Not All Features Are Equal: Feature Selection Based on Deep Learning

Von der Fakultät Informatik, Elektrotechnik und Informationstechnik
der Universität Stuttgart
zur Erlangung der Würde eines Doktor-Ingenieurs (Dr.-Ing.)
genehmigte Abhandlung

vorgelegt von

Yiwen Liao
aus Hubei

Hauptberichter:	Prof. Dr.-Ing. Bin Yang
Mitberichter:	Prof. Dr. rer. nat. Dirk Pflüger

Tag der mündlichen Prüfung: 21.03.2024

Institut für Signalverarbeitung und Systemtheorie
der Universität Stuttgart

2025

Acknowledgment

I would like to express my heartfelt gratitude to the individuals who provided me with invaluable support throughout my doctoral project journey. Without them, this dissertation would not have been possible.

First and foremost, I am immensely grateful to Prof. Dr.-Ing Bin Yang for giving me the opportunity to work at the Institute of Signal Processing and System Theory. Throughout the entire project, he granted me the freedom to explore different ideas and fostered numerous collaborations within the graduate school. Moreover, he exhibited immense patience in guiding me from a graduate student to an independent researcher.

I extend my sincere thanks to my mentors, Jochen and Raphaël, from Advantest. They consistently supported me throughout the entire project and generously shared their ideas. It is largely due to them that I decided to join Advantest after completing my doctoral project.

I am also thankful to all my colleagues from ISS and GS-IMTR. It has been a pleasure working with them over the past few years. I particularly want to express my gratitude to Patrick, who also supervised my master's thesis. Despite our relatively short time working together, I gained a wealth of knowledge from him in terms of academic research. I would also like to thank Alex, who was always available to address any questions or requests I had. Many thanks are due to Elisabeth for her assistance with various daily tasks, such as preparing for business trips. Additionally, I am grateful to Prof. Hans-Joachim Wunderlich, Prof. Hussam Amrouch, Paria, and Paul for their successful collaborations.

Last but certainly not least, I would like to express my deepest appreciation to my family. My parents have always supported me in every decision I have made. The lessons I

have learned from them have significantly shaped my way of thinking and my approach to work and life. I am also grateful to my girlfriend, Siyuan, for her unwavering encouragement during times when I faced challenging questions and issues in my research. I cannot imagine having a better family than the one I have.

Finally, I would like to dedicate this work to my grandfather, in loving memory.

Contents

Zusammenfassung ix

Abstract xi

Acronyms xiii

List of Symbols xv

1 Introduction 1

1.1 Motivation 2

1.2 Research Goals and Contributions 4

1.3 Outline 7

2 Backgrounds 9

2.1 Fundamentals of Feature Selection 9

2.1.1 Definition 9

2.1.2 Feature Selection Pipeline 12

2.1.3 Categorization in Feature Selection 14

2.2 Deep-Learning-Based Feature Selection 17

2.2.1 Frameworks for DL-Based Feature Selection 17

2.2.2 Comments on the DL-Based Feature Selection Framework . . 19

2.3 Datasets 20

2.3.1 Industrial Datasets 20

2.3.2 Benchmarking Datasets 21

3 Feature Mask Approach 27

3.1 Introduction 27

3.2 Methodology 29
3.2.1 Structure and Learning Objective 29
3.2.2 Feature Mask Module 31
3.3 Evaluations 34
3.3.1 Setup 34
3.3.2 Main Results 36
3.3.3 Case Study: Synthetic Datasets 42
3.3.4 Case Study: The Post-Silicon Validation Dataset 44
3.3.5 Robustness of Selection 46
3.4 Insights of the Feature Mask Approach 49
3.4.1 Ablation Study 50
3.4.2 Resistance Against Redundant Features 52
3.4.3 Computational Complexity 53
3.4.4 Hyperparameters 54
3.5 Discussion 57
3.6 Summary 61

4 Conditional Feature Selection **63**
4.1 Introduction 63
4.2 Related Works 65
4.3 Methodology 67
4.3.1 Notations 67
4.3.2 Conditional Feature Selection 67
4.3.3 Learning Objective 69
4.3.4 Implementation 70
4.4 Evaluations on Synthetic Datasets 71
4.4.1 Comparison with Reference Methods 72
4.4.2 Single Preselected Feature as Condition 73
4.4.3 Conditional Feature Selection with Redundant Features 74
4.5 Evaluations on Real-World Dataset 75
4.5.1 Comparison with the State-Of-The-Art 75
4.5.2 Conditional Feature Selection for a Single Device 76
4.5.3 Conditional Feature Selection on Multiple Devices 77

4.5.4 Unexpected Discovery 79
4.6 Study of Hyperparameters 80
4.7 Summary 82

5 Complementary Feature Mask 85
5.1 Introduction 85
5.2 Related Works 87
5.3 Methodology 87
5.3.1 Complementary Feature Mask 88
5.3.2 Implementation 90
5.4 Evaluations 93
5.4.1 Setup 93
5.4.2 Main Results 94
5.5 Design of the Complementary Feature Mask 98
5.6 Discussion 100
5.7 Summary 102

6 Batch-Wise Attention for One-Class Classification 103
6.1 Introduction 104
6.2 Background 107
6.2.1 Autoencoders for One-Class Classification 107
6.2.2 Undesirable Behavior of Autoencoders 108
6.3 Methodology 111
6.3.1 The Batch-Wise Attention Block 111
6.3.2 Structure and Learning Objective 112
6.4 Evaluations 113
6.4.1 Setup 114
6.4.2 Main Results 115
6.5 Insights into Batch-Wise Attention 118
6.5.1 BWA-AE is Not AE with Small Latent Dimensions 119
6.5.2 Reconstruction 122
6.5.3 BWA for Different Capacities 122
6.5.4 Sensitivity to Batch Size 124

6.6 Case Study: Resistive Open Defect Identification 125
6.6.1 Background 125
6.6.2 The Simulation Data 126
6.6.3 Comparison with Unsupervised Methods 127
6.6.4 Comparison with Supervised Methods 128
6.6.5 Further Investigations 131
6.7 Batch-Wise Attention as a Plug-In 133
6.8 Summary 136

7 Conclusion and Future Work **139**
7.1 Conclusion 139
7.2 Future Work 141

Bibliography **145**

Zusammenfassung

Im Zeitalter von Big Data besteht eine der größten Herausforderungen in dem Maschinellen Lernen darin, hochdimensionale Daten mit großen Volumen effizient zu verarbeiten. Eine speziell entwickelte Technik zur Bewältigung dieser Herausforderung ist die Dimensionsreduktion. Dies ist eine zentrale Aufgabe im maschinellen Lernen und Data Mining, da durch die Reduktion der Dimensionalität Probleme der Überanpassung (Overfitting) gemildert, Rechen- und Speicherkosten gesenkt, die Leistungsfähigkeit von Lernmodellen verbessert und die Implementierung auf Edge-Geräten ermöglicht werden kann.

Grundsätzlich gibt es zwei Hauptansätze zur Dimensionsreduktion: Merkmalsumwandlung (Feature Transformation) und Merkmalsauswahl (Feature Selection). Bei der Merkmalsumwandlung werden neue Repräsentationen auf Basis der ursprünglichen Merkmale erzeugt, wobei Berechnungen wie Addition und Multiplikation verwendet werden. Allerdings führt diese Transformation häufig zum Verlust der ursprünglichen Bedeutung der Merkmale. Im Gegensatz dazu zielt die Merkmalsauswahl darauf ab, eine Teilmenge der ursprünglichen Merkmale auszuwählen, ohne dabei Berechnungen an den gewählten Merkmalen durchzuführen. Dadurch bleibt die physikalische Bedeutung der ursprünglichen Merkmale erhalten, was für das Verständnis der Daten und die Interpretierbarkeit der ausgewählten Merkmale von entscheidender Bedeutung ist.

Diese Dissertation konzentriert sich auf die Merkmalsauswahl. Insbesondere werden Konzepte aus dem Deep Learning genutzt, um fortschrittlichere Algorithmen für die Merkmalsauswahl zu entwickeln, die eine präzise Auswahlleistung mit hoher Effizienz für großskalige, hochdimensionale Daten ermöglichen. Der Kern dieser Arbeit ist eine neuartige Feature-Masken-Methode, die auf batchweiser Abschwächung und Normalisierung der Merkmalsmasken basiert. Auf dieser Grundlage wird ein neues

Framework namens Conditional Feature Selection vorgeschlagen, das Expertenwissen in den Trainingsprozess der Merkmalsauswahl integriert. Darüber hinaus wird eine Komplementäre Feature-Masken-Methode entwickelt, um die ursprüngliche Feature-Masken-Technik weiter zu verbessern. Über die Merkmalsauswahl hinaus wird das Feature-Masken-Konzept zu einem Batchweisen Aufmerksamkeitsmechanismus abstrahiert und in Autoencoder für One-Class-Klassifikation und Anomalieerkennung integriert.

Die experimentellen Ergebnisse dieser Dissertation zeigen, dass die vorgeschlagenen Methoden im Vergleich zu aktuellen Ansätzen auf synthetischen, Benchmark- und realen Datensätzen eine führende Leistung erzielen. Darüber hinaus sind die entwickelten Verfahren in der Lage, großskalige Daten effizient zu verarbeiten und zeichnen sich durch eine geringe Anzahl an empfindlichen Hyperparametern aus, wodurch sie besonders benutzerfreundlich sind. Schließlich sind die in dieser Arbeit entwickelten Methoden allgemein anwendbar und können weiterführende Forschung in einem breiten Anwendungsspektrum inspirieren.

Abstract

In the era of big data, one of the major challenges in the machine learning community is to handle high-dimensional data with large volumes. A dedicated technique known as dimension reduction is typically employed for this purpose. This is a crucial task in machine learning and data mining, because by reducing the dimensionality we can alleviate overfitting problems, reduce computational and storage costs, increase the performance of learning machines, and enable deployment on edge devices.

Broadly speaking, there are two major directions in dimension reduction, namely feature transformation and feature selection. Feature transformation involves creating new representations based on the raw features, leveraging calculations such as addition and multiplication. However, the transformation often leads to the loss of the meanings of raw features. Conversely, feature selection aims to select a subset of raw features without performing any calculation on the selected features. This means that feature selection can preserve the original physical meaning of the raw features. This property is extremely important for understanding data and maintaining the interpretability of selected features.

This dissertation focuses on feature selection. Specifically, we leverage the concepts from the deep learning community to establish more advanced feature selection algorithms, which can provide accurate selection performance with high efficiency facing large-scale data of high dimensionality. The core idea of this dissertation is a novel Feature Mask method based on batch-wise attenuation and feature mask normalization. Starting from this, a novel framework, Conditional Feature Selection, is proposed to integrate expert knowledge to feature selection training. In addition, a novel Complementary Feature Mask approach is proposed to further enhance the original Feature Mask method. Beyond feature selection itself, we abstract the Feature Mask idea

to a Batch-Wise Attention mechanism and apply it to autoencoders in the context of one-class classification and anomaly detection.

Experimental results of this dissertation show that the proposed methodologies achieve state-of-the-art performance in comparison with contemporary approaches on synthetic, benchmarking, and real-world datasets. Moreover, the proposed methods in this dissertation can handle large-scale data and are easy to use due to their limited insensitive hyperparameters. Last but not least, the proposed methodologies are generic and can inspire subsequent research in a broad area.

Acronyms

ACCU	Accuracy
ADGAN	Anomaly Detection using Generative Adversarial Networks
AE	Autoencoder
AFS	Attention-based Feature Selection
AUC	Area Under the Receiver Operating Characteristic Curve
BA	Batch-wise Average
BWA	Batch-Wise Attention
CFM	Complementary Feature Mask
CFS	Conditional Feature Selection
ConAE	Concrete Autoencoder
DFS	Deep Feature Selection
DL	Deep Learning
DSVDD	Deep Support Vector Data Description
DUT	Device Under Test
ERT	Extremely Randomized Tree
FM	Feature Mask
FMN	Feature Mask Normalization
FNR	False Negative Rate
FoM	Figure-of-Merit
FS	Feature Selection
GPU	Graphics Processing Unit
ICS	Intra-Class Splitting
IoT	Internet of Things
kNN	k-Nearest Neighbor
LASSO	Least Absolute Shrinkage and Selection Operator

LOF	Local Outlier Factor
LSA	Latent Space Autoregression
mRMR	Minimum Redundancy and Maximum Relevance
MSE	Mean Squared Error
NN	Neural Network
OCC	One-Class Classification
OCGAN	one-Class Generative Adversarial Network
OCSVM	One-Class Support Vector Machine
PDF	Probability Density Function
PMF	Probability Mass Function
PSV	Post-Silicon Validation
RawF	Raw Features
RF	Random Forest
RFR	Random Forest Regression
RSF	Randomly Selected Features
SFS	Sequential Forward Selection
SPICE	Simulation Program with Integrated Circuit Emphasis
SVM	Support Vector Machine
UMAP	Uniform Manifold Approximation and Projection

List of Symbols

x	Scalar
$\boldsymbol{x}$	Column vector
$\mathbf{X}$	Matrix
x	Random variable
$\mathbf{x}$	Random vector
$\mathcal{X}$	Set
$\boldsymbol{x}^T$	Transpose of a vector
x_i	The i-th element of a vector
$\boldsymbol{x}_i$	The i-th vector of a set
$\mathbb{R}^d$	d-dimensional space
$f_{\boldsymbol{\Theta}}(\cdot)$ or $f(\cdot;\boldsymbol{\Theta})$	a function $f(\cdot)$ parameterized by $\boldsymbol{\Theta}$
$\odot$	Hadamard product (element-wise multiplication)
$\boxtimes$	Concatenation
∇	Nabla operator
$\mathbb{E}[\cdot]$	Expectation
$\|\|\cdot\|\|_2^2$	Squared 2-norm
$\sin(\cdot)$	Sine
$\cos(\cdot)$	Cosine
$\min(\boldsymbol{x})$	The minimum element of the vector $\boldsymbol{x}$
$\max(\boldsymbol{x})$	The maximal element of the vector $\boldsymbol{x}$
$x \leftarrow y$	Assignment of y to x
$\underset{\boldsymbol{\Theta}}{\operatorname{argmin}}$	Argument $\boldsymbol{\Theta}$, for which the minimum is obtained

Chapter 1

Introduction

Before diving into this dissertation, let us consider some common scenarios below:

- When we want to buy a new mobile phone, we often compare brands, prices, sizes, connection quality, camera performance and extension capabilities of different mobile phones. Clearly, different groups of people may prioritize different factors. For example, a student might more care about prices and camera performance, while a businessman might prioritize brands and connection quality.
- When we receive several job offers from different companies, we typically compare the compensations, career development opportunities, working environment, company benefits and locations of those companies. Naturally, a good job offer has different meanings for different groups. A graduated student may more focus on career development opportunities and compensations, while new parents may value locations and working environments more.
- When we plan a road trip, it is important for us to consider factors such as distances between locations, the time required to travel, the availability of accommodations, and the popularity of the destinations. A rookie road traveler may more care about the availability of accommodations and the popularity of the destinations, while a person with limited vacation might mainly consider the required time to travel.

Actually, from the machine learning perspective, the factors we consider in the examples above are understood as *features*, which characterize a specific target. Interestingly, as

shown above, given different targets (e.g. a good mobile phone for a student or a good mobile phone for a businessman), the most influential factors (features) are different.

Apparently, from the aforementioned daily scenarios, we can clearly see that not all features are equal for different objectives. Nevertheless, we humans are born with the ability to select the most influential factors for a certain target in order to make better decisions for ourselves. However, this ability is limited. Sometimes we may already feel stressed when we face only a couple of factors to make a good decision.

Nowadays, we are in the era of information and big data. This means that we have to deal with hundreds to thousands of features and may be presented with millions of historical data. Unfortunately, the inherent ability of humans seems not sufficient anymore. We are asking: How can we *extend* our ability to perform feature selection on big data? This dissertation presents our thoughts and answers to this question.

1.1 Motivation

Feature Selection (FS) is well known as a critical research domain in the data science and machine learning community. Broadly speaking, feature selection reduces the data dimensionality by selecting a subset of the candidate features. Accordingly, the data with the reduced features can accelerate the subsequent learning procedures and can also lead to more robust performance on the downstream tasks as shown in many previous works such as [6, 18, 21, 40, 67, 111]. Therefore, in conventional machine learning pipelines, FS is often considered as an important preprocessing step to construct powerful and robust learning systems. Due to the nature of feature selection that we only *select* some of the candidate features instead of *transforming* the raw features, feature selection enjoys a property of preserving the original physical meaning of the candidate features. This is extremely important when feature selection is used as an intelligent tool for experts from different fields to analyze the salient quantities of the data, including but not limited to gene analysis [8, 28, 69, 83], chemistry [46, 50, 127], medical domain [97, 100, 110] and semiconductor validation [34, 53, 121].

It is clear to see that the importance of feature selection is not only limited to the machine learning community but also interdisciplinary. Therefore, feature selection has been intensively studied over the last decades with many survey papers such as [18, 21, 40, 67, 111]. Intensive research activities [30, 58, 115] on feature selection exploded in the late 90's of the 20th century, during which the categorization of different feature selection approaches became clear and was generally accepted by the community. For example, two important categories, the filter [7, 57] and the wrapper [58] methods, were formally proposed during this period. Some approaches that belong to these two categories are nowadays still active and frequently used, including but not limited to chi square test [128], mutual-information-based feature selection [119], and forward and backward selection [3, 101, 120].

Since the 2010s, neural networks (NN) and deep learning (DL) have suddenly drawn everyone's attention due to their enormous success in many fields such as computer vision [49, 63, 102] and natural language processing [27, 99]. The rise of DL is actually not a coincidence. Firstly, with the development of the Internet and online services, we are now in the era of big data, which requires an intelligent tool to deal with the incredibly huge amount of data. Secondly, the advances in parallel and distributed computing in both software perspective (e.g. TensorFlow [1] and PyTorch [92]) and hardware perspective (e.g. Compute Unified Device Architecture of NVIDIA) have also contributed to the prospering of NN and DL.

Naturally, many recent studies in feature selection have taken concepts from the deep learning community. Representative prior works include but are not limited to leveraging attention mechanism to automatically identify the most salient parts of the input data in a sample-wise level [37, 122] and designing novel sparsity penalty terms to obtain sparse feature importance vectors [14, 69, 117]. According to these studies, we can easily identify several advantages of using DL techniques for the FS problem. Firstly, deep neural networks can be efficiently trained on Graphics Processing Units (GPUs) based on parallel computing, enabling a significantly higher computational speed than conventional machine learning approaches. Secondly, deep neural networks are highly modular, meaning that we can easily modify individual components to meet different requirements in practice. Finally, neural networks are more friendly to various data types, which has been unfortunately a challenge to many conventional FS approaches.

Therefore, it is natural and reasonable to design more reliable and better FS algorithms based on deep learning.

Despite the advancement of DL in the feature selection domain, the existing DL-based FS methods are not optimal and there are still several noticeable disadvantages. Firstly, most of them are born with extra loss terms with respect to different constraints, requiring a heavy fine-tuning of hyperparameters in order to obtain a satisfying feature selection quality, which notably enlarges the actual time consumption and is not friendly to non-DL experts. Secondly, some of them use overly large network architectures, leading to large computational complexity. In addition, many DL-based FS approaches can be considered to be inherited from the conventional FS methods such as LASSO [115] or Elastic Net [133], so they cannot well consider the non-linear relationship among the candidate features. Last but not least, to the best of our knowledge, the existing DL-based FS-methods have not leveraged potentially available expert knowledge and are purely data-driven, which might lead to inaccurate solutions. Overall, there is an urgent need for an advanced DL-based feature selection approach.

1.2 Research Goals and Contributions

Motivated by the necessity of feature selection and the enormous benefits of deep learning, this dissertation targets studying feature selection problems by using deep learning techniques and concepts. In summary, the main research objectives of this thesis are listed below:

1. We aim to design and develop a DL-based feature selection algorithm that has as few hyperparameters as possible, while it can still achieve comparable or even better feature selection performance than its state-of-the-art counterparts.

2. As FS algorithms are often used as a tool to assist domain experts to analyze big data, we aim at proposing a general framework that integrates expert knowledge into the feature selection algorithm to avoid bias or inaccurate selection results.

3. One research objective is to study how to further improve the DL-based feature selection performance in scenarios where only small feature subset sizes are

allowed. We aim to provide a new paradigm for designing the next DL-based feature selection algorithms.

4. We target bridging feature selection and attention mechanism and demonstrating to leverage the proposed DL-based feature selection algorithms in the areas beyond FS, which justifies the potential of our research.

5. The final research objective is to validate the proposed and developed feature selection algorithms on semiconductor industrial datasets. This is to examine the performance of our algorithms in real-world applications.

To achieve the research goals mentioned above, this dissertation mainly has the following contributions to characterize its novelty:

- A novel Feature Mask approach [72, 75] for feature selection based on batch-wise average and feature mask normalization leveraging neural networks;

- A generic framework called Conditional Feature Selection [73, 77] that for the first time integrates expert knowledge to feature selection and achieves more accurate selection results in real-world applications;

- A novel Complementary Feature Mask method [78] that significantly improves feature selection quality for small feature subset sizes with high stability;

- A novel Batch-Wise Attention mechanism [71, 74, 79] inherited from the feature mask method for enhancing one-class classification performance and anomaly detection capability of autoencoders.

During achieving the research goals, partial contributions of this dissertation have been published in advance:

- Yiwen Liao et al. [2021]. "Feature Selection Using Batch-Wise Attenuation and Feature Mask Normalization". *2021 International Joint Conference on Neural Networks (IJCNN)*. DOI: 10.1109/IJCNN52387.2021.9533531.

- Yiwen Liao et al. [2021]. "Anomaly Detection Based on Selection and Weighting in Latent Space". *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*. DOI: 10.1109/CASE49439.2021.9551267.

- Yiwen Liao et al. [2021]. “A Learning-Aided Generic Framework for Fault Detection and Recovery of Inertial Sensors in Automated Driving Systems”. *IEEE Systems Journal*, vol. 15, no. 2, pp. 3001-3011, June 2021, DOI: 10.1109/JSYST.2020.3004805.

- Yiwen Liao et al. [2021]. “Unsupervised fault detection and recovery for intelligent robotic rollators”. *Robotics and Autonomous Systems*, vol. 146, August 2021, DOI: 10.1016/j.robot.2021.103876.

- Yiwen Liao et al. [2022]. “To Generalize or Not to Generalize: Towards Autoencoders in One-Class Classification”. *2022 International Joint Conference on Neural Networks (IJCNN)*. DOI: 10.1109/IJCNN55064.2022.9892812.

- Yiwen Liao et al. [2022]. “Efficient and Robust Resistive Open Defect Detection Based on Unsupervised Deep Learning”. *2022 IEEE International Test Conference (ITC)*. DOI: 10.1109/ITC50671.2022.00026.

- Yiwen Liao et al. [2022]. “Wafer Map Defect Classification Based on the Fusion of Pattern and Pixel Information”. *2022 IEEE International Test Conference (ITC)*. DOI: 10.1109/ITC50671.2022.00006.

- Yiwen Liao et al. [2022]. “A Deep-Learning-Aided Pipeline for Efficient Post-Silicon Tuning”. *34th workshop on Test Methods and Reliability of Circuits and Systems 2022 (TuZ)*. DOI: 10.48550/arxiv.2207.00336.

- Yiwen Liao et al. [2022]. “Conditional Variable Selection for Intelligent Test”. *1st Workshop on Intelligent Methods for Test and Reliability (IMTR)*. DOI: 10.48550/arxiv.2207.00335.

- Yiwen Liao et al. [2022]. “Deep Feature Selection Using a Novel Complementary Feature Mask”. *preprint*. DOI: 10.48550/arxiv.2209.12282.

- Yiwen Liao et al. [2022]. “Experts in the Loop: Conditional Variable Selection for Accelerating Post-Silicon Analysis Based on Deep Learning”. *preprint*. DOI: 10.48550/arxiv.2209.15249.

1.3 Outline

The outline of this thesis is structured as follows.

Chapter 2 gives the background for feature selection. Formal definitions of feature selection, feature relevance and redundancy are provided here. In addition, the typical feature selection pipeline, popular categorization in feature selection and a general framework for most DL-based FS approaches are presented. Finally, some important benchmarking datasets are introduced.

The Feature Mask method, as the core of this thesis, is presented in Chapter 3. The detailed design and architecture are discussed here. Furthermore, evaluation on benchmarking datasets, synthetic datasets and real-world industrial datasets are conducted.

