L. Xiang, Q. Liu, H. Gilmore, J. Wu, J. Tang, and A. Madabhushi, “Stacked sparse autoencoder (ssae) for nuclei detection on breast cancer histopathology images,” *IEEE Transactions on Medical Imaging*, vol. 35, no. 1, pp. 119–130, 2015.

[232] X. Zong-Ben, G. Hai-Liang, W. Yao, and H. ZHANG, "Representative of l1/2 regularization among lq (0¡ q 1) regularizations: an experimental study based on phase diagram," *Acta Automatica Sinica*, vol. 38, no. 7, pp. 1225–1228, 2012.

[233] J. Zeng, S. Lin, and Z. Xu, "Sparse regularization: Convergence of iterative jumping thresholding algorithm," *IEEE Transactions on Signal Processing*, vol. 64, no. 19, pp. 5106–5118, 2016.

[234] Z. Xu, X. Chang, F. Xu, and H. Zhang, "L1/2 regularization: A thresholding representation theory and a fast solver," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 7, pp. 1013–1027, 2012.

[235] M. M. Anthimopoulos, L. Gianola, L. Scarnato, P. Diem, and S. G. Mougiakakou, "A food recognition system for diabetic patients based on an optimized bag-of-features model," *IEEE Journal of Biomedical and Health Informatics*, vol. 18, no. 4, pp. 1261–1271, 2014.

[236] W. Wu, W. Sun, Q. J. Wu, Y. Yang, H. Zhang, W.-L. Zheng, and B.-L. Lu, "Multimodal vigilance estimation using deep learning," *IEEE Transactions on Cybernetics*, 2020.

[237] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[238] K.-K. Lurz, M. Bashiri, K. F. Willeke, A. K. Jagadish, E. Wang, E. Y. Walker, S. Cadena, T. Muhammad, E. Cobos, A. Tolias *et al.*, "Generalization in data-driven models of primary visual cortex," in *International Conference on Learning Representations*, 2021.

[239] S. Khan, N. Islam, Z. Jan, I. U. Din, and J. J. C. Rodrigues, "A novel deep learning based framework for the detection and classification of breast cancer using transfer learning," *Pattern Recognition Letters*, vol. 125, pp. 1–6, 2019.

[240] S. Malhotra, V. Kumar, and A. Agarwal, "Bidirectional transfer learning model for sentiment analysis of natural language," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–21, 2021.

[241] H. Chen, Y. Wang, G. Wang, and Y. Qiao, "Lstd: A low-shot transfer detector for object detection," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[242] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A comprehensive survey on transfer learning," *Proceedings of the IEEE*, vol. 109, no. 1, pp. 43–76, 2020.

[243] L. Wen, L. Gao, and X. Li, "A new deep transfer learning based on sparse auto-encoder for fault diagnosis," *IEEE Transactions on systems, man, and cybernetics: systems*, vol. 49, no. 1, pp. 136–144, 2017.

[244] T. Kurbiel and S. Khaleghian, "Training of deep neural networks based on distance measures using rmsprop," *arXiv preprint arXiv:1708.01911*, 2017.

[245] P. Courrieu, "Fast computation of moore-penrose inverse matrices," *arXiv preprint arXiv:0804.4809*, 2008.

[246] R. MacAusland, "The moore-penrose inverse and least squares," *Math 420: Advanced Topics in Linear Algebra*, pp. 1–10, 2014.

[247] M. A. Rakha, "On the moore–penrose generalized inverse matrix," *Applied Mathematics and Computation*, vol. 158, no. 1, pp. 185–200, 2004.

[248] H. Zhuang, Z. Lin, and K.-A. Toh, "Blockwise recursive moore-penrose inverse for network learning," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2021.

[249] J. Liu, L. Zuo, X. Xu, X. Zhang, J. Ren, Q. Fang, and X. Liu, "Efficient batch-mode reinforcement learning using extreme learning machines," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2019.

[250] L. Zhang and P. N. Suganthan, “A comprehensive evaluation of random vector functional link networks,” *Information Sciences*, vol. 367, pp. 1094–1105, 2016.

[251] A. Brock, T. Lim, J. M. Ritchie, and N. Weston, “Freezeout: Accelerate training by progressively freezing layers,” *arXiv preprint arXiv:1706.04983*, 2017.

[252] S. J. Pan and Q. Yang, “A survey on transfer learning,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[253] K. Kamnitsas, C. Baumgartner, C. Ledig, V. Newcombe, J. Simpson, A. Kane, D. Menon, A. Nori, A. Criminisi, D. Rueckert *et al.*, “Unsupervised domain adaptation in brain lesion segmentation with adversarial networks,” in *International Conference on Information Processing in Medical Imaging.* Springer, 2017, pp. 597–609.

[254] Y. Zhang, H. Chen, Y. Wei, P. Zhao, J. Cao, X. Fan, X. Lou, H. Liu, J. Hou, X. Han *et al.*, “From whole slide imaging to microscopy: Deep microscopy adaptation network for histopathology cancer image classification,” in *International Conference on Medical Image Computing and Computer-Assisted Intervention.* Springer, 2019, pp. 360–368.

[255] L. Chen, M. J. Gales, and K. Chin, “Constrained discriminative mapping transforms for unsupervised speaker adaptation,” in *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* IEEE, 2011, pp. 5344–5347.

[256] H.-y. Lee, Y.-y. Chou, Y.-B. Wang, and L.-s. Lee, “Unsupervised domain adaptation for spoken document summarization with structured support vector machine,” in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing.* IEEE, 2013, pp. 8347–8351.

[257] I. Tolstikhin, N. Houlsby, A. Kolesnikov, L. Beyer, X. Zhai, T. Unterthiner, J. Yung, D. Keysers, J. Uszkoreit, M. Lucic *et al.*, “Mlp-mixer: An all-mlp architecture for vision,” *arXiv preprint arXiv:2105.01601*, 2021.

Appendix A
IEEE Permission to Reprint

In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of University of Windsor's products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

Appendix B
Elsevier Permission to Reprint

In reference to Elsevier copyrighted material which is used with permission in this thesis, the Elsevier does not endorse any of University of Windsors products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing Elsevier copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please visit: `https://www.elsevier.com/about/our-business/policies/copyright#Author-rights`.

Vita Auctoris

NAME	:	Wandong Zhang
BIRTH YEAR	:	1993
BIRTH PLACE	:	China
EDUCATION		
2022	:	**Doctor of Philosophy**
		Electrical and Computer Engineering
		University of Windsor, Windsor, Ontario, Canada
2018	:	**Masters of Applied Science**
		Pattern Recognition and Intelligent System
		Ocean University of China, Shandong, China
2015	:	**Bachelors of Engineering**
		Automation
		Ocean University of China, Shandong, China