Following the FM-method, Chapter 4 deals with the problem of integrating expert knowledge to feature selection by proposing a novel framework of Conditional Feature Selection (CFS). In this chapter, we provide a detailed real-world background to explain the necessity of CFS in terms of post-silicon validation. This chapter ends with an intensive evaluation of our method on the synthetic datasets and real-world datasets.

Chapter 5 targets presenting a novel Complementary Feature Mask that further enhances feature selection performance for different DL-based methods. This chapter especially discusses how to design the complementary feature mask and the derived multi-task learning architecture.

In Chapter 6, the core idea of this dissertation, the FM-module, is abstracted as a batch-wise attention mechanism and used for one-class classification and anomaly detection. We start with the unexpected behaviors of autoencoders in one-class classification. Then, we provide a neat solution by leveraging the batch-wise attention mechanism, which is evaluated on two benchmarking datasets. Furthermore, we included two case studies, where the idea is validated on resistive open defect identification and industrial intrusion detection scenarios.

Finally, this thesis ends with Chapter 7 with a summary of major research results and comments on future research in this direction.

Chapter 2

Backgrounds

2.1 Fundamentals of Feature Selection

This section presents the background knowledge on feature selection. We start from several important definitions in feature selection, including but not limited to relevant and irrelevant features. Following that, we briefly introduce the well accepted categorization of different feature selection approaches in literature. Subsequently, we explain the typical pipeline of training and evaluating feature selection algorithms. Next, we focus on the modern feature selection approaches using deep learning concepts. Finally, this section ends up with an introduction to the major datasets used in this dissertation.

2.1.1 Definition

In literature, feature selection is also understood as variable or attribute selection in different fields [40, 67]. In this dissertation, without explicit notes, we mainly use the term feature selection, which is more frequent in the machine learning community. Features, unless otherwise stated, are the quantities that characterize the collected data. For example, in medical diagnosis, we can use the features like gender, age, weight, height and blood pressure to characterize the health status for an individual patient. This example also shows an important challenge in feature selection: We may

face different data types, including numerical continuous (weight, height and blood pressure), numerical discrete (age), and categorical (gender) values.

Notoriously, feature selection primarily aims *i*) to identify the optimum feature subset of a given subset size and *ii*) to seek for the optimum subset size. This dissertation mainly focuses on the first research question because many use cases require a feature subset of a desired size due to computational and storage limitations in practice.

Formally, a d-dimensional data point, also known as a sample, a pattern vector or a feature vector, is defined as a column vector $\boldsymbol{x} = [x_1, x_2, \ldots, x_d]^T \in \mathbb{R}^d$, where the i-th entry denotes the value of the i-th feature. Frequently, a feature vector is paired with a label y for classification or regression which is also known as ground truth. From a statistical perspective, a feature vector is a realization of a random vector $\mathbf{x} = [\mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{x}_d]^T$ with the notation of probability as $P(\mathbf{x} = \boldsymbol{x})$. Likewise, y is a realization of a random variable y. Thereby, feature selection aims to find out a subset $\mathcal{V} \subset \mathcal{X} = \{\mathrm{x}_1, \ldots, \mathrm{x}_d\}$ that can still well characterize the original data for a certain given downstream task; i.e. $\mathcal{V}$ should well preserve the predictive power for y.

Feature Relevance

According to literature review, there are indeed a couple of different definitions on feature relevance [42, 58]. The difficulty is that given the same dataset, different definitions may yield different feature selection results. We refer the interested readers to the original paper [58] for a detailed comparison. This dissertation jointly uses the definition in [58] and [42] for better readability.

Specifically, $\mathbf{s}_i$ denotes the random vector without x_i as $\mathbf{s}_i = [\mathrm{x}_1, \ldots, \mathrm{x}_{i-1}, \mathrm{x}_{i+1}, \ldots, \mathrm{x}_d]^T$. $P(\cdot)$ denotes the probability density function (PDF) for continuous random variables and probability mass function (PMF) for discrete random variables. We do not discriminate PDF and PMF in the following definitions because the definitions below are generic and valid for both PDF and PMF.

Definition 1 (Relevant Features) *A feature* x_i *is relevant if and only if there exists* x_i*,* y *and* $\boldsymbol{s}_i$ *for which* $P(\mathrm{x}_i = x_i, \mathbf{s}_i = \boldsymbol{s}_i) > 0$ *such that*

$$P(\mathrm{y} = y|\mathbf{s}_i = \boldsymbol{s}_i, \mathrm{x}_i = x_i) \neq P(\mathrm{y} = y|\mathbf{s}_i = \boldsymbol{s}_i) \tag{2.1}$$

The definition above actually states that all features that can change the estimations on the give target variable y are relevant. This definition matches most applications, where we typically want to use features to predict certain targets in terms of a discrete-valued classification or continuous-valued regression.

Remarks: In literature, we often encounter strongly relevant and weakly relevant features. This qualitative difference comes from several reasons. Firstly, the target is often determined by several differently weighted relevant features. For example, let $\mathrm{y} = 10\mathrm{x}_1 + \mathrm{x}_2$ with $\mathrm{x}_1, \mathrm{x}_2 \sim N(0, 1)$. In this case, x_1 obviously has more predictive power than x_2. Moreover, data can contain noise due to the imperfection of the data collection process. In this case, strongly relevant features are the features that are less polluted by noise, while weakly relevant features are those that are more polluted by noise. Likewise, less polluted features often have better predictive power. In this dissertation, we do not rigorously discriminate strong and weak relevance because it is difficult to justify them in practice due to the absence of the ground truth of the optimal feature subset. Nevertheless, we may refer to both terms for synthetic datasets, where the predictive power of individual candidate features is known.

Based on the definitions on relevance, irrelevant features are defined as follows.

Definition 2 (Irrelevant Features) *A feature* x_i *is irrelevant if and only if it is not relevant, i.e.*

$$P(\mathrm{y} = y|\mathbf{s}_i = \boldsymbol{s}_i, \mathrm{x}_i = x_i) = P(\mathrm{y} = y|\mathbf{s}_i = \boldsymbol{s}_i) \tag{2.2}$$

It is clear to see that irrelevant features are the features that provide no predictive power for a given target variable with respect to a given downstream task.

Another prevalent term in feature selection is redundancy. However, there is no universally accepted definition of redundancy. Many previous studies used correlation to

indicate redundancy. However, in [42], the authors provided some toy examples to show that correlated features together can be relevant and contribute to the final prediction. Therefore, in this paper, we give the following pragmatic definition of redundancy to enable better understanding in the experimental analysis in the subsequent chapters.

Definition 3 (Redundant Features) *A feature* x_i *is redundant if it is relevant and there exists a subset* $\mathbf{s}'$ *of* $\mathbf{x} \setminus \{\mathrm{x}_i\}$ *for which* $P(\mathrm{x}_i = x_i, \mathbf{s}'_i = \boldsymbol{s}'_i) > 0$ *such that*

$$\mathrm{x}_i = f(\mathbf{s}') + \alpha \cdot \epsilon \tag{2.3}$$

In this definition, $f(\cdot)$ is a deterministic function, ϵ is an unknown random noise and $\alpha \geq 0$ denotes how much noise is added to $f(\mathbf{s}')$. This means, the feature x_i can be predicted by a deterministic function of $\mathbf{s}'$ up to a random uncertainty $\alpha\epsilon$. In practical scenarios, we find that even highly redundant features are never deterministic functions of each other due to noise in data collection and feature extraction. In addition, we can generalize this definition to a subset of features as well. For example, let $\mathbf{s}_1$ and $\mathbf{s}_2$ be two disjoint subsets of $\mathbf{x}$. Then, $\mathbf{s}_2$ is redundant if $\mathbf{s}_2 = f(\mathbf{s}_1) + \alpha \cdot \epsilon$. As described before, data collection can introduce noise, so the term $\alpha \cdot \epsilon$ also mimics the situation where one of the redundant feature pairs may be more polluted by the noise.

Definition 4 (Absolutely Redundant Features) *A feature* x_i *is absolutely redundant to a feature subset* $\mathbf{s}'$ *of* $\mathbf{x}$ *if it is redundant to the subset* $\mathbf{s}'$ *and* $\alpha = 0$.

In the experiments on the synthetic datasets which are presented in the subsequent chapters, absolutely redundant features are often encountered as the trivial cases for evaluating whether feature selection approaches can deal with redundant features.

2.1.2 Feature Selection Pipeline

Unlike directly training a dedicated classification or regression model, feature selection has its own pipeline in practice. To elaborate this, without loss of generality, this section considers feature selection with respect to a downstream classification task.

During the first stage, a feature selection algorithm is trained on the original input data with all candidate features, in which a specific classifier A can get involved for wrapper and embedded approaches. We will see more details on these two categories in the next section and here we do not need to know the concrete definitions of these two categories. After training the feature selection algorithm, we can obtain feature subsets of different sizes. The size is typically predefined by users and practitioners.

In order to justify the selection quality (i.e. which predictive power each feature subset has), a separate classifier B is needed. It should be noted that it is required that the training procedure of classifier B is independent of that of A, although both classifier A and classifier B can be of the same type of algorithms (e.g. both A and B are neural networks). Subsequently, we use the training data with selected features only to train the classifier B. Next, we feed a separate test dataset consisting only of the selected features into the trained classifier B to obtain the final prediction results. Note that classifier A was trained on all features for the purpose of feature selection and can thus be used to evaluate the selected features only. Finally, a predefined metric such as accuracy can be calculated based on the predictions made on the test set using the selected features only, providing an indicator of the quality of the selected features. Obviously, in feature selection, we typically do not have ground truth for the optimal feature subsets and rely on indirect metrics derived by downstream learning tasks.

Remarks: Feature selection is actually a discrete optimization problem. Unfortunately, it is practically infeasible to solve the problem directly (such as an exhaustive search). Therefore, many methods presented in the following sections aim to solve a relaxed feature selection problem. More precisely, many filter methods or embedded methods learn a continuous-valued feature importance score vector instead of a binary vector of ones and zeros. Obviously, the solution to the relaxed problem may differ from the optimum of the original feature selection problem. Nevertheless, according to [42], solving the relaxed problem has a higher likelihood to obtain a solution that can better generalize to unseen data, which is important for real-world scenarios.

2.1.3 Categorization in Feature Selection

Feature selection has been widely and intensively studied over the last few decades. Due to the bloom in this area, we have seen considerable novel techniques developed for feature selection such as LASSO regularization [115], elastic networks [133], concrete autoencoders [4] and sequential forward attention [10]. Despite various research directions, it is generally acknowledged that most feature selection approaches can be categorized into three groups, i.e. filter methods, wrapper methods and embedded methods. This section briefly introduces the categorization in feature selection according to the literature [40, 67] and discusses our view on the current trends.

Filter methods

Filter methods can track back to about thirty years ago [7, 57]. Literally, we aim to filter out some features according to certain predefined criteria independent of a specific learning algorithm. For example, an easy way is to remove the candidate features with extremely small variance, indicating that they have limited predictive power for downstream tasks. Some methods [67] took ideas from statistical analysis and calculate the correlation between the candidate features and the target variables (e.g. class labels for classification or numerical values for regression). In addition, it is also popular to take advantage from information theory, where some prior works [119] compared the mutual information between candidate features and target variables to identify the important or irrelevant features. As can be easily seen, filter methods typically do not involve any specific learning algorithms and can therefore efficiently identify critical candidate features. Despite their efficiency, conventional filter methods also have certain limitations such as inaccurate selection results due to the absence of learning algorithms as well as the restriction to linear models in some cases. It should be additionally noted that filter methods are often regarded as a preprocessing step because no downstream learning algorithms are involved at this stage. Therefore, it is often required that the quality of the selected features should be further evaluated in downstream tasks afterwards.

Wrapper methods

Wrapper methods were proposed in 1997 by Ron Kohavi and George H. John [58]. In contrast to filter methods, wrapper methods consider learning algorithms as a black-box for evaluating the quality of different feature subsets. That is to say, wrapper methods use feature subsets to train black-box algorithms for the downstream task and leverage the resulting learning performance (e.g. accuracy or fitting errors) as indicators to justify the current selection quality. Accordingly, for wrapper methods, the selected features are more suitable for the corresponding learning algorithm in downstream tasks. However, such better selection performance requires significantly longer training time than filter methods because for each modification of the selected features a complete ML model has to be trained and evaluated for the downstream task. This is unfortunately often infeasible in practice. Therefore, prior studies on the wrapper methods mainly focused on designing novel search strategies such as greedy search or best-first search algorithms [58]. Accordingly, wrapper methods provide a general framework of performing feature selection without specifying a concrete learning algorithm. That is to say, given a wrapper method, users can choose their preferred black-box algorithms for the downstream task based on their expertise.

Embedded methods

Embedded methods [64] became well-known since the beginning of 2000. They embed feature selection into a specific learning machine (e.g. a support vector machine or a neural network). Simply speaking, embedded methods aim to learn the importance scores for the candidate features jointly with a dedicated learning algorithm targeting a predefined task like classification or regression. A typical constraint on the joint optimization problem is forcing the sparsity in those coefficients as in [115] with the intention that critical candidate features can have non-zero coefficients after training and be selected accordingly. Obviously, embedded methods can be regarded as taking advantages from both filter and wrapper methods to have a lower computational complexity than wrapper methods because we solve a relaxed optimization problem. In practice, however, embedded methods have their own limitations. For example,

we actually cannot guarantee sparsity if we use non-linear models as our downstream learning machine [10], although in this case the values of the learned coefficients can be still considered as importance of corresponding candidate features. In addition, there is no guarantee that the solution of the relaxed feature selection problem is also the solution for the original feature selection problem. Thereby, to better justify the quality of the selected features, practically it is still required to evaluate selected feature combinations of different sizes [42].

Comments

Although in literature feature selection approaches are roughly categorized into the three groups described above, it should be admitted that the boundaries between different groups are actually not clear. In other words, it is often seen that existing approaches may contain properties from different groups. For example, in [42], a decision tree was used to provide the relevance index for a filter FS method. This means that a learning machine is involved in a filter method. In addition, as can be seen, FS is actually a discrete combinational optimization problem. As a result, search strategies can play a critical role for all three groups. For example, methods from all three groups can use the identical search strategy like sequential search or stochastic search in practice as stated in [42]. In a nutshell, with the development of FS approaches, it becomes less important to identify the categorization of an FS method. We give the descriptions above mainly to provide a brief overview of the feature selection history.

Recent trends

With the dramatic advances in deep learning, recent FS studies have started taking concepts from the deep learning community to build more robust and efficient FS algorithms including but not limited to [4, 14, 37, 69]. These methods often simultaneously learn to weight candidate features as well as the suitable parameters of the corresponding downstream neural networks. In this sense, DL-based FS methods can be categorized as embedded methods. Nevertheless, sometimes we only use the

learned importance scores to select the most pivotal candidate features. In this aspect, DL-based FS methods have some similarity to filter methods as well as shown in [42]. In order to give readers a clearer map on deep-learning-based feature selection approaches, the next section aims to provide a conscious and generic structure of most DL-based feature selection methods.

2.2 Deep-Learning-Based Feature Selection

This section uses the following notations. Let the input data with the candidate features be as $\mathcal{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$ with n samples, each sample is denoted as a column vector $\boldsymbol{x}_i \in \mathbb{R}^d$, meaning that we have d candidate features. In the following, the input data are also denoted as a matrix $\mathbf{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n]^T \in \mathbb{R}^{n \times d}$ and we use both notations interchangeably for better readability. Without loss of generality, we suppose that the ground truth label for each sample is a scalar in a supervised learning setup for classification, i.e. $\boldsymbol{y} = [y_1, y_2, \ldots, y_n]^T$ with $y_i \in \{1, 2, \ldots, c\}$ where c is the number of classes. Typically, we use one-hot-coding representations for the class labels and $\boldsymbol{y}$ can be thus formulated as $\mathbf{Y} = [\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_n]^T \in \mathbb{R}^{n \times c}$ where $\boldsymbol{e}_i \in \{0, 1\}^c$ and the only non-zero entry is the affiliation of the corresponding sample $\boldsymbol{x}_i$; i.e. the y_i-th entry of $\boldsymbol{e}_i$ is one while the other entries are zeros.

2.2.1 Frameworks for DL-Based Feature Selection

Based on the definition of feature selection presented in the last section, generally speaking, many prevalent DL-based feature selection approaches can be visualized into a structure as shown in Figure 2.1. In particular, they jointly learn a feature mask $\boldsymbol{m} = [m_1, m_2, \ldots, m_d]^T \in \mathbb{R}^d$ and a downstream neural network $g_{\boldsymbol{\Theta}_n}(\cdot)$ so that the prediction performance is maximized with respect to some carefully designed constraints on $\boldsymbol{m}$ and/or special generation process of $\boldsymbol{m}$.

Each entry m_i of $\boldsymbol{m}$ in this case indicates the importance of the corresponding i-th candidate feature, so m_i is typically non-negative. Hence, $\boldsymbol{m}$ is often called feature importance score vector. Some feature selection approaches such as [14, 69] are based on

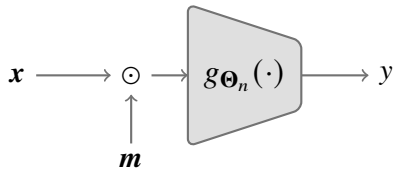


Figure 2.1: A typical DL-based feature selection framework. It jointly learns the feature mask $\boldsymbol{m}$ and a neural network $g_{\boldsymbol{\Theta}_n}(\cdot)$ dedicated to a given downstream learning task such as regression or classification.

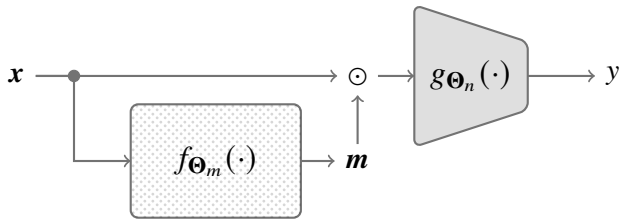


Figure 2.2: A variant of the DL-based feature selection framework. It is inspired by the attention mechanism and $\boldsymbol{m}$ is directly calculated from the input data $\boldsymbol{x}$ using a separate neural network parameterized by $\boldsymbol{\Theta}_m$.

this framework and assume that $\boldsymbol{m}$ is independent of $\boldsymbol{x}$ during the forward propagation, which are thereby mostly inspired by the ideas from LASSO regression [115].

During training, for these approaches, additional regularization $\mathcal{R}(\cdot)$ is often applied to the feature mask $\boldsymbol{m}$ in order to achieve some desired properties like sparsity in $\boldsymbol{m}$. In total, the learning objective of such DL-based feature selection methods can be formulated as

$$\min_{\boldsymbol{m},\boldsymbol{\Theta}_n} \mathcal{L}(g(\boldsymbol{x} \odot \boldsymbol{m}), y) + \lambda \cdot \mathcal{R}(\boldsymbol{m}), \tag{2.4}$$

where we omit the subscript for $g_{\boldsymbol{\Theta}_n}(\cdot)$ for simplicity.

Starting from the framework above, a variant driven by attention mechanism is presented in Figure 2.2 and representative prior works include [37]. In this design, $\boldsymbol{m}$ is directly calculated from the input data as $\boldsymbol{m} = f_{\boldsymbol{\Theta}_m}(\boldsymbol{x})$. Accordingly, the overall learning objective of this variant is modified to

$$\min_{\boldsymbol{\Theta}_m,\boldsymbol{\Theta}_n} \mathcal{L}(g(\boldsymbol{x} \odot f_{\boldsymbol{\Theta}_m}(\boldsymbol{x}), y) + \lambda \cdot \mathcal{R}(f_{\boldsymbol{\Theta}_m}(\boldsymbol{x})), \tag{2.5}$$

where the feature mask $\boldsymbol{m}$ is no longer directly updated by gradient descent algorithms during training, but indirectly obtained by updating the parameters $\boldsymbol{\Theta}_m$.

In the learning objectives defined in Eq. 2.4 and Eq. 2.5, the first term is a loss function dedicated to the given downstream learning task (e.g. a loss function for classification or regression) and $\lambda > 0$ is a weighting factor balancing the two terms. In should be additionally mentioned that the common regularization terms for training neural networks such as ℓ_2 weight decay [36] can be added to the entire learning objective to avoid overfitting, but such regularization is not directly related to the feature selection design and the resulting selection performance, so we omit it in the learning objectives above for simplicity. Finally, after training, we select the top-k features by comparing the learned importance scores in $\boldsymbol{m}$.

In order to provide a more straightforward understanding in this framework, below we briefly show a few exemplary DL-based FS approaches which follow the framework. For example, [69] followed the framework in Figure 2.1 and applied both ℓ_2 and ℓ_1 regularization on the feature mask during training and the downstream neural network $g(\cdot)$ was designed for classification or regression. The approach in [37] followed the structure presented in Figure 2.2 and $f_{\boldsymbol{\Theta}_m}(\cdot)$ was implemented with a large attention neural network. From these examples, we can see that the major research direction in DL-based feature selection is to design novel regularization terms on $\boldsymbol{m}$ and designing special generation process of the feature mask.

2.2.2 Comments on the DL-Based Feature Selection Framework

Having introduced the general idea of DL-based feature selection approaches, this section discusses the main reasons why DL-based FS research becomes increasingly prevalent and important. The first reason is that due to the fast advancements in hardware and software, deep neural networks can easily handle large-scale data with high dimensionality, which is more and more often in today's life. Conventional FS methods in contrast cannot deal with big data in an efficient way. Secondly, neural networks can cope with data with different types because neural networks can automatically learn suitable representations of input data, while some conventional approaches are designed

for specific data types (e.g. either categorical data or numerical data). Finally, as can be seen from the general framework and the examples above, DL-based approaches are highly modular, meaning that the downstream neural network can be easily modified according to specific user requirements. That is to say, $g(\cdot)$ can be designed for both regression and classification tasks, or for time series by being implemented using recurrent neural networks, or for images by being implemented using convolutional neural networks, or for unsupervised feature selection by implementing an autoencoder. Regardless of which downstream task is chosen, the core parts for FS (i.e. the regularization on $\boldsymbol{m}$ or the special general process $f_{\boldsymbol{\Theta}_m}(\cdot)$) are not required to change. Such easy-to-use property was not reachable by conventional FS approaches.

2.3 Datasets

In this dissertation, various benchmarking and real-world datasets are used to conduct intensive experiments. This section gives a brief introduction to the most important datasets for the subsequent chapters.

2.3.1 Industrial Datasets

The real-world Post-Silicon Validation (PSV) dataset is provided by Advantest, which is a leading manufacturer of automatic test equipment for semiconductor industry. A more detailed introduction and background information on semiconductor-related topics are along with Chapter 4 and we can omit these details now.

In particular, the real-world industrial dataset contains test cases (data samples) obtained by 9 different Devices Under Test (DUT). For each single DUT, we collected 100,000 test cases and there are therefore in total 900,000 test cases. Each test case can be understood as a feature vector with a corresponding target variable, where each feature vector has $d = 13$ candidate features. Moreover, the candidate features are of mixed data types including numerical (both continuous and discrete) and categorical features. Due to the confidentiality, we omit the actual physical meaning for individual candidate features and only use x_1 to x_{13} to represent them. Table 2.1 presents some exemplary

Table 2.1: Exemplary (anonymous) test cases of the real-world dataset.

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	y
$\boldsymbol{x}_1$	'foo'	'aa'	0.22	1.34	0.54	2.44	4	12	0	0.98	2	6	1	3.12
⋮						⋮								⋮
$\boldsymbol{x}_n$[1]	'fee'	'bb'	1.71	0.99	1.10	2.67	7	2	4	1.13	3	2	2	-1.8

Table 2.2: Statistics for the MNIST dataset.

	digit 0	digit 1	digit 2	digit 3	digit 4	digit 5	digit 6	digit 7	digit 8	digit 9
# Training	5923	6742	5958	6131	5842	5421	5918	6265	5851	5949
# Test	980	1135	1032	1010	982	892	958	1028	974	1009

data in an anonymous way that all quantitative values are artificial to maintain the confidentiality of the data for the company. Evidently, in this case, the first two candidate features x_1 and x_2 are categorical features, $x_7 \sim x_9$ and $x_{11} \sim x_{13}$ are numerical discrete features, while the rest features contain numerical continuous values.

In this task, as the target variable y is continuous, the downstream task is naturally a regression task. This means that we want to identify the most important features that can well predict y.

2.3.2 Benchmarking Datasets

MNIST The MNIST dataset [66] is an image dataset, containing 60000 training examples and 10000 test examples. Each example is a gray-scale digit image with 28×28 pixels. Exemplary images are shown in Figure 2.3 and detailed statistics are presented in Table 2.2. In the conducted experiments, each image was vectorized into a 784-dimensional feature vector, where each pixel was considered as a candidate feature. The downstream learning task is a 10-class classification problem.

[1] $n = 900,000$ for this real-world dataset.

Figure 2.3: Exemplary digit images of the MNIST dataset. Each row represents a specific digit. All digits are aligned and centered in this dataset.

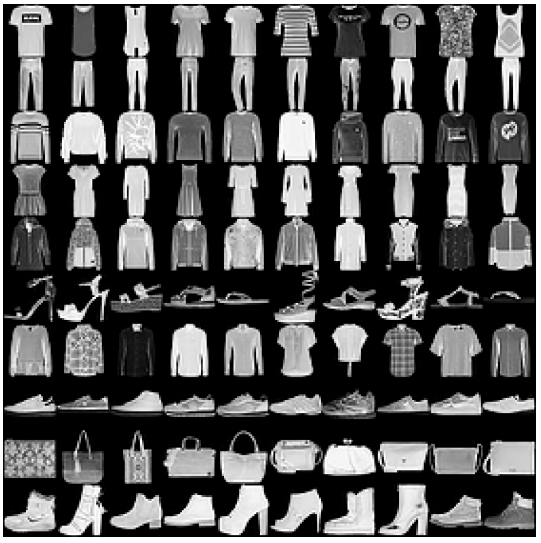


Figure 2.4: Exemplary digit images of the Fashion-MNIST dataset. Each row represents a specific article. All articles are aligned and centered in this dataset.

Fashion-MNIST The Fashion-MNIST dataset [126] contains gray-scale images of 28×28 pixels in the field of clothing, including T-shirt, trouser, pullover, dress, coat, sandal, shirt, sneaker, bag and ankle boot. There are 60000 training examples and 10000 test examples for each class. Likewise, each image was vectorized and each pixel corresponds to a candidate feature. This dataset is designed for a 10-class classification task. Some exemplary images are shown in Figure 2.4. The number of training and test samples in this dataset are 6000 and 1000 for each class.

Isolet The Isolated Letter Speech Recognition (Isolet) dataset [29] contains processed speech signals. In total, this dataset has 1560 feature vectors with 617 dimensions. In this dataset, all features are hand-crafted including but not limited to amplitude, spectral differences, duration and the number of consistent pitch peaks. A more detailed description for all features can be found in [29]. Each speech sample corresponds to a recording (speech signal) of a letter of the English alphabet. Therefore, this dataset is designed for a 26-class classification task. Table 2.3 shows three randomly selected exemplary training samples to provide an intuitive impression on this dataset.

Table 2.3: Exemplary sample of the Isolet dataset.

	x_1	x_2	x_3	...	x_{615}	x_{616}	x_{617}	Class label
$\boldsymbol{x}_1$	-0.7836	0.0366	-0.0092	...	0.0588	-0.4118	-0.5588	14
$\boldsymbol{x}_2$	-0.3834	0.1822	0.1244	...	0.1764	-0.0824	-0.7412	20
$\boldsymbol{x}_3$	0.0302	0.6010	0.8980	...	-0.1214	-0.1776	-0.7384	25

PCMAC The PCMAC dataset [65] is a text dataset with extracted features, containing 1943 feature vectors with 3289 dimensions for a binary classification problem. It is a challenging dataset because the number of candidate features is larger than that of the training samples. Although this is a standard benchmarking dataset for feature selection, almost no further information is available according to the resources provided by [67]. The data of this PCMAC are extremely sparse and it is difficult to show exemplary samples in a table. Therefore, we visualize the first 250 features of randomly selected 50 training samples as shown in Figure 2.5. Each row represents a training sample with

its first 250 features and each column represents a feature. As can be seen from the figure, deep blue parts are dominant, denoting values of zero, while light blue, white and red tiny squares denote the non-zero features of different training samples.

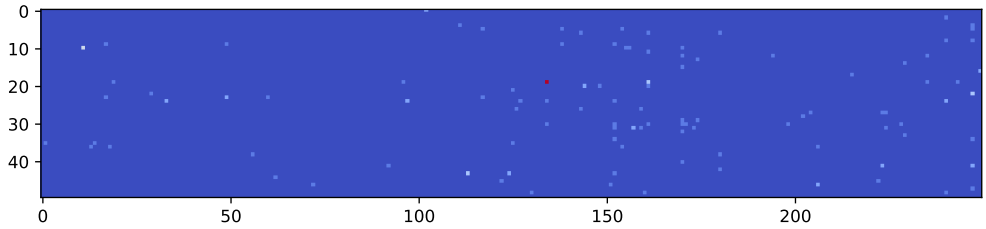


Figure 2.5: The first 250 features of the randomly selected 50 training samples from the PCMAC dataset. Each row represents a training sample, while each column denotes a candidate feature.

Madelon The Madelon dataset [41] is a binary classification dataset, containing 2600 samples with 500 candidate (integer) features. It is a synthetic dataset generated by a program in MATLAB. The details of setting the MATLAB program can be found in [42] and we omit them here because they are beyond the scope of this section. It is important to be aware that, in this dataset, only 5 features were used to determine the class of a data sample. In addition to them, another 15 redundant features were correspondingly created by a built-in function of the MATLAB program. Finally, 480 irrelevant features were added. Table 2.4 shows three randomly selected exemplary training samples to provide an intuitive impression on this dataset.

Table 2.4: Exemplary sample of the Madelon dataset.

	x_1	x_2	x_3	...	x_{498}	x_{499}	x_{500}	Class label
$\boldsymbol{x}_1$	484	491	572	...	481	558	470	0
$\boldsymbol{x}_2$	482	503	483	...	468	469	479	1
$\boldsymbol{x}_3$	491	478	448	...	474	560	463	0

Gisette The Gisette dataset [41] contains 7000 samples with 5000 dimensions. The primary goal is to discriminate digit 4 and digit 9 based on the MNIST dataset. The

authors [41] firstly normalized the data to the range of $[0, 1]$ and thresholded values below 0.5. Next, they created a feature set, consisting of the original normalized pixels and randomly selected subset of products of pairs of the pixels, where each member of the pair were the pixels that were normally distributed in a region of the image slightly biased upwards, i.e. the moderately upper region of the image. Then, all features that had only zero values were removed. Following that, all original pixels were used and complemented by the pairs to obtain 2500 features in total. Afterwards, another 2500 irrelevant features were created by randomly permuting the existing 2500 features across patterns. Finally, the order of the resulting features was randomized. Therefore, each feature vector finally contained 5000 candidate features. It should be additionally noted that this dataset is rather sparse, meaning only about 13% of the feature values are non-zero in average. Thereby, we visualize the first 250 features of randomly selected 50 training samples as shown in Figure 2.6. Each row represents a training sample with its first 250 features and each column represents a feature. In the figure, deep blue parts denotes of values of zero, while the tiny squares of other colors denote the non-zero features of different training samples.

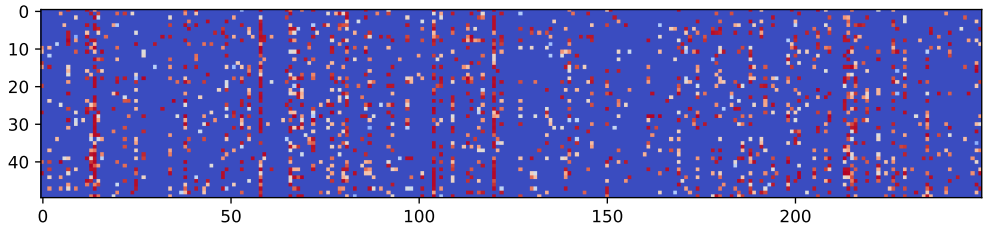


Figure 2.6: The first 250 features of the randomly selected 50 training samples from the Gisette dataset. Each row represents a training sample, while each column denotes a candidate feature.

Comments Indeed, whether it is optimal to consider each pixel in an image as a candidate feature for the MNIST and Fashion MNIST datasets, is still an open question, although many prior works [4, 37] did so. The difficulty is that feature selection generally requires that all feature vectors share the same important features, while images cannot guarantee that the same pixels are always meaningful and informative. Put simply, an object in an image can sometimes locate at the top left, while sometimes

at the bottom right. In this case, the most important pixels (features) should still be the features that characterize the target object (e.g. for the object recognition task). In this case, those features are highly dependent of individual images, and there might be no unique feature subset that can yield optimum performance for a given downstream task. Nevertheless, it is still acceptable to use the aforementioned three image datasets to benchmark feature selection approaches. First of all, it is fortunate that objects are perfectly aligned and located in the middle for all these three datasets so that the dilemma does not happen. Secondly, using image datasets is more intuitive for researchers and readers to have a visual understanding on the selected features (pixels).

Chapter 3

Feature Mask Approach

3.1 Introduction

This chapter describes the proposed Feature Mask (FM) approach, which aims to provide efficient feature selection while having limited and easy-to-tune hyperparameters. To achieve this goal, a novel FM-module is designed with three major components, namely a non-linear transformation block, a batch-wise average block[1] and a feature mask normalization block. Similar to other existing DL-based feature selection approaches, the FM-module is jointly trained with a downstream neural network with respect to a given learning task such as regression or classification. The general framework is illustrated in Figure 3.1. After training, the FM-module can generate a unique feature mask, of which each entry denotes the importance score of the corresponding candidate feature. Figure 3.2 shows several randomly selected digit images from the MNIST dataset to provide an intuitive impression on the selection performance of our FM approach, where the first row shows the original digit images, while the second row presents the selected features (pixels) based on the learned feature mask by our method. It is clear to see that the most critical pixels were successfully selected, because we can already recognize the digits by the selected pixels only. It should be noted that the

[‡]Contributions presented in this chapter were partially reported and published in our previous works [72, 75]. Some of the author's own formulations from [72, 75] are adopted in this chapter.

[1] In our published work [72], this block is called batch-wise attenuation block. In order to better reflect its functionality and to avoid confusion to the subsequent chapters, we uniformly use the term batch-wise average instead of batch-wise attenuation throughout this dissertation.

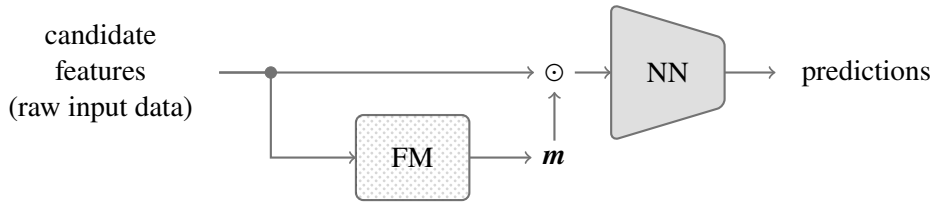


Figure 3.1: The Feature Mask method consists of an FM-module (*the dotted block*) and a neural network (NN in *the gray block*) dedicated to a given downstream learning task. Training data with all candidate features are used to train the entire model to predict the given target variables (e.g. class labels or regression targets). After training, the FM-module can calculate a unique feature mask $\boldsymbol{m}$ based on all training data.

Figure 3.2: Exemplary feature selection results on the MNIST dataset by using the novel FM-method. *top*: Raw digit images. *bottom*: The pixels selected by the proposed FM-method. It is compelling to see that the selected pixels by our approach already provide adequate information to visually discriminate different digits.

identical features are selected for all images in the MNIST dataset, instead of selecting different features for each individual input image. This is a crucial difference between feature selection and conventional sample-wise attention mechanism.

The next sections of this chapter are organized as follows. We firstly present the philosophy and detailed design of the novel FM approach in Section 3.2. Following that, intensive experimental results are shown and discussed in Section 3.3 as well as in Section 3.4, including evaluation on benchmarking and real-world datasets and in-depth investigation of the proposed method. Then, we demonstrate applying the FM-method to unsupervised feature selection with brief experimental results in Section 3.5. Finally, this chapter ends with a summary in Section 3.6.

3.2 Methodology

3.2.1 Structure and Learning Objective

The overall structure of the FM-method is illustrated in Figure 3.3. To have a better readability, in this dissertation, we use FM-module (or FM-block) to denote the function $f_{\text{FM}}(\cdot)$ only, i.e. the dotted block in Figure 3.3, while we use FM-method (or the FM approach) to denote the entire methodology.

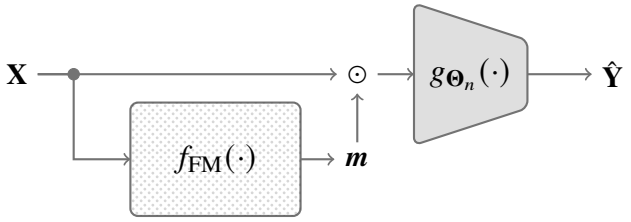


Figure 3.3: The structure of the proposed feature selection method. The FM-module $f_{\text{FM}}(\cdot)$ can generate a fixed feature mask $\boldsymbol{m}$ from the data $\mathbf{X}$. The learning network $g_{\boldsymbol{\Theta}_d}(\cdot)$ can be an arbitrary network parameterized by $\boldsymbol{\Theta}$ with respect to a given learning task.

Let the downstream neural network be denoted as $g_{\boldsymbol{\Theta}_n}(\cdot)$ parameterized by $\boldsymbol{\Theta}_n$ and the loss function for the downstream learning task is notated as $\mathcal{L}(\cdot,\cdot)$. The overall learning objective is therefore defined as

$$\min_{\boldsymbol{\Theta}_n,\boldsymbol{\Theta}_{\text{FM}}} \mathbb{E}_{(\mathbf{x},\mathrm{y})\sim p_{\text{data}}}\Big[\mathcal{L}\big(g(\boldsymbol{x}\odot\boldsymbol{m}),y)\big)\Big], \tag{3.1}$$

where $\mathbb{E}_{(\mathbf{x},\mathrm{y})\sim p_{\text{data}}}$ denotes the joint distribution of the data points and their labels. $\odot$ denotes the Hadamard product (i.e. element-wise multiplication) and the feature mask $\boldsymbol{m} = f_{\text{FM}}(\mathbf{X})$ parameterized by $\boldsymbol{\Theta}_{\text{FM}}$[2]. From the learning objective defined above, we can see that the proposed FM-method does not introduce other regularization terms or carefully designed loss functions in addition to an ordinary downstream learning objective (i.e. classification and regression). This property is specifically valuable and meaningful for practitioners facing new datasets in application, since

[2] $\boldsymbol{\Theta}_{\text{FM}}$ has the same meaning as that of $\boldsymbol{\Theta}_m$ defined in Section 2.2. We use the subscript of "FM" in order to emphasize that these trainable parameters characterize the FM-block only.

our method has notably less hyperparameters than many existing DL-based feature selection approaches.

Notoriously, modern neural networks are typically trained with mini-batch gradient descent algorithms for better efficiency, so $\mathbf{X}$ in Figure 3.3 indicates a minibatch during training. Accordingly, with $\mathbf{\Theta} = \{\mathbf{\Theta}_n, \mathbf{\Theta}_{\text{FM}}\}$, the training procedure of the FM-method is summarized in Algorithm 1. It should be noted that we show an ordinary mini-batch gradient descent algorithm for simplicity, but in practice we can use any modern variants of mini-batch gradient descent algorithms.

Algorithm 1 Learning Feature Mask

Require: Training dataset pair $(\boldsymbol{x}, \boldsymbol{y})$ in minibatch $(\mathbf{X}, \mathbf{Y})$, learning rate α

Randomly initialize the FM network (f_{FM} and g) with the initial parameters $\mathbf{\Theta}$
for e = 1 to maximal training epochs **do**
  for b = 1 to the number of minibatches **do**
    $\boldsymbol{m} \leftarrow f_{\text{FM}}(\mathbf{X})$ based on Eq. 3.4 to Eq. 3.6
    Calculate weighted input: $\mathbf{X} \odot \boldsymbol{m}$
    $\hat{\mathbf{Y}} \leftarrow g(\mathbf{X} \odot \boldsymbol{m})$
    Calculate training loss $\mathcal{L} \leftarrow \mathcal{L}(\hat{\mathbf{Y}}, \mathbf{Y})$
    Update network parameters $\mathbf{\Theta} \leftarrow \mathbf{\Theta} - \alpha \cdot \nabla_{\mathbf{\Theta}} \mathcal{L}$
  end for
end for

Obviously, the design of the FM-method is generic and can be used for both classification and regression tasks. In classification, a canonical choice of the loss function is the categorical cross entropy loss and the learning objective is reformulated as:

$$\min_{\mathbf{\Theta}_n, \mathbf{\Theta}_{\text{FM}}} \mathbb{E}_{(\mathbf{x},\mathrm{y})\sim p_{\text{data}}} \Big[- \boldsymbol{y}^T \log g(\boldsymbol{x} \odot \boldsymbol{m}) \Big]. \tag{3.2}$$

As for regression tasks, the Mean Squared Error (MSE) loss is more common and the learning objective becomes:

$$\min_{\mathbf{\Theta}_n, \mathbf{\Theta}_{\text{FM}}} \mathbb{E}_{(\mathbf{x},\mathrm{y})\sim p_{\text{data}}} ||g(\boldsymbol{x} \odot \boldsymbol{m}) - \boldsymbol{y}||_2^2. \tag{3.3}$$

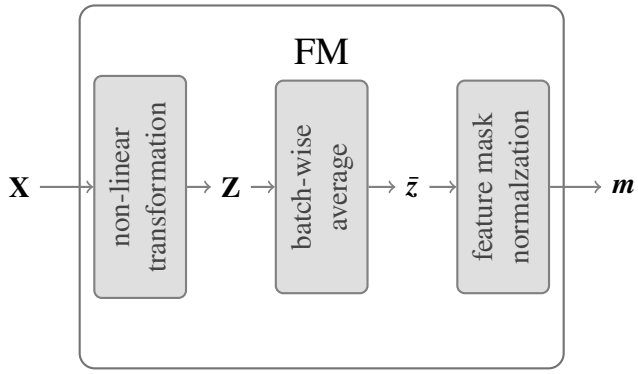


Figure 3.4: The structure of the FM-module. In each iteration, a minibatch $\mathbf{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_b]^T$ with b training samples is mapped to $\mathbf{Z}$ under a non-linear transformation. Subsequently, a primitive feature mask $\bar{\boldsymbol{z}}$ is obtained by averaging $\mathbf{Z} = [\boldsymbol{z}_1, \ldots, \boldsymbol{z}_b]^T$ over this minibatch. Finally, the feature mask $\boldsymbol{m}$ is calculated by applying feature mask normalization.

3.2.2 Feature Mask Module

The key idea is the novel Feature Mask module (FM-module) which is notated as a mapping $f_{\mathrm{FM}}(\cdot)$. The goal of the FM-module is to generate a fixed unique feature mask $\boldsymbol{m} \in \mathbb{R}^d_{\geq 0}$, namely a feature importance vector, calculated from the entire dataset. Accordingly, each element in $\boldsymbol{m}$ indicates a learned importance score of the corresponding candidate feature of the dataset. Broadly speaking, as presented in Figure 3.4, the FM-module is composed of three major blocks: *i*) non-linear transformation; *ii*) batch-wise average; *iii*) feature mask normalization.

Non-Linear Transformation

The goal of the FM-module is to directly learn a feature mask (feature importance vector) from the input data. Thereby, in this work, two fully connected layers are used to construct a non-linear transformation between the input data and a feature mask as

$$\boldsymbol{z} = \mathbf{W}_2 \cdot \phi(\mathbf{W}_1 \cdot \boldsymbol{x} + \boldsymbol{b}_1) + \boldsymbol{b}_2 \,, \tag{3.4}$$

where $\mathbf{W}_1 \in \mathbb{R}^{d' \times d}$, $\mathbf{W}_2 \in \mathbb{R}^{d \times d'}$, $\boldsymbol{b}_1 \in \mathbb{R}^{d'}$ and $\boldsymbol{b}_2 \in \mathbb{R}^{d}$ with $d' < d$. $\phi(\cdot)$ is a user-specified nonlinear function such as $\tanh(\cdot)$.

We argue that the non-linear transformation is necessary for three reasons. Firstly, we cannot in advance know how complex the underlying mapping is and an assumption of non-linear transformation can better cover more general cases. Secondly, according to [37], using a bottleneck structure in this non-linear transformation is expected to be able to capture complex relations between candidate features and remove potentially redundant candidate features. Finally, such design also inherits the philosophy of attention mechanism as in [47, 125], where the most critical parts of the input data can be automatically focused.

It should be additionally noted that the non-linear transformation shown above can be understood as one possible implementation. In practice, we can stack more layers, or use other layer types such as convolutional layer. A concrete choice is dependent on a specific use case and prior knowledge on the data. Nevertheless, our experiments in subsequent sections show that the current implementation is already able to deal with a variety of different datasets and we therefore recommend other researchers and practitioners to follow such design.

Batch-Wise Average

One important requirement in feature selection is that all samples share identical important and representative features. However, given a training minibatch $\mathbf{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_b]^T$ with a minibatch size of b, the output of the non-linear transformation is another matrix $\mathbf{Z} = [z_1, \ldots, z_b]^T$, where each row vector $\boldsymbol{z}_i^T$ is calculated from the corresponding input data point $\boldsymbol{x}_i^T$. In other words, z_i can differ from z_j if $i \neq j$, which contradicts the requirement in feature selection. To cope with this problem, it is natural to propose a batch-wise average defined as

$$\bar{z} = \frac{1}{b} \sum_{i=1}^{b} z_i \tag{3.5}$$

by explicitly averaging $\mathbf{Z}$ along its minibatch axis. In this case, we can understand $\bar{z}$ as a primitive feature mask (i.e. before normalization) that ensures a final unique feature importance vector for the entire data. Obviously, the impact of the individual data point is attenuated by this averaging process so that more reliable primitive feature mask can be learned during training.

Feature Mask Normalization

A normalization is designed for the primitive feature mask for two reasons. One reason is that bounded feature importance scores are desired after training to have a better interpretability, as negative importance scores are difficult to understand and explain. The other reason is that the relative importance of candidate features should be taken into account during training. In other words, it is expected that the importance of certain candidate features should decrease, if some other candidate features become more important. Based on the concerns above and inspired by modern attention mechanism [47, 118, 125], we propose to use a $softmax(\cdot)$ function to normalize the primitive feature mask as

$$\boldsymbol{m} = softmax(\bar{\boldsymbol{z}}), \text{ with } m_i = \frac{e^{\bar{z}_i}}{\sum_{j=1}^{d} e^{\bar{z}_j}}. \tag{3.6}$$

In this way, the importance scores of all candidate features are bounded in the range of $(0, 1)$. Furthermore, the sum of all importance scores is exactly one, meaning that some features would be less important, if certain features are assigned with larger scores during training.

Comments

The design of the FM-module introduced above has a neat property that all three blocks are differentiable. This enables a joint training of the FM-module and a downstream neural network with an arbitrary architecture. Thereby, guided by the downstream task as an embedded feature selection approach, the FM-method gradually learns to assign larger weights to the more critical candidate features, while down-weighting

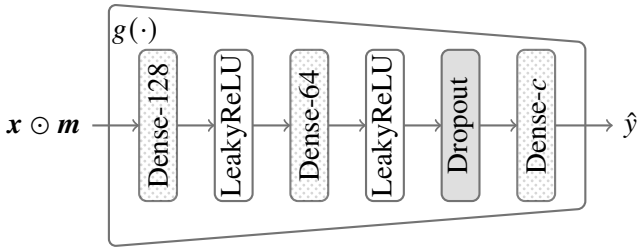


Figure 3.5: One possible implementation of the neural network $g(\cdot)$ used for the proposed FM-method in the conducted experiments.

less meaningful or irrelevant candidate features. After training, the final unique feature mask $\boldsymbol{m}$ is separately calculated over the entire training data using the trained FM-module. More concretely, the same calculations in Eq. 3.4, Eq. 3.5 and Eq. 3.6 are done on the whole training dataset, where the average calculation in Eq. 3.5 is done over all training samples and not over one minibatch. Then, for the selection, we can easily identify the k most important candidate features by finding out the top-k largest entries in $\boldsymbol{m}$.

3.3 Evaluations

This section reports the experimental results of our novel FM-method. We conducted extensive experiments on synthetic, benchmarking and real-world datasets to achieve a comprehensive understanding regarding the advantages as well as the limitations of our approach. In addition, several contemporary reference DL-based feature selection approaches were compared to justify the superiority of our approach.

3.3.1 Setup

Settings Both the reference methods and our method can be categorized to embedded feature selection and do not require special architecture for the downstream neural network $g(\cdot)$ so we used the same architecture $g(\cdot)$ for a fair comparison as shown in Figure 3.5. Specifically, the downstream neural network was composed of two fully

Table 3.1: Overview of the benchmarking datasets.

	# Features	# Train	# Test	# Classes	Type
MNIST	784	60000	10000	10	Image
fMNIST	784	60000	10000	10	Image
Isolet	617	1248	312	26	Speech
PCMAC	3289	1555	388	2	Text
Madelon	500	2080	520	2	Artificial
Gisette	5000	5600	1400	2	Extracted

connected layers with 128 and 64 neurons, respectively. LeakyReLU [84] with the rate of 0.2 was used as an activation function for each fully connected layer. A dropout layer [112] with the rate of 0.3 was added before the output layer in order to avoid overfitting during training. d' introduced in Eq. 3.4 was set to 128 for all cases as it generally yielded satisfied performance for all considered datasets. We repeated each individual experiment for five times with different random seeds in order to remove the randomness within the neural network initialization. We report the averaged results with their standard deviations (where applicable).

Metrics In the conducted experiments, the ground truth about the relevant features is available for both synthetic and real-world datasets so we only report whether different approaches successfully selected those relevant features. On the contrary, benchmarking datasets do not have ground truth of the relevant features. Therefore, to be consistent with prior works, we used accuracy obtained by a separately trained classifier to quantify the quality of the selected features. The accuracy is defined as

$$\text{accuracy} = \frac{\text{\# correctly classified samples}}{\text{\# all samples}}. \tag{3.7}$$

Benchmarking Datasets Six well-known benchmarking datasets were used in this chapter, including two image datasets, namely MNIST [66] and Fashion-MNIST (fMNIST) [126], one speech dataset Isolet [29], one text dataset PCMAC [65], one artificial

dataset Madelon [41] and one dataset with extracted features called Gisette [41]. The benchmarking datasets used in this dissertation can well represent most common dataset types in the real world. Table 3.1 contains more details about the datasets.

Reference Methods This chapter mainly compares the proposed FM-method with three contemporary reference DL-based feature selection approaches as follows: *i*) Attention-Based Feature Selection (AFS) [37]; *ii*) CancelOut [14]; *iii*) Deep Feature Selection (DFS) [69]. The most crucial hyperparameters of all reference methods mainly followed the original papers or individually optimized by us for the considered datasets using grid search on a separate validation set, corresponding to about 10% of the training data. In addition, in our experiments, all raw features (RawF) were used as a performance upper bound[3]. On the other hand, randomly selected features (RSF) serve as a performance lower bound because a well designed FS method must notably outperform the RSF.

Downstream Classifiers This chapter includes three popular algorithms as the downstream classifiers for evaluation: *i*) Random Forest (RF) [16]; *ii*) Extremely Randomized Tree (ERT) [32]; *iii*) k-Nearest Neighbor (kNN) [39]. We use several downstream classifiers different from neural networks because we want to justify the quality of the selected features for more various and general scenarios. The three classification algorithms above were implemented using scikit-learn [93] and TensorFlow [1]. We followed the default or commonly used settings for all three classifiers in all experiments for a fair comparison.

3.3.2 Main Results

Performance on Benchmarking Datasets

The primary experimental results are presented in Table 3.2, Table 3.3 and Table 3.4, where we evaluated the top 2.5% features selected by our method and the reference

[3]Typically, we can roughly consider that using all features works better than using a subset of those features. Although some noisy features can indeed affect downstream learning performance for specific algorithms, we reduce such concerns by comparing many different downstream classifiers in the subsequent experiments.

methods on the three previously mentioned downstream classifiers. It is clear to see that our FM-method performed better than other contemporary methods in 15 out of 18 cases, indicating that our FM-method can generally perform well on different data types. It is additionally noticeable that the selected features by our method resulted in more consistent classification accuracy in the downstream learning tasks over different classifiers, which is valuable for practical usage.

Broadly speaking, our FM-method achieved significantly better performance than the other reference methods for most cases. While DFS reached slightly better accuracy on the Gisette dataset than ours, our FM-method has significantly fewer hyperparameters than DFS. Specifically, DFS requires to use ℓ_1- and ℓ_2-regularization in the loss terms for both feature selection and the downstream learning, corresponding to at least 4 additional hyperparameters. On the contrary, our method does not change the original learning objective and does not introduce new loss terms, so it is much easier to use the FM-method in practice.

Our experiments also included RawF and RSF as the upper and lower bounds to assess the quality of the selected features by different methods, respectively. We argue that a carefully designed FS algorithm must yield significantly better performance than randomly selected features (RSF). Although many previous studies often neglected the experiments with RSF, we surprisingly found that RSF led to better results than some carefully designed methods in some cases. As an example, the method CancelOut has poorer performance than RSF on the datasets Madelon and Gisette. One feasible reason can be that CancelOut was sensitive to its hyperparameters and network initialization. In contrast, as expected, the FM-method outperforms the RSF on all considered datasets with a significant margin.

2D Visualization In addition to justifying the selection quality by comparing the classification accuracy on different downstream classifiers, we visually judged whether the selected features of our methods and the reference methods can well preserve the original data structure by comparing them with raw features. To provide visual evidence, we used UMAP [85] to respectively map the selected features and raw features into two dimensional space for easy visualization. Specifically, we report the results

Table 3.2: Downstream classification accuracy based on the top selected features based on RF (2.5% of all candidate features).

	MNIST	fMNIST	Isolet	PCMAC	Madelon	Gisette
RSF	0.537	0.766	0.551	0.577	0.496	0.926
RawF	0.970	0.876	0.955	0.948	0.727	0.970
AFS	0.816	0.771	0.579	0.857	0.550	0.936
CancelOut	0.813	0.776	0.551	0.878	0.485	0.911
DFS	0.840	0.787	0.647	0.861	0.758	**0.972**
FM	**0.851**	**0.798**	**0.804**	**0.892**	**0.876**	0.946

Table 3.3: Downstream classification accuracy based on the top selected features based on ERT (2.5% of all candidate features).

	MNIST	fMNIST	Isolet	PCMAC	Madelon	Gisette
RSF	0.529	0.766	0.494	0.572	0.465	0.927
RawF	0.972	0.878	0.949	0.969	0.696	0.974
AFS	0.810	0.769	0.541	0.862	0.568	0.939
CancelOut	0.807	0.774	0.571	0.880	0.486	0.908
DFS	0.834	0.784	0.673	0.864	0.769	**0.973**
FM	**0.845**	**0.792**	**0.810**	**0.900**	**0.869**	0.949

Table 3.4: Downstream classification accuracy based on the top selected features based on kNN (2.5% of all candidate features).

	MNIST	fMNIST	Isolet	PCMAC	Madelon	Gisette
RSF	0.479	0.711	0.442	0.613	0.485	0.853
RawF	0.969	0.858	0.904	0.660	0.546	0.959
AFS	0.749	0.727	0.470	0.766	0.533	0.904
CancelOut	0.748	0.734	0.509	0.771	0.503	0.885
DFS	0.778	0.757	0.615	0.799	0.691	**0.963**
FM	**0.796**	**0.762**	**0.760**	**0.819**	**0.829**	0.922

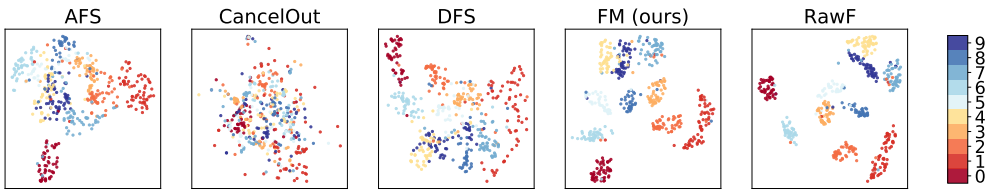


Figure 3.6: 2D visualization of the MNIST dataset using the top-50 features (pixels). The selected features by the proposed FM-module well retain the data structure and perform similarly to the raw features.

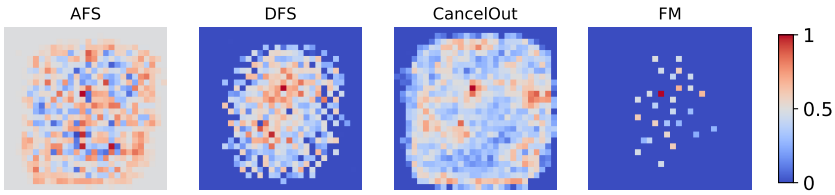


Figure 3.7: The visualization of the learned feature importance scores for the reference methods and the FM-method on the MNIST dataset.

on the MNIST dataset and we selected 50 features (i.e. 6.4% of all candidate features) as an example.

Figure 3.6 illustrates the corresponding 2D visualization results, where each color denotes a specific class (digit). Without surprise, the features selected by our method well preserve the original structure. Firstly, the images affiliated with the same class (digit) were well clustered, meaning that the selected 50 features already provide enough and meaningful information for effective discrimination between different classes. Moreover, the inter-class relationship was well maintained. For example, the raw features show that digit 4, 7 and 9 are spatially close to each other, which can be observed by our case as well. In contrast, some reference methods such as CancelOut did not well preserve the original data structure, as it resulted in a mixture in the 2D space without discernible clusters for the different classes (digits).

Feature Masks Visualization As shown in Figure 3.7, we visualized the learned feature importance vectors of our method as well as all reference methods on the

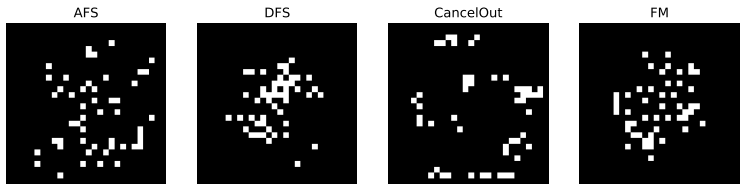
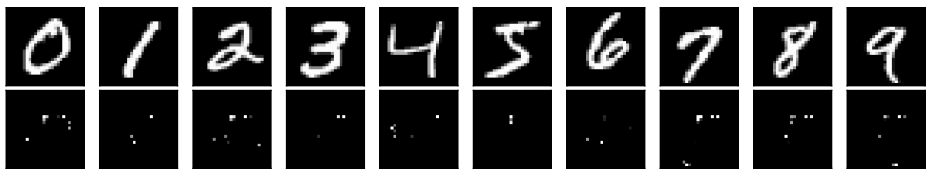


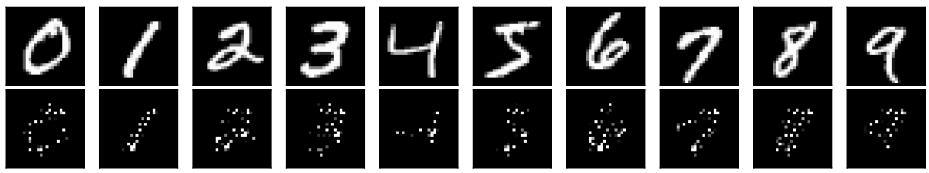
Figure 3.8: The selected top-50 features (*white pixels*) based on the learned feature importance vectors in a supervised setup on the MNIST dataset.

(a) The selected features by the AFS method.

(b) The selected features by the DFS method.



(c) The selected features by the CancelOut method.



(d) The selected features by the FM-method.

Figure 3.9: Comparison of the selected top-50 features.

MNIST dataset. As the learned feature importance vectors of different approaches are at different scale, they were normalized into the range between 0 and 1 by

$$m_{i,\text{normalized}} = \frac{m_i - \min(\boldsymbol{m})}{\max(\boldsymbol{m}) - \min(\boldsymbol{m})}, \tag{3.8}$$

where $\min(\cdot)$ and $\max(\cdot)$ takes a vector as input and gives the minimal and maximal element of the vector, respectively. Afterwards, the feature importance vectors were reshaped into 28×28 for the MNIST dataset for better visualization and a clearer comparison. The red and light-color pixels (corresponding to the features with larger importance scores) show that the FM-module successfully selected the most informative features, i.e. the pixels located in the center of digit images. This aligns the nature of the MNIST dataset, in which each digit is aligned to the center. In addition to our FM-method, DFS selected the centered pixels, but it unfortunately assigned similar importance scores to the neighboring pixels. This indicates that the redundant features (pixels) were mistakenly selected, while they do not bring any additional information for the downstream learning tasks.

Conversely, the other methods CancelOut and AFS tended to assign high importance scores to the pixels around the edges of an image. However, for this MNIST dataset, the pixels around the edges have values of zero and do not provide any information for correct classification and should be thus removed by feature selection. That is to say, irrelevant features were mistakenly selected by CancelOut and AFS. Figure 3.8 highlights the top-50 features in white based on the learned feature mask in Figure 3.7 to demonstrate the selection quality, which further confirms the observations above.

Figure 3.9 shows some exemplary selection results of our method as well as the reference methods. In each subfigure, the first row shows the raw digit images, while the second row shows the product of the raw images and the learned binary feature mask (shown in Figure 3.8). It is evident that the features selected by our FM-method well represent the structure of different digits, allowing humans to easily identify digits using only 50 pixels. Although the features selected by DFS also shows the digit structure to a certain degree, it is notably more difficult to recognize. Obviously, some methods such as CancelOut or AFS result in many black pixels, meaning that they mistakenly selected background pixels as the top-50 features, which are irrelevant to classification.

3.3.3 Case Study: Synthetic Datasets

As introduced in the previous chapter, in feature selection, we typically do not have access to the real ground truth of the optimal feature subsets, so we often use the performance of the selected features on downstream tasks to indirectly justify the selection quality. This section aims to use a synthetic dataset to validate our method and compare it with other reference approaches so that the selection results can be reliably justified with ground truth.

Synthetic Data

The synthetic dataset consists of 5000 training examples with 20 candidate features from x_1 to x_{20}. The target variable y is designed by

$$y = x_1^2 + 10x_2x_3x_4 + 5x_5x_6 + \epsilon, \tag{3.9}$$

where $\epsilon \sim N(0, 1)$ and $x_i \sim U(0, 1)$ and all of them are statistically independent. In this setup, according to Definition 1 and Definition 2, the first 6 candidate features are relevant to the target y, while the remaining 14 candidate features are irrelevant.

Results

Figure 3.10 presents the learned feature masks (importance scores) of our method and the reference methods. On this simple synthetic dataset, all methods successfully assigned large scores to the first six candidate features as expected. It should be specifically noted that only our approach assigned extremely small scores to the irrelevant candidate features, automatically leading to an approximate sparsity in the feature mask, while the other three methods still assign moderate importance scores to irrelevant features, although they stated that their design can result in sparsity. Evidently, the learned scores of different approaches are of different scale. For example, DFS does not apply any constraints on the upper bound of the scores and some of them were consequently larger than one. In order to have a clearer comparison,

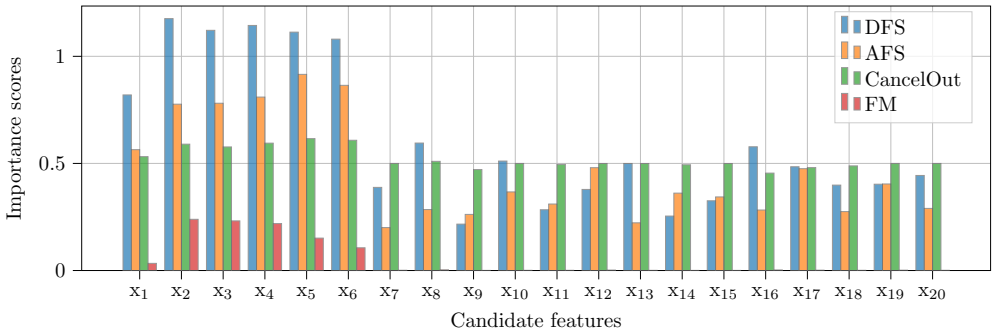


Figure 3.10: Comparison of the learned feature importance scores of different feature selection approaches on the synthetic dataset. It is clear to see that the proposed FM-method successfully assigned extremely small values to the irrelevant candidate features (x_7 to x_{20}), while the other methods undesirably assigned large scores to them.

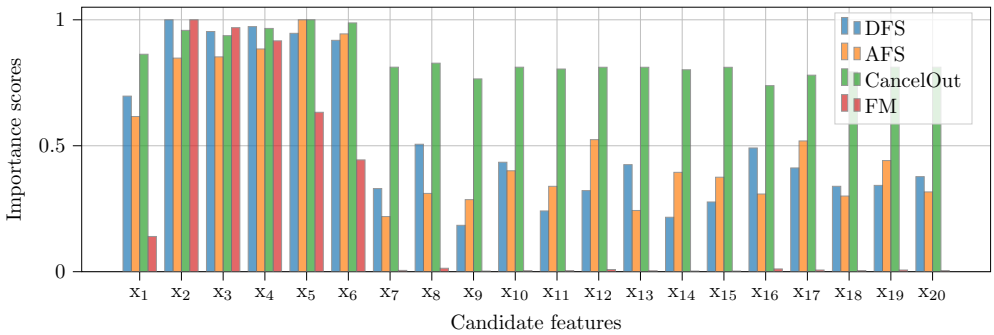


Figure 3.11: Comparison of the learned feature importance scores of different feature selection approaches on the synthetic dataset (normalized for better visualization). Obviously, the importance scores learned by our FM-method well reflected the relative importance (predictive power with respect to the target y) of individual candidate features.

we rescale the learned scores to the range between 0 and 1 as shown in Figure 3.11. We can easily see that our approach still maintains the approximate sparsity as before. In contrast, CancelOut showed almost indistinguishable scores after rescaling. An important observation is that the importance scores of our FM-method also reveals the relative contributions of the relevant candidate features. For example, x_2, x_3 and x_4 had the largest impact of the target y and they were indeed assigned with the largest scores. Likewise, x_5 and x_6 had similar scores and ranked after the top three. x_1 was relevant but had the least impact on the target, so a smaller importance score was correctly assigned to x_1. On the contrary, although the other three methods could identify all relevant candidate features, they failed to show the relative importance among the relevant features. This property of the FM-method is valuable in practice, if users want to use the learned importance scores to analyze the impact of individual features with respect to the targets.

3.3.4 Case Study: The Post-Silicon Validation Dataset

The proposed FM-method was evaluated with the real-world dataset which is introduced in Chapter 2. Specifically, we mainly focus on the test cases (samples) obtained from a single DUT. This means that we have 100,000 training samples with 11 candidate features. Since the number of the candidate features is small, we use $d' = 8$ for the FM-module in this experiment with an extremely large batch size of 25000 to take full advantage of GPUs.

Results

We compared our FM-method with an exhaustive search approach. In this way, the ground truth of the optimal feature subsets for the real-world dataset can be obtained by the exhaustive search. In particular, the exhaustive search evaluated all feature combinations by training a separate neural network on them to predict the target variable. The neural network had the same architecture as the $g(\cdot)$ network in our FM-method for a fair comparison. This means that the exhaustive method trained $\sum_{i=1}^{11} \binom{11}{i} = 2047$ models. According to the experimental results, our FM-method

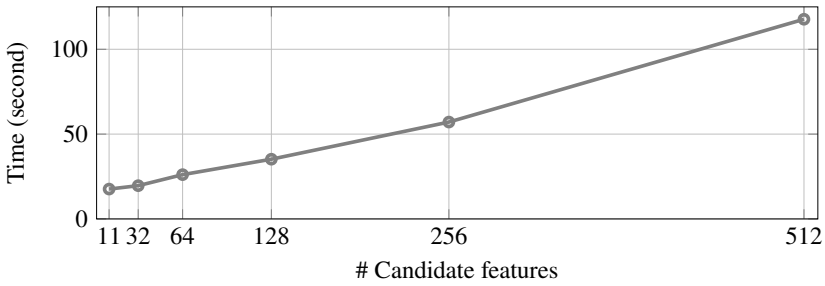


Figure 3.12: Time consumption versus the number of candidate features on the real-world PSV dataset.

successfully selected the relevant candidate features and provided a correct importance order of those features (the ground truth for the importance: $x_6 > x_7 > x_9 > x_5 > x_8 > x_{10}$). However, our FM-method only used about 0.067% time consumption compared to the exhaustive search.

Scalability In practice, the scalability plays a critical role to save time and money. Therefore, we empirically justify the scalability of our FM-method with respect to the number of candidate features and the number of training instances. Figure 3.12 illustrates the time consumption (in second) to train our FM-method versus the total number of input candidate features[4]. The computational complexity is independent of the number of the desired selected features because the FM-method determines one importance score vector for all input features. It is clear to see that our method shows a nearly linear behavior in this case. In contrast, many conventional approaches can have more than exponential complexity in terms of the number of candidate features. The main reason here is that the number of candidate features mainly affects the size of the first layer within the FM-module as well as the first hidden layer in the downstream neural network. Nevertheless, the majority of the network parameters maintain the same with increasing features. In other word, due to the nature of DL-based solutions, we can achieve a linear complexity.

In terms of the number of training samples, as shown in Figure 3.13, our method also shows a linear behavior due to the similar reason mentioned above. This is also

[4]The experiment was done on an NVIDIA RTX 2060 Super GPU at 1470 MHz.

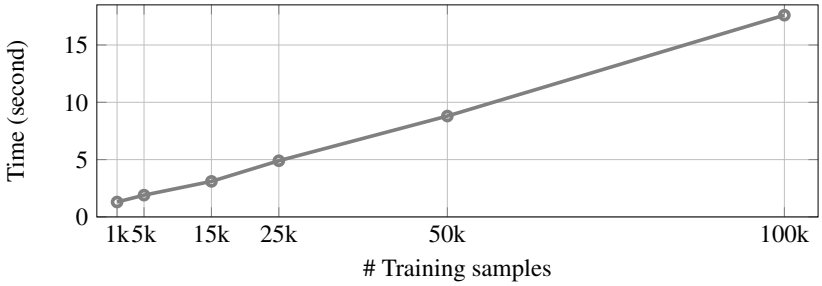


Figure 3.13: Time consumption versus the number of training samples on the real-world PSV dataset.

meaningful and valuable for practitioners because they can easily estimate potential time consumption for different dataset sizes.

More importantly, according to our experiments, the novel FM-method already correctly selected the most important candidate features by using only 500 training instances of the PSV dataset as shown in Figure 3.14. This result empirically shows the effectiveness of our approach on small-scale data for this real-world PSV use case.

3.3.5 Robustness of Selection

As introduced in Chapter 2, estimating the optimal subset feature subset sizes is also a task for feature selection. Therefore, after obtaining the importance scores by DL-based feature selection approaches, it is important to evaluate the feature subsets of different sizes. The motivation behind is that in practice we typically do not know the exact size of an optimal subset size, or we want to have a more comprehensive understanding of the trade-off between the subset sizes and the corresponding predictive power. Figure 3.15 presents the accuracy with different subset sizes with respect to the three downstream classifiers. Specifically, we chose seven different feature subset sizes, corresponding to 1%, 1.5%, 2%, 2.5%, 5%, 7.5% and 10% of all candidate features.

Generally speaking, the FM-method outperformed the other reference methods for most cases with respect to different selected feature subset sizes, suggesting the superiority

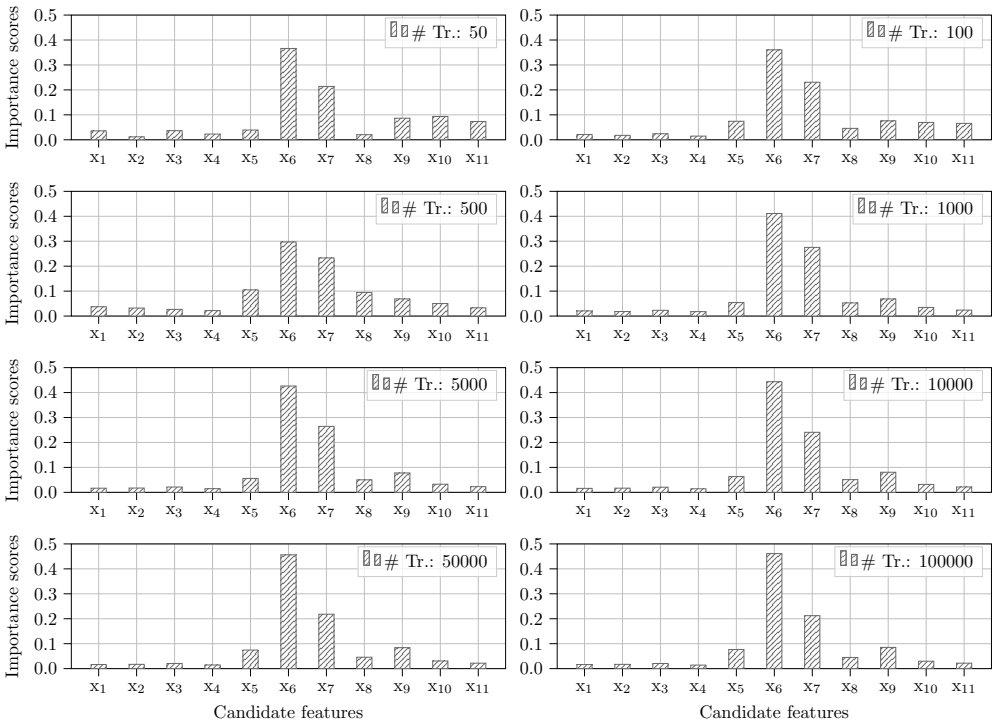
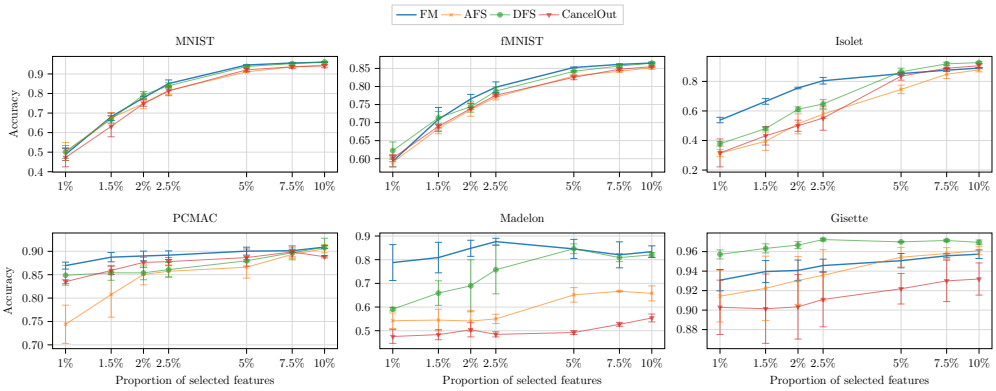
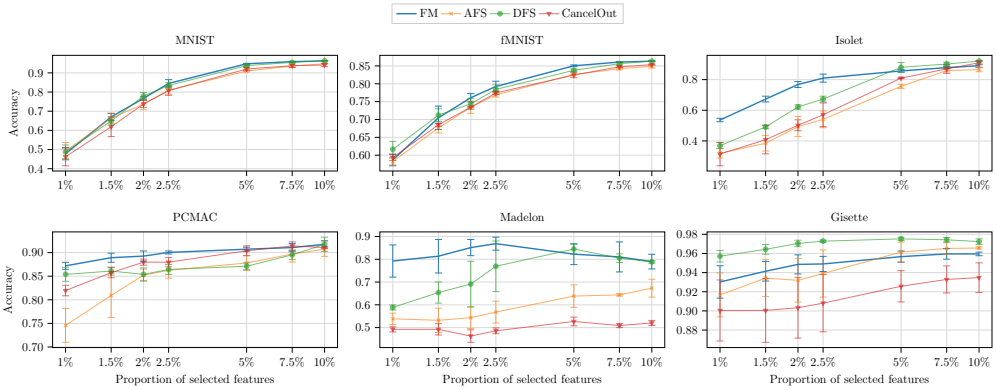


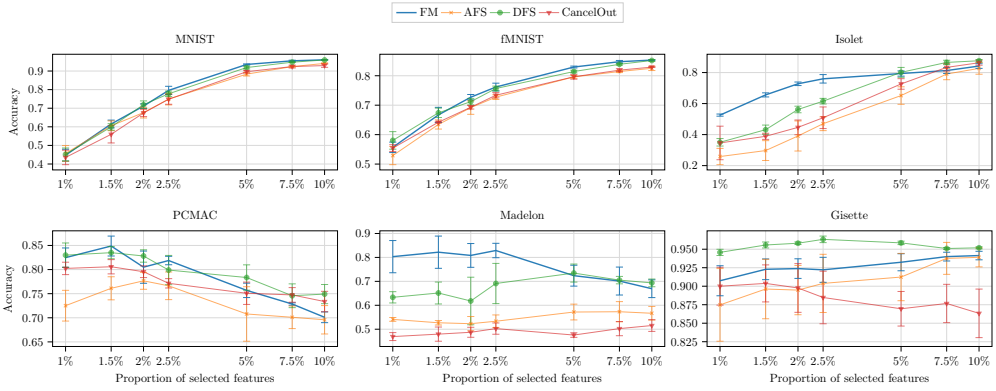
Figure 3.14: The learned feature masks using different training dataset sizes (notated as # Tr.). Using 100 training samples only, our FM-method can already correctly identify the relevant candidate features. Using 1000 training samples only, our FM-method can provide correct importance order of all relevant candidate features.



(a) Classification accuracy on RF over different selected feature subset sizes.



(b) Classification accuracy on ERT over different selected feature subset sizes.



(c) Classification accuracy on kNN over different selected feature subset sizes.

Figure 3.15: Downstream accuracy versus different feature subset sizes.

of our method. In particular, on the datasets Isolet, PCMAC and Madelon, our FM-method achieved significant better selection quality for different subset sizes than the reference methods, while our method had the least hyperparameters than the others. Unfortunately, our FM-method was ranked second in terms of the selection quality on the Gisette dataset. Furthermore, we can observe that our FM-method often led to the smallest deviation in terms of the resulting accuracy, presenting the stability of the selected features and robustness over different classifiers.

It is worth noting that the performance of the DL-based feature selection approaches can be affected by various factors, such as the complexity of the dataset, the quality of the features, and the choice of the downstream classifiers. In limited cases, certain reference methods may perform better than the proposed FM-method. However, as demonstrated in this experiment, our FM-method generally outperformed the reference methods in terms of selection quality and stability over different classifiers on a wide range of various datasets.

These results suggest the potential of the FM-method for practical applications, where the accurate and efficient feature selection is essential for improving the performance of machine learning models.

3.4 Insights of the Feature Mask Approach

The experimental results in the previous sections have demonstrated the effectiveness and superiority of the proposed FM-method for feature selection. Despite its success in both public benchmarking datasets and real-world datasets, this section further dives into the design of the FM-method. In particular, this chapter aims to answer the following questions: *i*) Whether both the batch-wise average and the feature mask normalization blocks contribute to the selection performance? *ii*) Can our FM-method generally reject redundant features during training? *iii*) Does our FM-method have comparable computational complexity as the reference methods? *iv*) Is our FM-method sensitive to the limited hyperparameters within the structure?

Table 3.5: Ablation study of the FM-method (exemplary classification accuracy on MNIST using a random forest classifier).

	w/o FMN	**w/ FMN**	Gain
w/o BA	0.720 (± 0.005)	0.932 (± 0.005)	29.4% ↑
w/ BA	0.898 (± 0.008)	0.954 (± 0.003)	6.2% ↑
Gain	24.7% ↑	2.4% ↑	

3.4.1 Ablation Study

The ablation study was designed in order to investigate the individual contribution brought by the Batch-wise Average (BA) and Feature Mask Normalization (FMN) within the FM-module, respectively.

Table 3.5 summarizes the resulting accuracy on a random forest as the downstream classifier on the MNIST dataset as an example. It is evident from the table that both the BA and the FMN blocks contribute to the final feature selection quality, i.e. leading to higher accuracy on the random forest. More precisely, we can observe that using the FMN block alone had more significant contribution to the accuracy than the case where the BA block was additionally applied. This indicates that in our design, the FMN block might be slightly more critical than the use of BA alone. We suggest that there are at least two advantages of introducing feature mask normalization to feature selection. Firstly, FMN uses the $softmax(\cdot)$ function to guarantee non-negativity (actually a bound) of the resulting feature mask, leading to more reasonable weighted inputs $\mathbf{X} \odot \boldsymbol{m}$ that were fed to the downstream neural network. The other advantage is, in our opinion, that the relative importance of different candidate features were considered due to the denominator of $softmax(\cdot)$, which has been, however, rarely explored in literature.

Moreover, in order to provide a more intuitive understanding of the individual effect brought by BA and FMN, the ablation study was conducted on the synthetic dataset as well. Figure 3.16 compares the learned feature masks of the three cases, i.e. without BA, without FMN and with both blocks (i.e. the proposed FM-method). We can

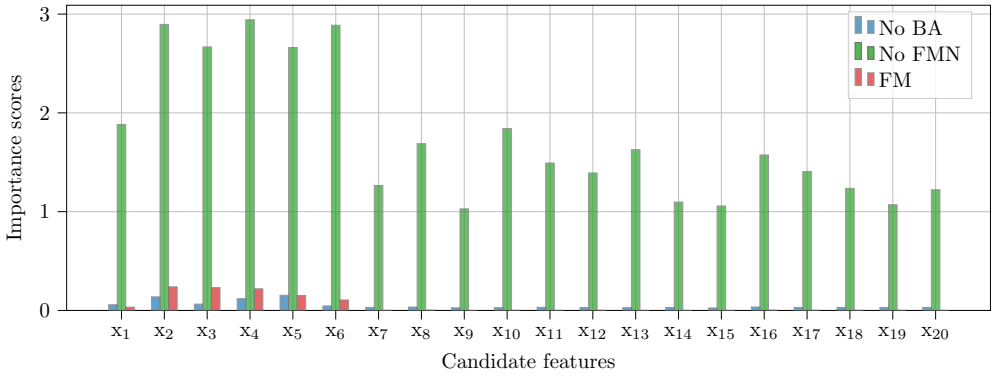


Figure 3.16: The learned feature masks for the ablation study of the FM-method on a synthetic dataset. It is easy to see that unbounded scores were learned without using the FMN-block and a few irrelevant candidate features were incorrectly assigned with large scores, which are almost indistinguishable to the relevant feature x_1.

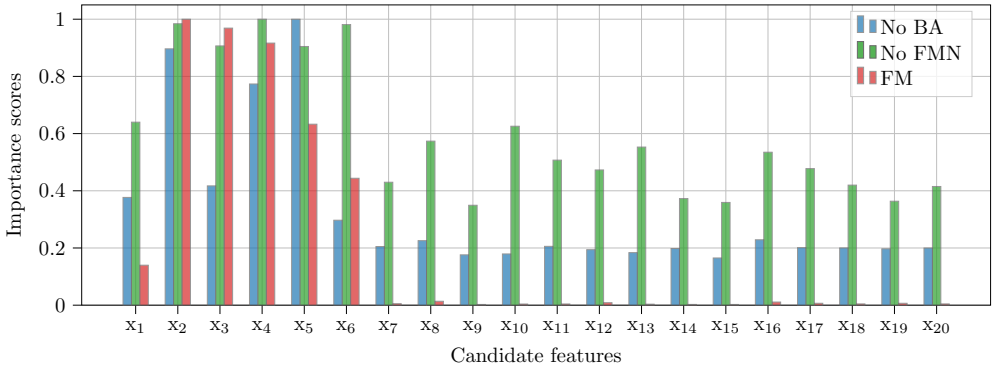


Figure 3.17: The learned feature masks for the ablation study of the FM-method on a synthetic dataset. The feature masks were rescaled for better visualization. Obviously, the BA-block contributes to correct importance score orders for the relevant candidate features.

observe that the baseline without FMN resulted in unbounded importance scores[5] and a few irrelevant features unfortunately had similar importance scores as that of x_1, which was undesirable and did not match the ground truth. In addition, we can see that the sparsity vanished if no FMN was applied. Figure 3.17 shows the rescaled feature importance scores for better visualization. It is clear to see that BA mainly contributed to correct importance score orders for the relevant candidate features, which is valuable for identifying the most important feature subset of different sizes. In total, the experimental results on the synthetic dataset clearly demonstrate the necessity of both FMN and BA blocks for the final selection quality.

3.4.2 Resistance Against Redundant Features

Chapter 2 introduced that selecting features with the exposure of redundancy is a challenge for many FS algorithms. Although the property of resisting redundant features is not specifically required for the FM-method, the experimental results on the synthetic dataset surprisingly show that the FM-method can automatically remove redundant features during training.

Figure 3.18 compares the learned importance scores of different FS methods with the exposure of redundancy, where we let $x_7 = x_6$, meaning that the seventh feature was exactly the same as the sixth feature (redundant to each other). In this case, we can see that the FM-method finally assigned large scores only to x_7, while x_6 had almost a zero importance score. In contrast, the other reference methods assigned large scores to both x_6 and x_7, having no capability of removing redundant features for selection. This property of the FM-method is especially valuable if small subset sizes are desired in practice.

Furthermore, considering that the learned importance scores by our FM-method can reflect the actual relative importance among the relevant candidate features, we compared redundant features with different noise level. In particular, we let $x_7 = x_6 + \alpha \cdot \epsilon$, where $\epsilon \sim N(0, 1)$ with $\alpha \in \{0, 0.05, 0.5\}$. Figure 3.19 presents the learned feature

[5] In this example, it happened to have obtained non-negative feature importance scores. However, it is not guaranteed if no FMN is applied in practice.

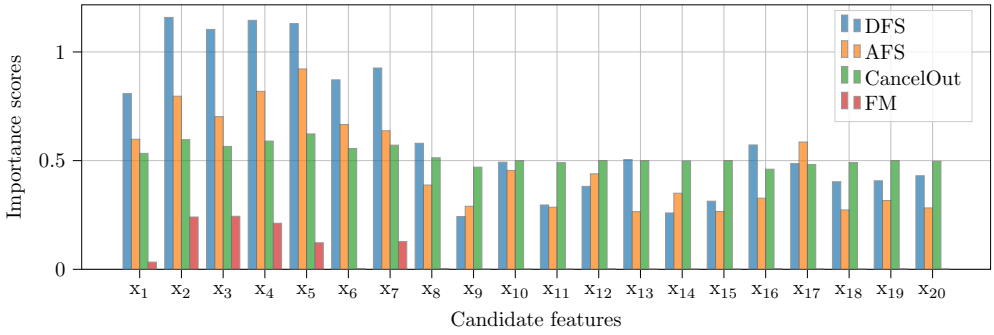


Figure 3.18: Feature selection performance under the exposure of a redundant candidate feature $x_7 = x_6$.

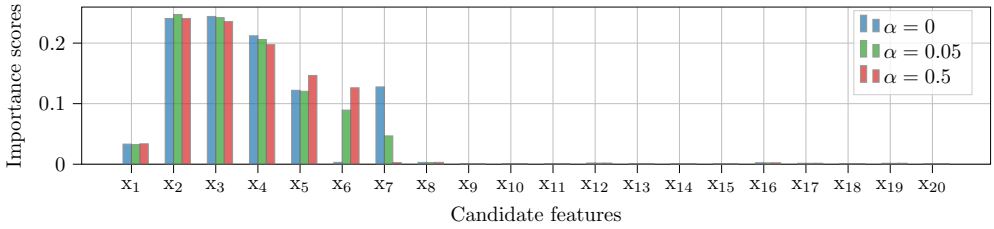


Figure 3.19: Feature selection performance under the exposure of a redundant candidate feature $x_7 = x_6$ with noise.

masks. It is plausible that the noise level in the redundant features is automatically identified by the FM-method as well. For example, given a small noise $\alpha = 0.05$, x_6 was assigned a moderately more important score than x_7, while a large noise level of $\alpha = 0.5$ led to the observation that x_6 was significantly more important than x_7, where x_7 in this cases was no longer important for correctly predicting the target y.

3.4.3 Computational Complexity

Despite the efficient training of neural networks due to parallel computing on GPUs, it is essential to justify the computational complexity to have a comprehensive understanding of our method. To enable this, the batch size is denoted as b and the number of raw features is denoted as d. Moreover, the reference method AFS and our FM method

have an additional d' to denote the latent dimensions for the non-linear transformation. This section omits the bias terms in all layers for simplicity.

Figure 3.20 shows the actual training time consumption and the computational complexity for a minibatch, where we omit the complexity of the learning network $g(\cdot)$. Broadly speaking, CancelOut and DFS have the lowest time complexity $O(bd)$ for learning the corresponding feature masks, because they do not have an additional mapping between the input data and the learnable feature masks. In contrast, AFS and our FM-method have a slightly higher time complexity $O(bdd')$ due to the matrix multiplications due to the non-linear transformation[6].

Although our method unfortunately does not have the lowest computational complexity, in practice, this drawback is actually almost negligible because the downstream neural network $g(\cdot)$ normally has significantly larger computational complexity. To show this, Figure 3.20 plots the actual training time for all DL-based feature selection approaches trained on MNIST with 100 epochs as an example. Apparently, extremely small differences in terms of computing time can be observed among different methods. It is interesting to see that our FM-method has less time consumption than that of the CancelOut method, although CancelOut has a lower computational complexity. The feasible reason is that it has more loss terms and requires additional regularization terms for training, which leads the extra time consumption. The experimental results show that our FM-method can provide reliable and accurate feature selection results with limited or even no additional time consumption in practice.

3.4.4 Hyperparameters

Deeper FM-module

The non-linear transformation block in the FM-module can be extended to multiple layers in practice, i.e. stacking a couple of layers as $\boldsymbol{z}_i = f_1 \circ f_2 \circ \cdots \circ f_l(\boldsymbol{x}_i)$, where $f_i(\cdot)$ is one fully-connected layer. This can be specified by users or practitioners. However,

[6]The structure of CancelOut and DFS follows the one presented in Figure 2.1. AFS can have much more complex architecture and here we consider its minimal case, which has the same non-linear transformation block as ours.

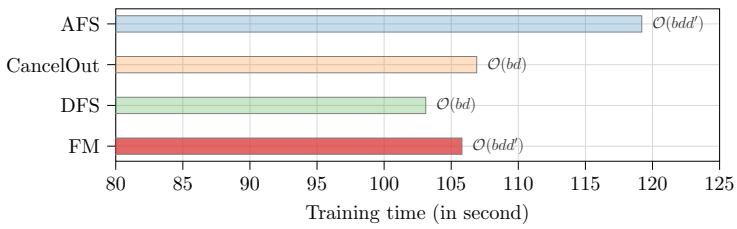


Figure 3.20: Comparison of the computational complexity and the actual time consumption of the reference methods and the FM-method. Our FM-method has larger computational complexity due to the non-linear transformation block within the FM-module. Nevertheless, the actual time consumption is comparable to the state-of-the-art.

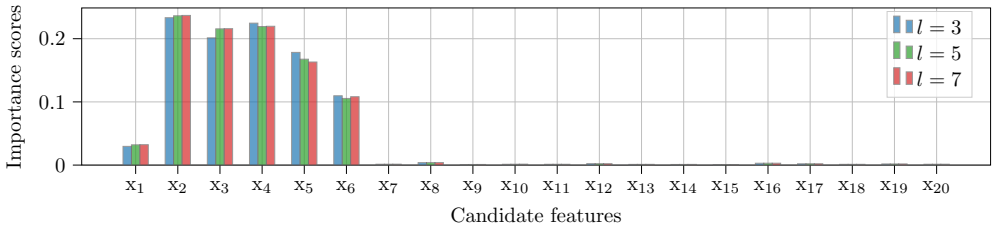


Figure 3.21: The learned feature masks with respect to three different network depths within the FM-module for the synthetic dataset.

the conducted experiments suggest that such extension is not necessary. More precisely, taking the MNIST dataset as an example, we compared FM-modules with six different depths from two layers to seven layers and the resulting accuracy was between 0.950 to 0.959.

Additionally, we compared three different depths (3 layers, 5 layers and 7 layers) within the FM-module on the synthetic dataset. The resulting feature importance scores are shown in Figure 3.21. Obviously, for each individual candidate feature, the FM-module of different depths finally learned similar scores, indicating an insensitivity to the network depth within the FM-module. A feasible reason is that the current FM-module already has enough modeling capability to learn reasonable feature masks. Therefore, we recommend that the users and practitioners can stick to the current implementation to save training time and storage consumption.

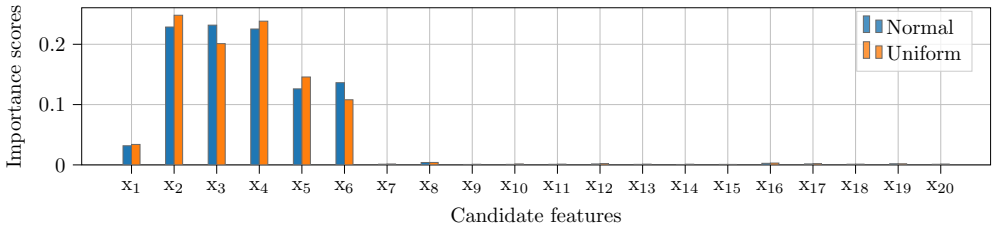


Figure 3.22: The learned feature masks with respect to different neural network initialization methods. It can be observed that the FM-method was not sensitive to different initialization methods for the neural network and learned similar feature masks.

Sensitivity to Initialization

As the FM-method is based on neural networks, we evaluated our method with respect to different network initialization methods: *i*) weights initialized from a uniform distribution; *ii*) weights initialized from a normal distribution; *iii*) weights initialized with constants (ones); *iv*) weights initialized by using Xavier methods [33]. The standard deviation of the final classification accuracy for different initialization methods was within 0.4% on the MNIST dataset, showing the stability of the selection results regarding different initialization methods.

Furthermore, to have a better understanding, Figure 3.22 compares the feature masks learned under the first two cases (*i* and *ii*), respectively. As expected, no significant differences can be observed between different initialization approaches. However, interestingly, we can see that for the relevant features of same predictive power (e.g., x_2, x_3 and x_4), the relative importance may slightly change due to the randomness, while such change does not affect the final selection quality.

Overall, the FM-method has only limited sensitivity to initialization. This is an important benefit of using our FM-method because some reference methods such as AFS and CancelOut explicitly require a careful initialization according to the authors [14, 37]. The insensitivity to initialization is valuable for users to easily deploy our FM-method to different applications without overly concerning random seeds.

Impact of Batch Size

The proposed batch-wise average calculates the primitive feature mask by averaging the intermediate results over a minibatch during each training iteration. Naturally, the batch size plays a critical role in the learning procedure. To justify this issue, using the MNIST dataset as an example, we evaluated the feature selection performance with respect to the following batch sizes: 4, 8, 16, 32, 64, 128, 256, 512 and 1024. Specifically, the model was trained for a fixed number of iterations with different batch sizes to obtain a fair comparison. Broadly speaking, the FM-method achieved a consistent performance with different batch sizes in a wide range from 16 to 1024 with a classification accuracy ranging from 0.951 to 0.956. Without surprise, extremely small batch sizes (4 and 8) led to worse performance with the accuracy of 0.930 and 0.947. Last but not least, although the differences in performance were limited, we still observed a slight tendency that larger batch sizes led to better selection quality. This matches our expectation because the batch-wise average can generate more representative $\bar{z}$ with larger batch sizes.

The experimental results on the synthetic dataset also matches the statement above as presented in Figure 3.23. We can easily see that batch size 1 could also assign large importance scores to the correct relevant candidate features. However, the order of importance among the relevant features was not fully correct. Moreover, irrelevant features were unexpectedly assigned with large scores, which matches the observation in the ablation study, because batch size of 1 can be understood as not using the BA block. Generally, for this simple small-scale synthetic dataset, correct selection result can be achieved already from the batch size of 4, also indicating the insensitivity of our FM-method for a wide range of batch sizes.

3.5 Discussion

The previous sections have demonstrated the effectiveness of the novel FM-method in a supervised setting for both classification and regression tasks. Due to the nature of DL-based feature selection structure, the choice of the downstream neural network is

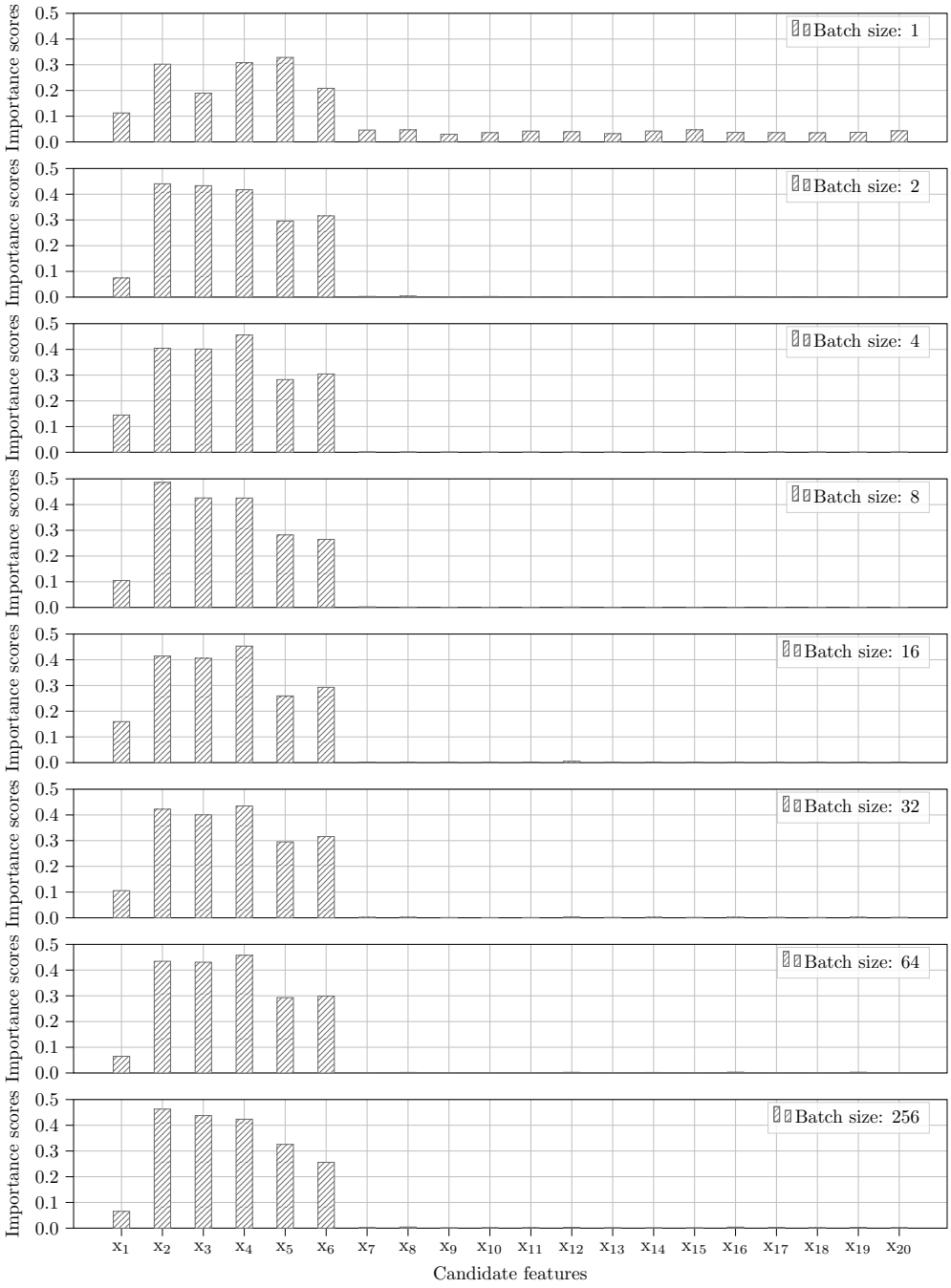


Figure 3.23: The learned feature masks with respect to different batch sizes for the synthetic dataset.

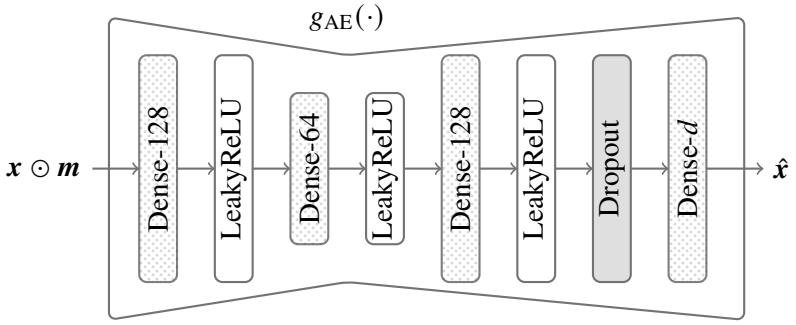


Figure 3.24: The architecture of the learning network used in the proposed FM-method for an unsupervised feature selection problem, where we maximize the similarity between $\hat{\boldsymbol{x}}$ and the raw input $\boldsymbol{x}$ as our learning objective.

flexible, meaning that this network can be constructed according to different requirements by users or practitioners. This enables our FM-method to select features in unsupervised settings as well.

In an unsupervised setting, we only need to use an autoencoder-like neural network $g_{\mathrm{AE}}(\cdot)$ as our downstream neural network for training and use an arbitrary differentiable similarity-based metric as the loss function such as MSE-loss, the minimization of which equals to maximize the similarity of the raw input data and the reconstructed data from the weighted input. Thereby, the resulting training procedure is summarized in Algorithm 2.

Algorithm 2 Training the Unsupervised Feature Mask Method

Require: Training data without labels $\mathbf{X}$, learning rate α, autoencoder-like neural network $g_{\mathrm{AE}}(\cdot)$ as the downstream neural network

Randomly initialize $f_{\mathrm{FM}}(\cdot)$ and $g_{\mathrm{AE}}(\cdot)$ with the initial parameters $\boldsymbol{\Theta}$
for $e = 1$ to maximal training epochs **do**
  for $b = 1$ to the number of minibatches **do**
    $\boldsymbol{m} \leftarrow f_{\mathrm{FM}}(\mathbf{X})$
    Calculate weighted inputs: $\mathbf{X} \odot \boldsymbol{m}$
    $\hat{\mathbf{X}} \leftarrow g_{\mathrm{AE}}(\mathbf{X} \odot \boldsymbol{m})$
    Calculate training loss $\mathcal{L} \leftarrow \mathcal{L}(\mathbf{X}, \hat{\mathbf{X}})$
    Update network parameters $\boldsymbol{\Theta} \leftarrow \boldsymbol{\Theta} - \alpha \cdot \nabla_{\boldsymbol{\Theta}} \mathcal{L}$
  end for
end for

Table 3.6: Downstream classification accuracy based on the top-50 selected features in an unsupervised learning setup (random forest as the classifier).

	MNIST	**fMNIST**	**Isolet**	**PCMAC**
RSF	0.842 (± 0.038)	0.828 (± 0.008)	0.835 (± 0.027)	0.584 (± 0.038)
RawF	0.969 (± 0.000)	0.876 (± 0.000)	0.931 (± 0.000)	0.930 (± 0.000)
ConAE	0.868 (± 0.027)	0.808 (± 0.024)	0.790 (± 0.030)	0.575 (± 0.011)
FM (ours)	**0.936** (± 0.009)	**0.842** (± 0.003)	**0.865** (± 0.005)	**0.692** (± 0.033)

Although unsupervised feature selection is not the main focus of this dissertation, this section still briefly demonstrates the performance on the aforementioned benchmarking datasets. Specifically, as shown in Figure 3.24, $g_{\text{AE}}(\cdot)$ was implemented with three hidden dense layers with 128, 64 and 128 neurons, respectively. Activation functions and other hyperparameters were maintained the same as before, since we only need to have a basic impression on the performance in an unsupervised setting.

Table 3.6 summarizes the quality of the selected top-50 features of our method as well as Concrete Autoencoder (ConAE) [4] which is considered as a state-of-the-art DL-based method for unsupervised feature selection. Our method outperformed ConAE on all four benchmarking datasets with significant margin, although we just followed the same hyperparameters from the previous sections and did not fine-tune them.

Interestingly, we can observe that the PCMAC dataset became notably more challenging in an unsupervised setting. One feasible reason is that this dataset contained sparse raw features which led to extremely small losses and gradients, making training notably more difficult than the cases with other datasets. One potential solution is to introduce entity embedding [38] to reduce the sparsity in the input space to improve training efficiency. This is, however, beyond the scope of this dissertation and we leave it as a future work.

3.6 Summary

This chapter presented a deep-learning-based feature selection approach by proposing a novel FM-module that acts as the basis of the entire dissertation. The components of the FM-module, namely the non-linear transformation block, the batch-wise average block and the feature mask normalization block, are differentiable, which enables a joint training with an arbitrary downstream neural network. It is worth noting that one of the major advantages of our FM-method is that no additional loss terms are introduced, leading to easy usage in practice.

Moreover, intensive experiments on synthetic data, benchmarking data and real-world data have confirmed the effectiveness and efficiency of our approach. Empirical studies have also shown that using the feature mask normalization can additionally lead to sparsity results similar to traditional LASSO-based approaches but without additional hyperparameters. Furthermore, our design can automatically avoid selecting redundant candidate features, leading to higher reliability and efficiency in practical applications. Although the conducted experiments have shown that the batch size could be the major hyperparameter of our FM-method, it is clear that our method can well perform for common batch sizes with superior selection quality.

Chapter 4

Conditional Feature Selection

4.1 Introduction

As introduced in Chapter 2 and Chapter 3, feature selection based on deep learning has drawn increasingly attention over the last few years. The previous studies have shown various research directions such as attention-based [37], sparsity-based [69] and stochastic-based [4] feature selection algorithms using deep learning. Despite the success of DL-based FS approaches in many different fields, the existing popular DL-based FS methods share a common characteristic that they are purely driven by data, while taking almost no expert knowledge into account. Indeed, data-driven techniques are overwhelming since the bloom of deep learning and big data, but we cannot deny that they might be risky and less efficient in the real world if they ignore valuable expert knowledge. In particular, the collected data in practice can be limited due to the time and cost constraints. In this case, data-driven approaches, especially deep learning, can overfit the training data, leading to poor generalization ability and being a threat to reliability in practice. Furthermore, in industrial scenarios, fruitful expert knowledge is typically available and can provide valuable information on the collected data, which has two major advantages. Firstly, using expert knowledge as certain constraints or regularization to data-driven algorithms can alleviate the overfitting problem. Secondly, expert knowledge as prior information can help algorithms to avoid exploring unnecessary cases, which saves time and money. Thereby, this chapter

focuses on bringing expert knowledge to feature selection based on a case study of real-world scenarios defined by the automatic test equipment manufacturer Advantest.

Post-Silicon Validation Modern semiconductor manufacturing is characterized by a complex workflow, including design phases, pre- and post-silicon validation, volume production and in-field test. As stated in [86, 87], Post-Silicon Validation (PSV) is generally known as one of the most difficult and expensive processes of the entire semiconductor validation procedure for chip design.

PSV typically consists of two main tasks, namely tuning and debugging. Tuning means to adjust the equipped tuning knobs on the manufactured devices to meet specifications or maximize some user-defined Figure-of-Merit (FoM), aiming to combat uncertainties during manufacturing such as process variations [68, 132]. Debugging is a procedure to identify the root causes for certain defects and flaws in manufactured devices and provide feedback and analysis back to the design phase. To elaborate further, Figure 4.1 shows a Device Under Test (DUT) in the middle. In PSV, a test case is understood as a record of various tuning conditions c_1 to c_n (e.g. process variations or operational conditions) and the settings of different tuning knobs t_1 to t_m with the final FoM y calculated from intermediate test results r_1 to r_l. From the machine learning perspective, a test case can be interpreted as a training data point. Having adequate test cases, PSV experts focus on analyzing potential relationships between different conditions, tuning knob settings and FoM. This analysis is normally done based on the experience of human experts with intensive manual inspection.

However, with the rapid development of semiconductor industry due to the ever increasing demand on chips in various fields, nowadays manufactured devices become more complex. Typically, they can be thus equipped with tens of tuning knobs and tested under up to hundreds of different operational conditions in order to meet the customers' requirements. From a data perspective, test cases of modern devices are of high dimensionality and are therefore difficult for human semiconductor experts to easily investigate and understand.

[‡]Contributions presented in this chapter were partially reported and published in our previous works [73, 77]. Some of the author's own formulations from [73, 77] are adopted in this chapter.

In order to enable efficient analysis in PSV, as shown in Chapter 3, we have successfully applied our FM-method to the real-world use case [75]. Nevertheless, the FM-method does not consider possibly available expert knowledge in practice and might face similar issues as described above. Moreover, in PSV, sometimes we already know some critical features (certain operational conditions or tuning knobs) that must be selected. Therefore, to take advantages of such prior knowledge and keep experts in the loop, we ask ourselves: How can we integrate the expert knowledge into training feature selection algorithms?

This chapter presents a neat solution by introducing a novel Conditional Feature Selection (CFS) framework that integrates expert knowledge as training conditions into our FM-method to obtain more accurate selection results with a high efficiency. More concretely, the proposed framework uses fully connected layers to automatically learn reasonable representations of the preselected features as conditions for training feature selection algorithms.

In summary, the main contributions of this chapter are listed below:

- A novel conditional feature selection framework is proposed to take into account expert knowledge during training;
- Experiments on both synthetic and real-world datasets from the leading manufacturer of automatic test equipment for semiconductors have shown that our method can effectively identify the most critical features;
- Our framework has provided a paradigm for other fellow data-driven approaches on integrating expert knowledge into learning algorithms for semiconductor testing.

4.2 Related Works

Different from Chapter 2, where we have systematically introduced feature selection techniques from a data science perspective, this section mainly reviews some previous studies which used feature selection specifically in the semiconductor manufacturing.

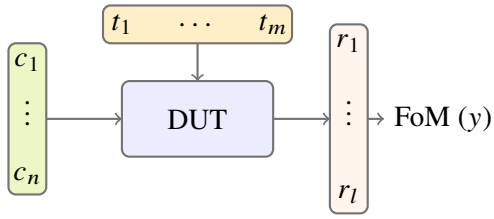


Figure 4.1: In post-silicon validation, for a modern DUT, there can be up to hundreds of tuning conditions (c_1 to c_n) and tens of tuning knobs (t_1 to t_m). The resulting high-dimensional data are difficult for human experts to investigate.

One early representative work is [22], in which Chen and Hong proposed a piecewise linear regression tree called SERT that combined forward feature selection and regression analysis for estimating the effects of individual attributes in terms of yield loss causes given a certain semiconductor manufacturing setting. Ye et al. [129] used minimum Redundancy and Maximum Relevance (mRMR) and correlation analysis to identify both critical and irrelevant syndromes (features) to enable a reasoning-based functional-fault diagnosis system. Kang et al. [52] proposed a novel random forward search and combined it with a backward elimination process to efficiently select the most important features to enhance virtual metrology modeling, which outperformed the ordinary forward-backward feature selection approach. Moreover, [20] used the ordinary LASSO method to select features in virtual metrology modeling as well. Afterwards, in [55] , LASSO was used in combination with partial least squares in a semi-supervised setup for in-process wafer quality analysis. Heo et al. [45] further considered interactions between candidate features in using LASSO for feature selection and improved the yield analysis capability in semiconductor fabrication lines. Despite the wide usage of LASSO as shown above, those methods were often restricted themselves in highly linear use cases. To enable non-linear feature selection, Mohammadi et al. [88] introduced a one-layer feed forward neural network using column-wise sparsity regularization to select the most critical process and performance parameters (features) for digital circuit timing analysis in both FinFET and Silicon Nanowire technologies. In [121], a feed-forward feature selection based on logistic regression models was proposed to enhance the cycle time prediction performance in wafer fabrication systems. Apparently, despite the wide application and development of feature selection, almost all existing feature selection approaches are purely data-driven techniques.

4.3 Methodology

To enable conditional feature selection with neural networks, we propose to fuse the preselected and candidate features based on learnable encoding neural layers and feed the fused representations to a neural network targeting a given learning task (e.g. regression or classification). Accordingly, the neural network acts as a guide in a way that the candidate features, which minimize training losses under the condition of having given the preselected features, should be assigned with greater importance scores. Thereby, after training, the most critical candidate features can be easily determined based on the learned scores as the ordinary FM-method does.

4.3.1 Notations

In this chapter, we use the following notations. The input data are denoted as a matrix $\mathbf{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n]^T \in \mathbb{R}^{n \times d}$, where n is the number of data points (test cases) and d is the number of input features (tuning conditions and knobs). In the following, we also use $\{\mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{x}_d\}$ to denote the d different input features of a given dataset for better readability. According to expert knowledge, $\mathbf{X}$ is composed of two parts as $\mathbf{X} = [\mathbf{X}_p, \mathbf{X}_c]$. That is, the preselected d_p features $\mathbf{X}_p = [\boldsymbol{x}_{p1}, \boldsymbol{x}_{p2}, \ldots, \boldsymbol{x}_{pn}]^T \in \mathbb{R}^{n \times d_p}$ and the d_c candidate features $\mathbf{X}_c = [\boldsymbol{x}_{c1}, \boldsymbol{x}_{c2}, \ldots, \boldsymbol{x}_{cn}]^T \in \mathbb{R}^{n \times d_c}$ with $d = d_c + d_p$. Furthermore, for a supervised conditional feature selection, the labels (FoM or target features) are denoted as $\mathbf{Y} = [\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_n]^T \in \mathbb{R}^{n \times d_y}$, where d_y is defined by the given learning task. For example, $d_y = 1$ for univariate regression which is one of the most common cases in post-silicon validation.

4.3.2 Conditional Feature Selection

The overall framework is illustrated in Figure 4.2. Broadly speaking, it consists of three major components: *i*) an FM-module as a learnable feature weighting block; *ii*) a representation fusion block notated as ⊠; and *iii*) a downstream neural network $g(\cdot)$.

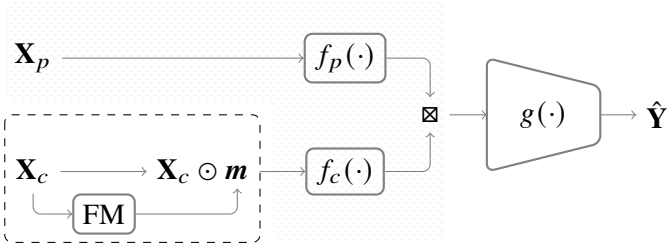


Figure 4.2: The proposed generic framework for conditional feature selection based on neural networks. $f_p(\cdot)$ and $f_c(\cdot)$ are used to embed preselected and candidate features, respectively.

Learnable Feature Weighting Block

This block is designed to learn a feature mask $\boldsymbol{m} = [m_1, m_2, \ldots, m_{d_c}]^T \in \mathbb{R}^{d_c}$ with $m_i \in (0, 1)$ and $\sum_{i=1}^{d_c} m_i = 1$ which denotes the importance of the corresponding i-th candidate feature. This is exactly the FM-module proposed in the previous section but applied to $\mathbf{X}_c$ only, so the feature mask is calculated as

$$\boldsymbol{m} = \mathrm{FM}(\mathbf{X}_c) = softmax\Big(\frac{1}{b}\sum_{i=1}^{b} \mathbf{W}_2 tanh(\mathbf{W}_1 \cdot \boldsymbol{x}_c + \boldsymbol{b}_1) + \boldsymbol{b}_2\Big), \tag{4.1}$$

where $\mathbf{W}_1 \in \mathbb{R}^{d_c' \times d_c}$, $\boldsymbol{b}_1 \in \mathbb{R}^{d_c'}$, $\mathbf{W}_2 \in \mathbb{R}^{d_c \times d_c'}$, $\boldsymbol{b}_2 \in \mathbb{R}^{d_c}$ are trainable parameters of the FM-module and b is the minibatch size during training. The symbol $\odot$ in Figure 4.2 denotes the element-wise multiplication; i.e. $\mathbf{X}_c \odot \boldsymbol{m} = [\boldsymbol{x}_{c1} \odot \boldsymbol{m}, \boldsymbol{x}_{c2} \odot \boldsymbol{m}, \ldots, \boldsymbol{x}_{cb} \odot \boldsymbol{m}]^T \in \mathbb{R}^{b \times d_c}$, corresponding to weighted candidate features. After training, we obtain the final $\boldsymbol{m}$ by feeding all training data into the trained FM-module. Then we identify the k most important candidate features according to the top-k largest entries in $\boldsymbol{m}$.

Representation Fusion Block

The most conspicuous design in our framework is that we use two learnable non-linear functions $f_p(\cdot)$ and $f_c(\cdot)$ to respectively encode the preselected and candidate features. Both non-linear functions are implemented by fully connected layers in this work.

More precisely, $f_p(\cdot)$ is defined as

$$f_p(\boldsymbol{x}_p) = \sigma(\mathbf{W}_p \cdot \boldsymbol{x}_p + \boldsymbol{b}_p), \tag{4.2}$$

and $f_c(\cdot)$ is defined as

$$f_c(\boldsymbol{x}_c \odot \boldsymbol{m}) = \sigma(\mathbf{W}_c \cdot (\boldsymbol{x}_c \odot \boldsymbol{m}) + \boldsymbol{b}_c), \tag{4.3}$$

where $\mathbf{W}_p \in \mathbb{R}^{d'_p \times d_p}$, $\boldsymbol{b}_p \in \mathbb{R}^{d'_p}$, $\mathbf{W}_c \in \mathbb{R}^{d'_c \times d_c}$, $\boldsymbol{b}_c \in \mathbb{R}^{d'_c}$ are trainable network parameters (i.e. weights and biases) and d'_p and d'_c are user-specified dimensions of the encoded representations. Moreover, $\sigma(\cdot)$ is an activation function to provide non-linearity and $\boxtimes$ denotes the concatenation between the two encoded representations. That is to say, the concatenated representation $\boldsymbol{z}$ is defined as

$$\boldsymbol{z} = f_p(\boldsymbol{x}_p) \boxtimes f_c(\boldsymbol{x}_c \odot \boldsymbol{m}) = \begin{bmatrix} f_p(\boldsymbol{x}_p) \\ f_c(\boldsymbol{x}_c \odot \boldsymbol{m}) \end{bmatrix}, \tag{4.4}$$

where $\boldsymbol{z} \in \mathbb{R}^{d'_p + d'_c}$.

Task-Specific Neural Network

Similar to other DL-based feature selection approaches such as [4, 37, 72], there are no special requirements on the structure of the task-specific neural network $g(\cdot)$. In this work, $g(\cdot)$ is implemented by a few fully connected layers parameterized by $\boldsymbol{\Theta}_n$ for simplicity.

4.3.3 Learning Objective

In PSV, we aim to identify which candidate features can best predict the FoM value which is typically a continuous numerical value. Naturally, this corresponds to a regression task for the entire proposed framework. Therefore, the canonical loss

function can be the Mean Squared Error (MSE) loss as

$$\mathcal{L}_{\text{MSE}} = \frac{1}{n}\sum_{i=1}^{n} ||\boldsymbol{y}_i - \hat{\boldsymbol{y}}_i||_2^2, \tag{4.5}$$

where $||\cdot||_2^2$ denotes the squared 2-norm. Accordingly, the overall training objective is to minimize the MSE loss as

$$\underset{\boldsymbol{\Theta}}{\text{argmin}} \frac{1}{n}\sum_{i=1}^{n} ||\boldsymbol{y}_i - g(f_p(\boldsymbol{x}_p) \boxtimes f_c(\boldsymbol{x}_c \odot \boldsymbol{m}))||_2^2, \tag{4.6}$$

where $\boldsymbol{\Theta}$ denotes all trainable parameters of the entire framework (FM, f_p, f_c and g). As a result, the corresponding training procedure of CFS is summarized in Algorithm 3.

Algorithm 3 Training Conditional Feature Selection

Require: Training dataset tuple $(\boldsymbol{x}_p, \boldsymbol{x}_c, \boldsymbol{y})$ in minibatch $(\mathbf{X}_p, \mathbf{X}_c, \mathbf{Y})$, learning rate α

Randomly initialize the entire network with the initial parameters $\boldsymbol{\Theta}$
for $e = 1$ to maximal training epochs **do**
  for $b = 1$ to the number of minibatches **do**
    Encode preselected features: $f_p(\boldsymbol{x}_p)$
    Calculate the feature mask: $\boldsymbol{m} = \text{FM}(\mathbf{X}_c)$
    Encode candidate features: $f_c(\boldsymbol{x}_c \odot \boldsymbol{m})$
    Concatenation: $\boldsymbol{z} = f_p(\boldsymbol{x}_p) \boxtimes f_c(\boldsymbol{x}_c \odot \boldsymbol{m})$
    Obtain current prediction $\hat{\boldsymbol{y}} = g(\boldsymbol{z})$
    Calculate training loss $\mathcal{L}_b = \frac{1}{b}\sum_{i=1}^{b} ||\boldsymbol{y}_i - \hat{\boldsymbol{y}}_i||^2$
    Update network parameters $\boldsymbol{\Theta} \leftarrow \boldsymbol{\Theta} - \alpha \cdot \nabla_{\boldsymbol{\Theta}} \mathcal{L}_b$
  end for
end for

4.3.4 Implementation

In this work, the task-specific network $g(\cdot)$ was implemented with a two-hidden-layer neural network as shown in Figure 4.3. Both layers were fully connected layers with 128, 64 neurons respectively. LeakyReLU [84] with the ratio of 0.02 was used as the activation function for each hidden layer. Before the output layer, we used

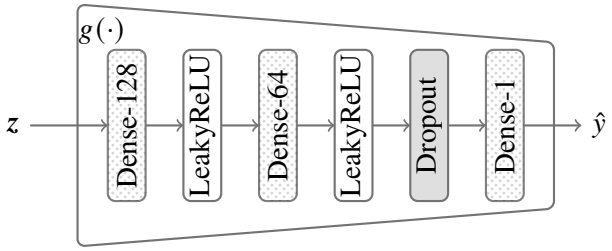


Figure 4.3: The architecture of the task-specific downstream neural network $g(\cdot)$.

Dropout [112] with a ratio of 0.3 to avoid overfitting during training. We used the Adam optimizer [56], which is a modern variant of gradient-descent algorithms, to train our neural networks.

4.4 Evaluations on Synthetic Datasets

Before diving into real-world datasets, we firstly conducted experiments on a synthetic dataset to justify the effectiveness and obtain a primary understanding of the properties of the CFS framework. In particular, we constructed a synthetic dataset of 2000 training samples with 15 input features x_1 to x_{15}. Moreover, we assumed that the regression target y was related to 6 out of 15 input features based on the formula

$$y = x_1^2 + 10x_2x_3x_4 + 5x_5x_6 + \epsilon, \tag{4.7}$$

where $\epsilon \sim N(0, 1)$, $x_i \sim U(0, 1)$ and all candidate features are independent of each other. In this synthetic dataset, only the first 6 features x_1 to x_6 are relevant for predicting y, while the remaining 9 input features from x_7 to x_{15} are irrelevant.

In the following subsections, we individually divided the input features into preselected and candidate features to investigate whether our approach can learn correct importance scores for the candidate features given certain preselected features.

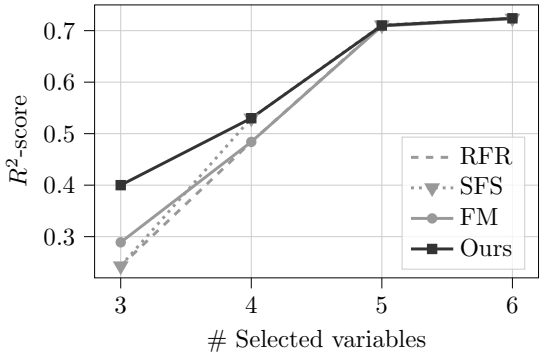


Figure 4.4: Regression performance comparison between our CFS method and the reference methods.

4.4.1 Comparison with Reference Methods

We compared our CFS framework to three popular reference feature selection methods, namely two conventional approaches, Random Forest Regression (RFR) [16] and Sequential Forward Selection (SFS) [30], as well as our FM-method introduced in the previous chapter. The CFS method and the three reference methods were trained on the synthetic data where x_2 and x_3 were preselected as training condition. Accordingly, the rest 13 input features were considered as candidate features and accessible for the reference methods, while our CFS approach was able to consider the preselected features during training. It should be noted that the reference methods cannot consider preselected features, so x_2 and x_3 were not accessible during training for them.

Figure 4.4 presents the R^2-scores [9] with respect to different selected feature subset sizes. Our approach significantly outperformed the reference methods with more than 36% improvement when selecting the most important three features. All methods performed the same when selecting five or six features (including the preselected two features), which matched our expectation because x_1 to x_6 were indeed relevant to the target variable y according to the definition in Eq. 4.7.

In Figure 4.5, for a more intuitive understanding, we further plot the learned feature importance scores of the CFS method and the FM-method, respectively. The bar plots clearly show the difference between using and without using expert knowledge

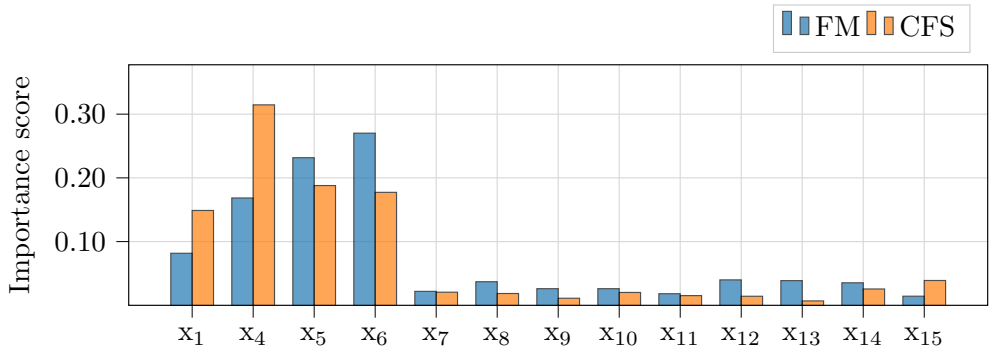


Figure 4.5: Comparison of the learned feature importance scores between the novel CFS method and the FM method, given the preselected features x_2 and x_3.

(i.e. preselected features) for feature importance learning. Due to the lack of training condition, the original FM-method mistakenly assigned the largest importance scores to x_5 and x_6, while the CFS method successfully identify the truly important feature x_4 given the condition of x_2 and x_3 due to the large factor before them. In addition, both methods can successfully identify irrelevant candidate features by assigning them with extremely small values.

4.4.2 Single Preselected Feature as Condition

In the previous section, we have shown the case where multiple features are preselected. This section evaluates the case for one preselected feature only. Figure 4.6 shows the learned feature importance scores given the preselected feature x_1 as the training condition. It can be easily seen that x_2 to x_6 were successfully assigned with large importance scores after training, while the remaining nine features were with extremely small scores. This matches the relation defined in Eq. 4.7. In a separate experiment, we considered x_2 as the preselected feature as training condition. The resulting learned scores are presented in Figure 4.7. As expected, the irrelevant features from x_7 to x_{15} were still assigned with small values in comparison to the relevant features (x_1, x_3 to x_6). Another inspiring observation is that the learned importance score really reflects the relative contribution of each individual candidate feature, meaning that x_3 and x_4

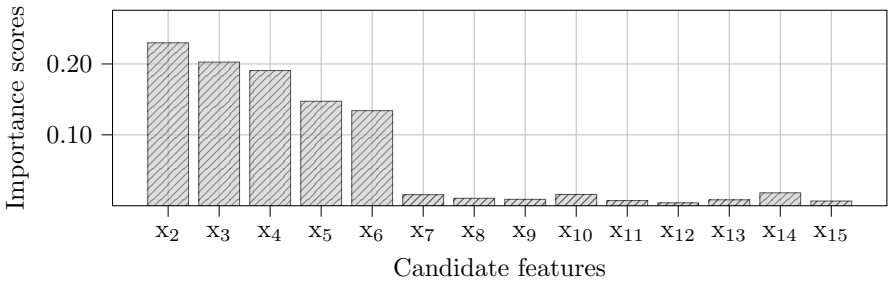


Figure 4.6: The learned feature importance scores when x_1 was preselected as training condition.

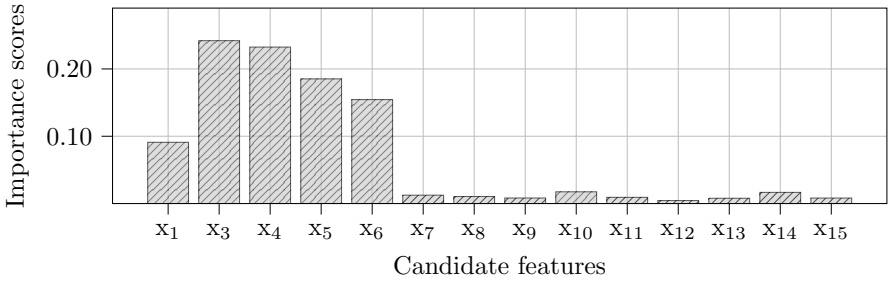


Figure 4.7: The learned feature importance scores when x_2 was preselected as training condition.

were more important under the condition of x_2, while x_1 had the smallest importance among all relevant candidate features.

4.4.3 Conditional Feature Selection with Redundant Features

Avoiding selecting redundant features is a challenging task in feature selection [40]. Nevertheless, as the implementation of CFS is inherited from the FM-method, it is expected that it can avoid selecting redundant features. To justify how our algorithm behaves in the presence of redundant features, we additionally let the input feature $x_7 = x_1^2$ be a redundant feature towards x_1, and x_7 was the preselected feature as training condition. From Figure 4.8, we can easily see that x_2 to x_6 were successfully assigned with large importance and thus selected, while the redundant feature x_1 was with an extremely small score. This means that our method can automatically avoid selecting

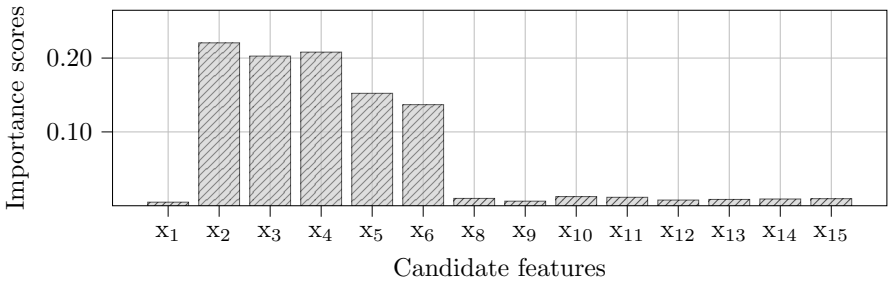


Figure 4.8: The learned feature importance scores when x_7 was redundant to x_1 and preselected.

the candidate features which are redundant to the conditions (preselected features). This property also confirms that the condition has been successfully involved into the training procedure for feature selection.

4.5 Evaluations on Real-World Dataset

This section investigates the feature selection performance of our novel CFS framework on a real-world post-silicon validation dataset introduced in Chapter 2.

4.5.1 Comparison with the State-Of-The-Art

In this experiment, we considered all 900,000 test cases. In particular, x_1 was preselected as training condition according to expert knowledge. Figure 4.9 compares the learned importance scores of the CFS method and the FM-method, where we excluded the conventional feature selection algorithms (RFR and SFS) which could not well scale to high-volume data.

It is evident that using expert knowledge (preselected features) as training condition led to different feature importance scores on the real-world dataset. More precisely, given x_1 as condition, the top-3 features for CFS are x_1, x_8 and x_9, while the top-3 features for the FM-method are x_1, x_2 and x_8. To evaluate the selected features of both methods, we used the selected features to predict the target variable. The

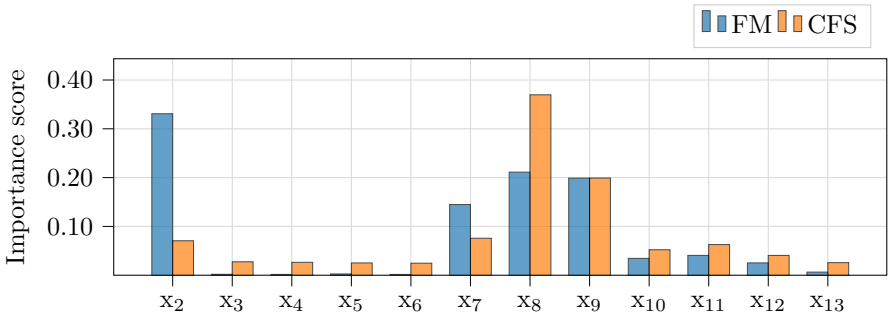


Figure 4.9: Comparison of the learned feature importance scores when x_1 was preselected as training condition.

corresponding experimental results are shown in TABLE 4.1. As expected, our CFS method outperformed the original FM method with more than 36% improvement in terms of R^2-score. Furthermore, the order of importance scores obtained by the CFS method matched the ground truth based on an exhaustive search by the PSV experts. The experiment clearly shows that in practice the integration of expert knowledge to the data-driven algorithms can indeed enhance the performance, which has been, however, rarely explored in the feature selection field.

Table 4.1: Predictive power of the selected features.

	FM	CFS	Gain
R^2-score	0.556	**0.759**	36.5%↑

4.5.2 Conditional Feature Selection for a Single Device

In post-silicon validation, domain experts often face two common scenarios, i.e. tuning a specific DUT for deep investigation and tuning multiple DUTs for a comprehensive overview. This section evaluates the CFS framework on a single DUT and the next section evaluates it on multiple DUTs.

Specifically, we consider all test cases obtained from one single DUT only in this section. It should be noted that two input features were removed before training on a

single DUT in this subsection, because they were identical within a fixed DUT, meaning that we considered the rest 11 features in total.

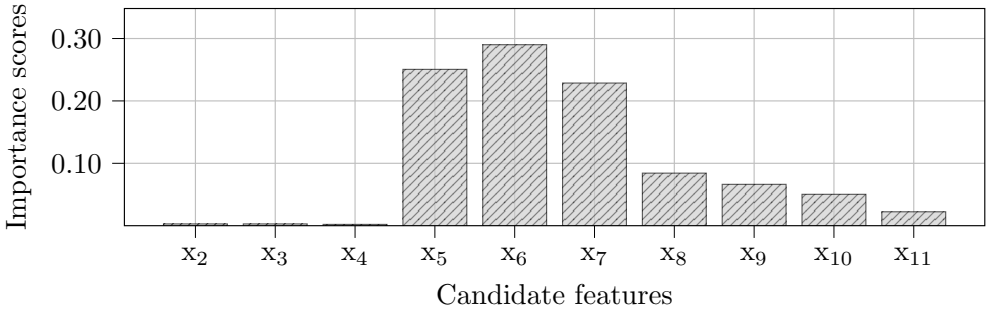
Figure 4.10a and Figure 4.10b show the learned importance scores by preselecting x_1 and x_6 as conditions, respectively. Evidently, given x_1 as condition, x_5 to x_{11} were all important, as x_5 to x_{11} had significantly larger scores than those of other features. Interestingly, given x_6 as conditions, x_5 and x_7 were the two most important candidate features, while x_1 seemed to be irrelevant in this case. This result matched the ground truth according to a separate exhaustive search. Furthermore, we simultaneously let x_{10} and x_{11} be the preselected features as the training condition. Figure 4.10c presents the importance scores. We can obviously observe that x_6 is notably more important than other candidate features, taking x_{10} and x_{11} into consideration. The experiment clearly shows that different conditions may lead to different importance for the same candidate features, revealing the significance of our CFS framework.

As a comparison, an exhaustive search can take significantly longer time than our algorithm. For example, considering a case where we want to identify the most critical five features given one preselected feature as condition, an exhaustive search requires to explore $\binom{10}{5} = 252$ candidate feature combinations for a single DUT. This implies that we need to train a model for 252 times to obtain the final selection results, while our method can provide the importance scores for training only once. The high efficiency of the proposed CFS framework enables almost real-time usage in practice.

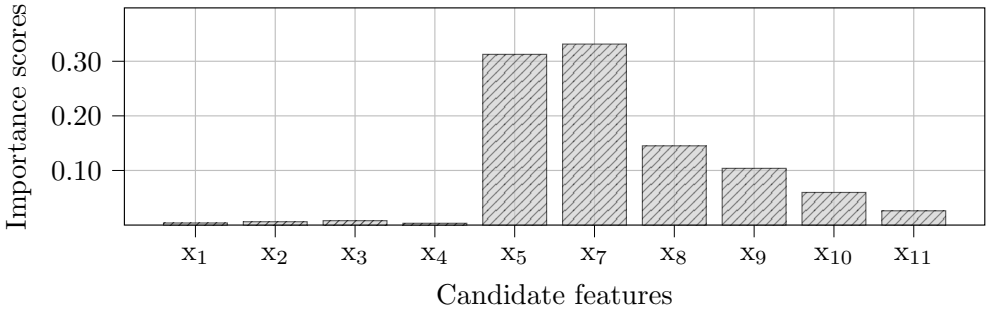
4.5.3 Conditional Feature Selection on Multiple Devices

In this experiment, we considered all 900,000 test cases from 9 different DUTs. As an example, x_1 was preselected as training condition and Figure 4.11 shows the learned importance scores. We can observe that x_2 and x_7 to x_{11} were notably more critical than the other candidate features under the condition of preselecting x_1, which matched the ground truth obtained from an exhaustive search by the PSV experts.

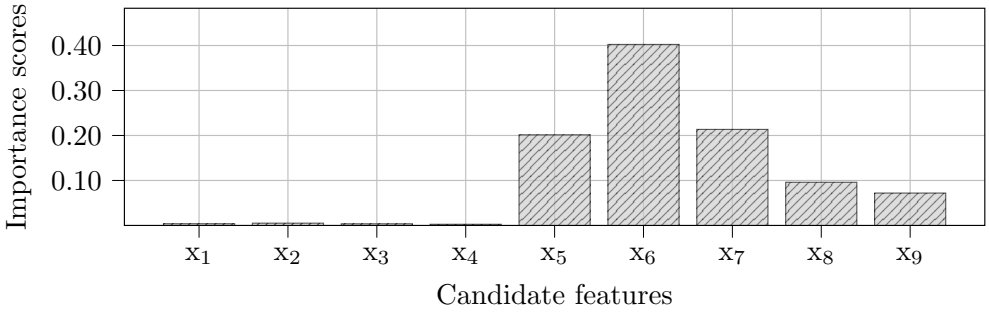
Next, we justified the effectiveness of our framework under redundant candidate features. In particular, we constructed a redundant candidate feature x_{8r} towards x_8 by



(a) The learned feature importance scores when x_1 is preselected as condition.



(b) The learned feature importance scores when x_6 is preselected as condition.



(c) The learned feature importance scores when x_{10} and x_{11} are preselected as condition.

Figure 4.10: Experiments on a single DUT.

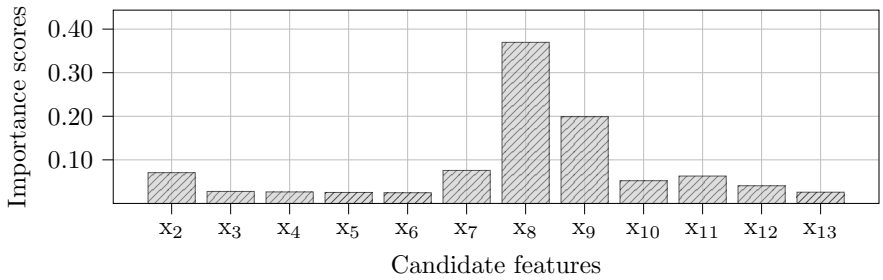


Figure 4.11: The learned feature importance scores when x_1 is preselected as condition on all nine DUTs.

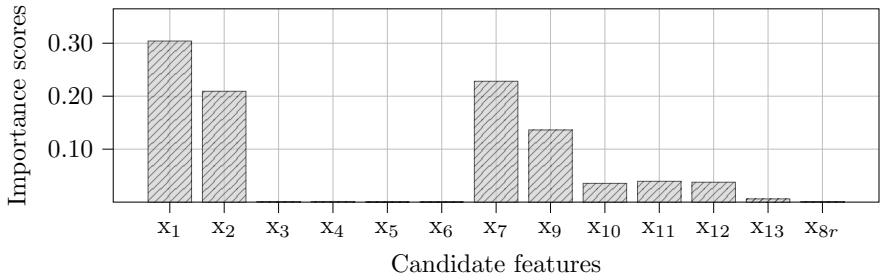


Figure 4.12: The learned feature importance scores when x_8 is selected as condition on all nine DUTs and a redundant feature v_{8r} was added to the candidate feature list.

a simple duplication as $x_{8r} = x_8$. The resulting importance scores are shown in Figure 4.12. As expected, x_{8r} was assigned with extremely small score and not selected by our algorithm.

4.5.4 Unexpected Discovery

Figure 4.12 shows that our algorithm assigned similar scores to both x_1 and x_2, which surprised the domain experts of the company that provided us the dataset. According to the experts' experience, x_1 and x_2 should be redundant to each other. However, our algorithm assigned apparently large importance scores to both features. In addition, in Figure 4.11, x_2 was also important under the condition of preselecting x_1, seemingly contradicting to the previous experiments for redundant features.

This observation reveals a disagreement between observations of data-driven approaches and expert knowledge, motivating the domain experts to rethink the relation between x_1 and x_2. Coincidentally, in this dataset, there exists a relation between both features as

$$x_2 = \begin{cases} 1, & \text{if } x_1 \in \{1, 2, 3\} \\ 2, & \text{if } x_1 \in \{4, 5, 6\} \\ 3, & \text{if } x_1 \in \{7, 8, 9\} \end{cases} \tag{4.8}$$

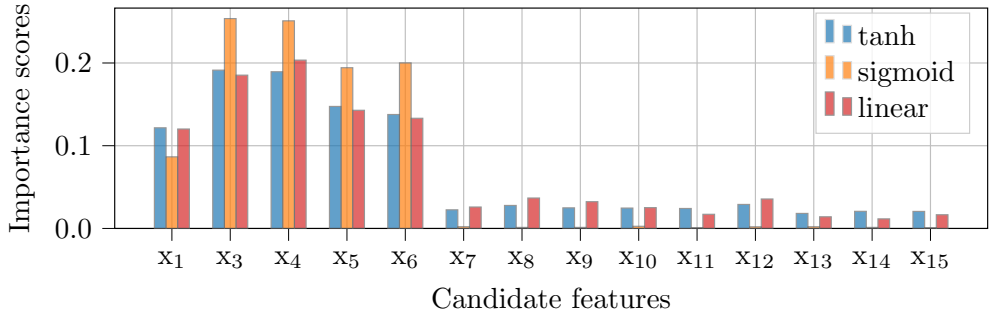
This suggests that x_1 and x_2 might have provided information of different levels during training. In other words, x_1 provided information for considering test cases into 9 groups, while x_2 provided information for categorizing test cases into 3 groups. This means that both features provide information of different levels and are not fully redundant to each other.

4.6 Study of Hyperparameters

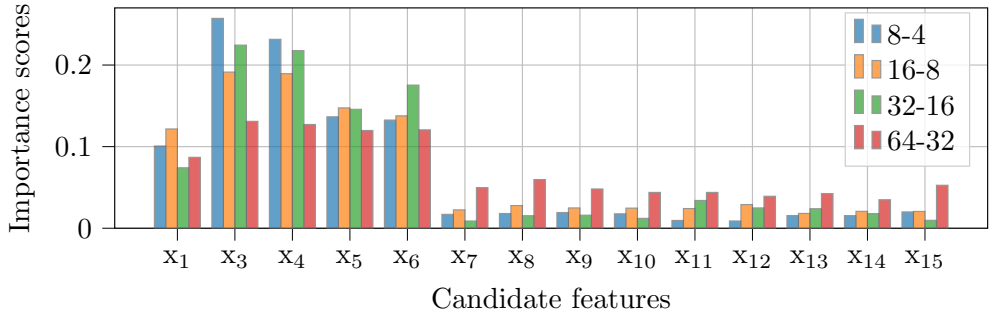
As introduced before, the main idea is to introduce two embedding layers to both candidate and preselected features before the fusion module (concatenation). Naturally, two key hyperparameters are the activation function within the embedding layers and the capacity of the embedding layers. Therefore, this section justifies the performance of CFS under different hyperparameters. In particular, we conducted experiments on the synthetic data and considered x_2 as the preselected feature as an example.

Figure 4.13a compares three different popular activation functions of the embedding layer, namely *tanh*, *sigmoid* and linear activations. It is clear to see that all three activation functions can learn correct importance scores. Interestingly, *tanh* and linear activation functions performed similarly, while *sigmoid* assigned significantly smaller scores to the irrelevant candidate features. Nevertheless, all three activation functions can be used reliably in practice. This additionally means that for this PSV dataset, nonlinear embedding layers are not necessary and we can use linear embedding layers to save training time by avoiding the exponential computations.

Figure 4.13b compares four different capacities. The notation of α-β denotes that the candidate features are encoded into an α-dimensional vector, while the preselected



(a) The learned feature masks with respect to different activation functions within the embedding layers. We can observe that the relevant features can be successfully selected by using all three activation functions, although they led to slightly different absolute importance scores.



(b) The learned feature masks with respect to different capacities of the embedding layers. Generally speaking, the relevant features were correctly identified for all four cases. We can observe that the embedding layers with too large capacity (64-32) assigned moderately larger scores to the irrelevant candidate features. Therefore, it is recommend to use embedding layers with small capacity in practice as a start point.

Figure 4.13: Hyperparameter study in CFS.

features are encoded into an β-dimensional vector. Generally speaking, all four combinations work well for the conditional feature selection. However, we can observe that a too large capacity (i.e. 64-32) might give overly large importance to irrelevant features, because the neural network was trying to learn mapping between noise and targets. On the contrary, with reasonable capacity, the learned importance scores are similar and stable. Practically, we recommend to use smaller capacity as a start point to fine-tune this hyperparameter.

In total, although the two hyperparameters can affect the learned importance scores, the final importance order of different candidate features maintain the same with respect to a wide range of hyperparameter settings. This is valuable for practical usage.

4.7 Summary

This chapter proposed a novel framework of conditional feature selection for efficient post-silicon analysis by taking expert knowledge into account during training. Our approach was based on artificial neural networks and can thus easily handle large-scale data of high-dimensionality.

The experiments conducted and presented in the previous sections have shown how our conditional feature selection algorithm assists expert in efficiently identifying the most critical candidate features when presenting a few preselected features as conditions. However, the potential of the proposed idea goes beyond this. Firstly, as demonstrated above, the preselected features were used as expert knowledge to guide the training procedure. Actually, we suggest that almost any other meta information, in addition to expert knowledge, can be used in our framework by considering them as $\mathbf{X}_p$. For example, manufacturing dates and locations, process parameters and chip specifications can be fed to our framework as conditions for training.

Furthermore, the idea of keeping expert in the loop can be applied to other related fields such as wafer map defect pattern classification, where many recent studies [81] have focused on using deep neural networks to perform classification. In this case, some

expert knowledge such as wafer test lots or historical test information can be fused and fed to the original neural network to further enhance the efficiency and effectiveness.

Last but not least, this work is expected to inspire other fellow researchers to consider expert knowledge into data-driven approaches to enhance the overall efficiency, reliability and performance.

Chapter 5

Complementary Feature Mask

5.1 Introduction

The previous chapters have proposed the novel FM-method and one of its special extension forms, conditional feature selection based on the FM-module. In addition, several representative DL-based feature selection approaches have been briefly introduced and compared. According to the literature and the conducted experiments, our method and the contemporary reference methods can generally handle many feature selection tasks with high efficiency and reliable performance. Nevertheless, we ask ourselves: Can we go one step further and improve the selection performance?

To answer the question above, we need to revisit existing DL-based feature selection approaches. As introduced in Chapter 2, existing DL-based FS methods typically have various design philosophies such as sparsity regularization, attention mechanism or stochastic search. Despite such variety, most of these approaches are trained by feeding the weighted input data (i.e. the Hadamard product of the raw features and feature masks) to a downstream neural network dedicated to a given learning task. This leads to an interesting observation that the candidate features with large weights are dominant for the downstream NN, while the candidate features with small weights might be not sufficiently considered during training.

[‡]Contributions presented in this chapter were partially reported and published in our previous works [78]. Some of the author's own formulations from [78] are adopted in this chapter.

This raises two concerns. The first concern comes from the possible neglect of down-weighted but relevant candidate features during training. Put simply, due to the nature of (minibatch) stochastic gradient descent algorithms for training neural networks, the feature mask at a certain training iteration might coincidentally have large weights to some strongly relevant candidate features as well as some weakly relevant or even irrelevant features, while the left relevant features can be unfortunately assigned with small weights. Such a feature mask is actually suboptimal and not desirable, but the training loss at the current iteration can be still very small due to the power of the downstream neural network, because the downstream neural network still has access to all candidate features in a weighted fashion, or if the network $g(\cdot)$ has a too large model capacity and is overfitted to the downstream task. As a result, the entire algorithm can get stuck at some local minima and fail to explore other better feature masks. This concern has unfortunately been rarely discussed in the literature. To the best of our knowledge, only a recent work [131] has slightly discussed similar observations.

Secondly, it is reasonable that a strong predictive power (e.g. high accuracy for classification) relies on relevant candidate features. However, what is the other side of this statement? Is poor predictive capability equal to irrelevant features only? We can imagine a feature selection problem targeting a binary classification task. If the selected features result in the poorest prediction performance which is exactly an accuracy of 0%, these selected features actually can optimally discriminate both classes by simply inverting the predicted class labels. In other words, these features are indeed relevant for the given learning task. Based on this example, we argue that a poor predictive power in the context of feature selection is the most *uncertain* predictive ability. Unfortunately, to the best of our knowledge, no prior work has considered this property during designing FS algorithms using deep learning.

The two concerns briefly introduced above motivate us to rethink the feature selection problem by considering the candidate features with small importance scores during training. That is to say, during training, the feature mask should assign small weights to as few relevant features as possible, and meanwhile, the features with small scores should have uncertain predictive capability. To this end, this chapter describes a feature selection framework based on a novel complementary feature mask. In addition to learning an ordinary feature mask only, we specifically define a complementary

feature mask and use it to improve the feature selection quality.

In a nutshell, the main contributions of this chapter are summarized as follows:

- We have proposed a novel complementary feature mask method that considers features assigned with small or incorrect importance scores during training;
- A generic multi-task feature selection framework leveraging the complementary feature mask is presented which can be considered as a paradigm for designing new feature selection algorithms;
- We provide an implementation of the new framework and have conducted extensive experiments to show its effectiveness and superiority.

5.2 Related Works

To the best of our knowledge, nearly no prior studies have noticed the role of the importance score order or leveraged the features with small scores during training. The most related works to us can be the feature selection algorithms that are partially supported by unselected features. For example, [19] first noticed the advantage of discarded features. In particular, they proposed a concept that the unselected features can be useful for classification tasks in a multi-task learning framework. Afterwards, [114] extended this idea to further improve the performance of predictive models. Nonetheless, both studies did not provide new feature selection methods but aimed to enhance the downstream learning performance. A recent work [131] leveraged the unselected features and designs a minimax optimization problem between selected and unselected features. However, it restricted by linear models and has complex training procedures, which makes it difficult to extend to other methods.

5.3 Methodology

As stated in the previous section, the main problem of the existing DL-based feature selection approaches is that the downstream neural network might be misled due to the

large-weighted candidate features during training. Therefore, our idea is to create a complementary feature mask that aims to penalize mistakenly down-weighted relevant candidate features during training.

5.3.1 Complementary Feature Mask

Definition 5 (Complementary Feature Mask) *$\tilde{\boldsymbol{m}}$ is the complementary feature mask of a given feature mask $\boldsymbol{m}$ if the ranking of elements in $\tilde{\boldsymbol{m}}$ is opposite to that of $\boldsymbol{m}$.*

Naturally, in practice, we expect that the weighted input data by the original feature mask as $\mathbf{X} \odot \boldsymbol{m}$ must have good predictive performance in the downstream neural network as usual, while the weighted input data by the Complementary Feature Mask (CFM) as $\mathbf{X} \odot \tilde{\boldsymbol{m}}$ should result in uncertain predictions during training. To have a better readability, the path using the original feature mask is called the *main* path, while the path using the complementary feature mask is called the *complementary* path. Accordingly, we can define the overall multi-task learning objective for the CFM-method as

$$\underset{\boldsymbol{\Theta}}{\operatorname{argmin}}\, \mathcal{L}_m(g(\boldsymbol{x} \odot \boldsymbol{m}), y) + \lambda \cdot \mathcal{R}(\boldsymbol{m}) + \gamma \cdot \mathcal{L}_c(g(\boldsymbol{x} \odot \tilde{\boldsymbol{m}})), \tag{5.1}$$

where $\boldsymbol{\Theta}$ denotes all trainable parameters within the CFM-method. In the learning objective above, the first two terms together are exactly the original learning objective for most DL-based FS algorithms defined in Chapter 2; i.e. $\mathcal{L}_m$ denotes a loss function for a given task such as regression or classification and $\mathcal{R}$ is a regularization term on the feature mask during training, which is not required by all methods. The third term $\mathcal{L}_c(\cdot)$ is a special loss function for the novel complementary path that measures how uncertain the predictions obtained by $\boldsymbol{x} \odot \tilde{\boldsymbol{m}}$ are. In other words, a more uncertain prediction in the complementary path corresponds to a smaller loss of $\mathcal{L}_c(\cdot)$, which is actually not considered by the main path (or many other existing DL-based FS approaches).

In total, minimizing the loss function defined in 5.1 is equivalent to searching for such a feature mask that leads to stronger predictive power in the main path, while

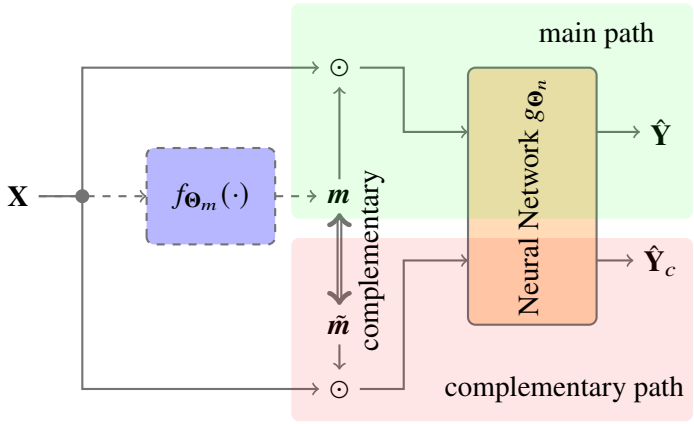


Figure 5.1: The proposed generic framework for feature selection using complementary feature mask for an auxiliary learning task.

forcing that the candidate features with the complementary importance scores to pose only uncertain predictions for a given learning task. Consequently, we expect that all candidate features can be scored in a correct order and really pivotal and representative candidate features can be weighted with higher values.

Since the designed overall learning objective indicates a multi-task learning procedure, this yields a two-path structure of the propose CFM-method as shown in Figure 5.1. The path "$\mathbf{X}$-$\boldsymbol{m}$-NN-$\hat{\mathbf{Y}}$" is called the main path, which is an ordinary DL-based FS structure as in many previous works [37, 69, 72] and introduced in Chapter 2. In addition to this, the complementary path in pink is attached to the $\tilde{\boldsymbol{m}}$ to force an uncertain prediction as a regularization during training.

Last but not least, obviously, the entire CFM-method can be understood as a generic framework with a twofold meaning. Firstly, it means that the main path can be implemented by many existing DL-based FS approaches if they follow the structure defined in Chapter 2, showing the flexibility of our design. Secondly, we do not restrict the design of $\tilde{\boldsymbol{m}}$ and $\mathcal{L}_c$, which can be directly specified according to available prior knowledge. Nevertheless, in the next section, we present one implementation by us. The corresponding experimental results also show that our implementation already achieved the state-of-the-art feature selection performance in the context of classification.

5.3.2 Implementation

CFM Design

In our FM-method, which has achieved superior feature selection performance in comparison with many contemporary FS approaches, the ordinary feature mask $\boldsymbol{m}$ is defined as:

$$\boldsymbol{m} = softmax\Big(\frac{1}{b}\sum_{i=1}^{b}\big(\mathbf{W}_2\tanh(\mathbf{W}_1\boldsymbol{x}_i + \boldsymbol{b}_1) + \boldsymbol{b}_2\big)\Big), \tag{5.2}$$

where b denotes the minibatch size during training and this equation defines the mapping between the minibatch input data $\mathbf{X}$ and feature mask $\boldsymbol{m}$ as $\boldsymbol{m} = f_{\boldsymbol{\Theta}_m}(\mathbf{X})$ with $\boldsymbol{\Theta}_m = \{\mathbf{W}_1, \mathbf{W}_2, \boldsymbol{b}_1, \boldsymbol{b}_2\}$ being the trainable parameters. It should be emphasized that in order to guarantee that the values of both the feature mask $\boldsymbol{m}$ and its complementary version $\tilde{\boldsymbol{m}}$ are of similar scale during training, we implement the complementary feature mask $\tilde{\boldsymbol{m}}$ as

$$\tilde{\boldsymbol{m}} = softmax\Big(-\frac{1}{b}\sum_{i=1}^{b}\big(\mathbf{W}_2\tanh(\mathbf{W}_1\boldsymbol{x}_i + \boldsymbol{b}_1) + \boldsymbol{b}_2\big)\Big). \tag{5.3}$$

It is clear to see that the only difference between $\boldsymbol{m}$ and $\tilde{\boldsymbol{m}}$ is that we use *softmax* to normalize the negative primitive feature mask. As a result, the ranking of the elements in $\tilde{\boldsymbol{m}}$ is exactly opposite to that of the ordinary feature mask $\boldsymbol{m}$ and meanwhile the values of both $\boldsymbol{m}$ and $\tilde{\boldsymbol{m}}$ are of similar scale as desired.

Architecture

Figure 5.2 shows our concrete architecture, where the most conspicuous design is that we use a multi-output architecture as $g_{\boldsymbol{\Theta}_n}(\cdot)$. More precisely, g_{fea} includes all layers from the first input layer to the penultimate layer acting as a shared deep feature extractor for the input data, while it is followed by two independent dense layers, one for the main path g_m and one for the complementary path g_c. This design ensures that the network maps $\boldsymbol{x} \odot \boldsymbol{m}$ and $\boldsymbol{x} \odot \tilde{\boldsymbol{m}}$ into the same feature space. This makes more sense for the requirement that the original feature mask should select features leading to better performance than those selected by the complementary feature mask.

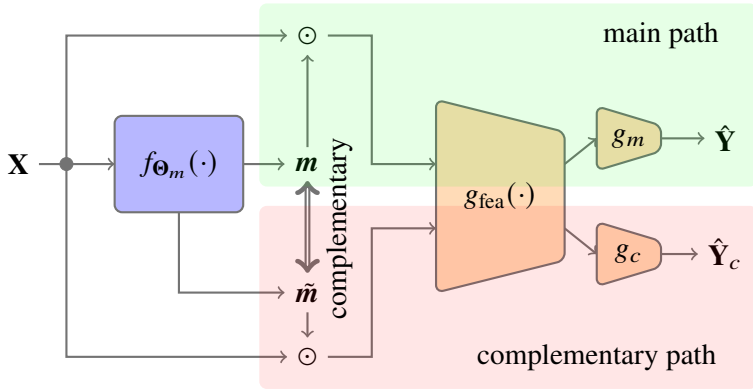


Figure 5.2: The architecture of our implementation for CFM.

The downstream neural network mentioned above was implemented as in Figure 5.3, which was inherited from the original FM-method presented in Chapter 3. The feature extraction network g_{fea} was composed of two hidden dense layers with 128 and 64 neurons respectively, a dropout layer with the rate of 0.3 following the two dense layers, and one output dense layer for classification. The activation function for both hidden layers was LeakyReLU [84] with $\alpha = 0.02$. For CFM, as shown in Figure 5.3, we concatenated two separate dense layers to the dropout layer to enable the multi-task learning structure. All output layers used the activation of *softmax* due to the classification task.

Learning Objective

Since the design of the complementary feature mask is inherited from the original FM-method, the proposed CFM-method can enjoy the neat property of being free from additional regularization terms on $\boldsymbol{m}$. As a result, the overall learning objective is simplified to

$$\underset{\boldsymbol{\Theta}}{\operatorname{argmin}}\, \mathcal{L}_m(g_m(g_{\text{fea}}(\boldsymbol{x} \odot \boldsymbol{m})), y) + \gamma \cdot \mathcal{L}_c(g_c(g_{\text{fea}}(\boldsymbol{x} \odot \tilde{\boldsymbol{m}}))), \tag{5.4}$$

where $\boldsymbol{\Theta}$ denotes all trainable parameters of the entire CFM-method. We first consider supervised classification tasks only for simplicity. Thereby, as presented in Chapter 3,

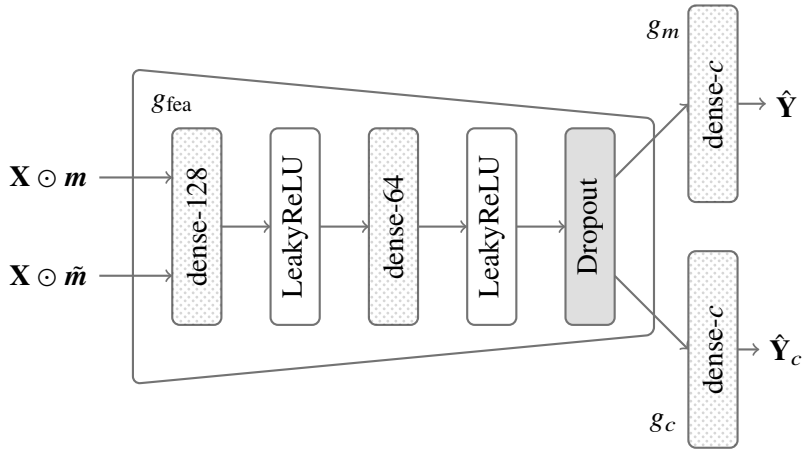


Figure 5.3: The architecture of the downstream learning network used for the proposed CFM-method. The downstream learning network has two independent output layers respectively for the main and complementary paths.

$\mathcal{L}_m$ can be a simple categorical cross entropy loss for most cases. However, $\mathcal{L}_c$ is more difficult to have a reasonable design, because there are many ways to force a poor predictive power, but not all poor solutions of the complementary path are related to a good choice of the feature mask $\boldsymbol{m}$ and its corresponding complementary feature mask $\tilde{\boldsymbol{m}}$. We suggest that it is not reliable to directly maximize the cross entropy loss between the prediction $\hat{\boldsymbol{y}}_c$ and the ground truth $\boldsymbol{y}$ to achieve a poor prediction performance because it can lead to trivial and biased solutions. For example, if the complementary path always outputs a certain but fixed class, this is a poor prediction but it does not improve the quality of the importance scores in $\boldsymbol{m}$. This is because in this case, the downstream learning network $g(\cdot)$ likely learns a trivial mapping that maps all inputs to a single class (a single output value), regardless whether $\boldsymbol{m}$ is correctly learned. On the contrary, as mentioned before, a more conscious interpretation of a poor predictive capability should be uncertain predictions for $\boldsymbol{x} \odot \tilde{\boldsymbol{m}}$; i.e. a given input sample can be randomly classified to any one of the known classes if it is element-wisely multiplied with $\tilde{\boldsymbol{m}}$. Hence, we define the loss $\mathcal{L}_c$ as

$$\mathcal{L}_c = -\frac{1}{n}\sum_{i=1}^{n} \boldsymbol{e}_{r,i}^T \cdot \log\Big(g_c(g_{\text{fea}}(\boldsymbol{x}_i \odot \tilde{\boldsymbol{m}}))\Big), \tag{5.5}$$

where $\boldsymbol{e}_{r,i}$ is the one-hot coding of a random class label for a training data point $\boldsymbol{x}_i$. Specifically, the random class label is sampled from a discrete uniform distribution $\mathcal{U}(1, c)$. As can be seen, minimizing the loss defined in 5.4 maximizes the uncertainty of the predictions, because the samples from the same class are forced to be predicted to different classes in the complementary path.

5.4 Evaluations

This section evaluates the feature selection performance of the proposed novel CFM framework, especially the quality of the feature importance order with high scores.

5.4.1 Setup

Settings Put simply, the general experimental settings followed those described in Chapter 3; i.e. we trained feature selection algorithms on a given dataset and selected the top-k features according to the learned feature mask. Subsequently, the selected k features were evaluated on a separate classifier. Finally, the resulting classification accuracy on the test set was used to indicate the feature selection quality. We repeated the experiments with five different random seeds and report the averaged results with deviation for a fair analysis.

Benchmarking Datasets This chapter uses the same benchmarking datasets as in Chapter 3, namely MNIST [66], Fashion-MNIST (fMNIST) [126], Isolet [29], PCMAC [65], Madelon [41] and Gisette [41]. An overview of the used benchmarking datasets is listed in Table 3.1.

Reference Method As presented in Chapter 3, our FM-method has achieved state-of-the-art feature selection performance in comparison with many contemporary approaches. Therefore, this chapter uses the FM-method only as the reference method in all conducted experiments. Similar to the previous chapters, to guarantee a fair

comparison and reliable analysis, both the FM- and our CFM-method used the same downstream neural network, except that the CFM-method had one additional complementary path. The hyperparameter γ of our method was chosen by a grid search from $\{0.001, 0.01, 0.1, 1, 10, 100\}$ on a separate validation set (10% of the training data).

Downstream Classifiers The previous chapters have shown that the features selected by most DL-based feature selection approaches can perform similarly on different downstream classifiers. This section follows the same settings as before so we consider the following downstream classifiers: i) Random Forest (RF) [16]; ii) Extremely Randomized Tree (ERT) [32]; iii) k-Nearest Neighbor (kNN) [39].

5.4.2 Main Results

As discussed in the previous sections, the introduction of the novel CFM-path is expected to learn a more reasonable feature mask, meaning that the importance scores better match the true but unknown importance of the candidate features[1]. To clearly show this property, the main experiment compared the CFM-method and the FM-method with respect to different feature subset sizes on the three downstream classifiers. Specifically, ρ is defined as the proportion of the selected features as $\rho = \frac{k}{D} \times 100\%$ with k denoting the number of selected features and D denoting the number of all candidate features. In this section, to have a comprehensive performance comparison, we considered 7 different values as $\rho \in \{1\%, 1.5\%, 2\%, 2.5\%, 5\%, 7.5\%, 10\%\}$. Clearly, we mainly focused on small proportion because feature selection is known to be more challenging when the subset size is small. Accordingly, for example, with $\rho = 1\%$ for the MNIST dataset, in total $0.01 \times 784 \approx 8$ features (pixels) were selected for training separate downstream classifiers for evaluation.

Figure 5.4 summarizes the experimental results on RF, ERT and kNN, respectively. In total, the proposed CFM-method outperforms the FM-method in 109 out of 126 cases,

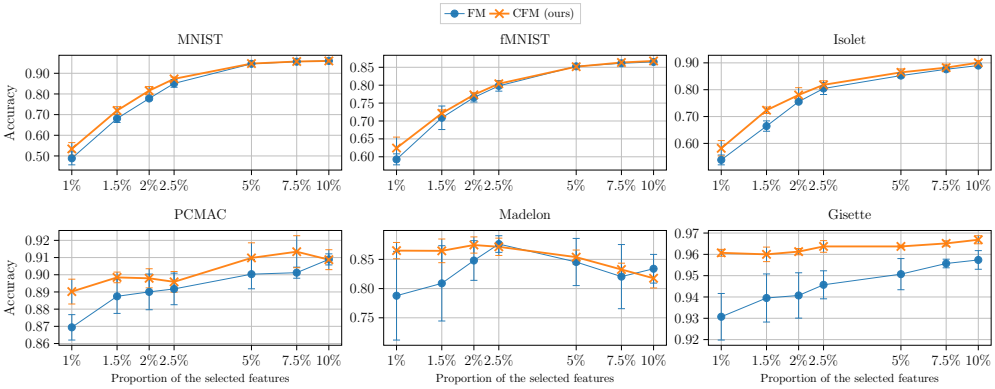
[1] Here we should recall that DL-based feature selection methods (including ours) often aim to approximate instead of guaranteeing the optimal solution of a corresponding exhaustive search. Therefore, we use the performance of the selected features on the downstream tasks to indirectly denote the quality of the selected features.

corresponding to a significantly better selection performance for different feature subset sizes and various downstream classifiers. This empirically confirms that the novel complementary path successfully regularized the learning procedure so that more representative and critical features were better scored.

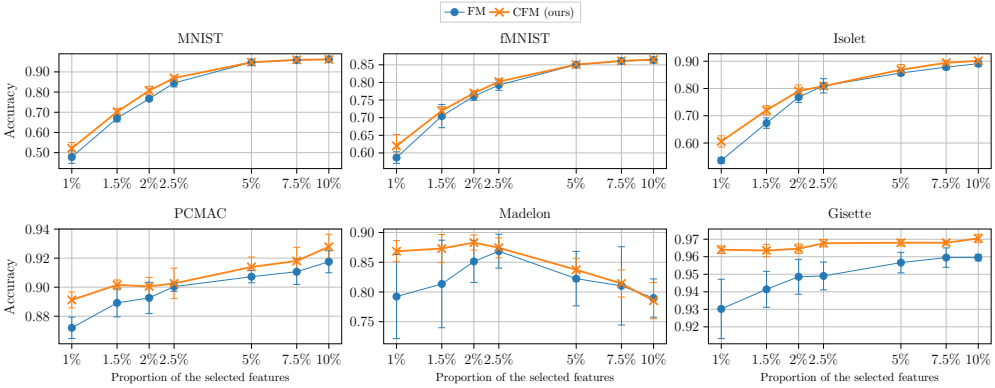
Analysis One important observation is that the CFM-method resulted in notably smaller deviations in accuracy than those of the FM-method for almost all cases. This can be easily seen especially for the datasets PCMAC, Madelon and Gisette. This experimental result suggests that the CFM-method has notably more stable selection results (smaller standard deviation values) under different initialization seeds of neural networks, which remains, however, a challenge for many other DL-based feature selection approaches. One feasible reason is that the novel complementary feature mask forces the main path to learn feature importance in a correct order as much as possible. This advantage is valuable because better stability in feature selection is extremely critical for understanding and analyzing high-dimensional data in different fields.

Another important observation is that our method generally performed extremely well for small feature subset sizes (i.e. small ρ), indicating that the learned feature mask really reflects the relative importance of each individual feature, especially for the highly-scored candidate features. On the contrary, the FM method without the regularization of the complementary path unfortunately cannot well maintain the correct importance order after training, although it can identify the overall important features.

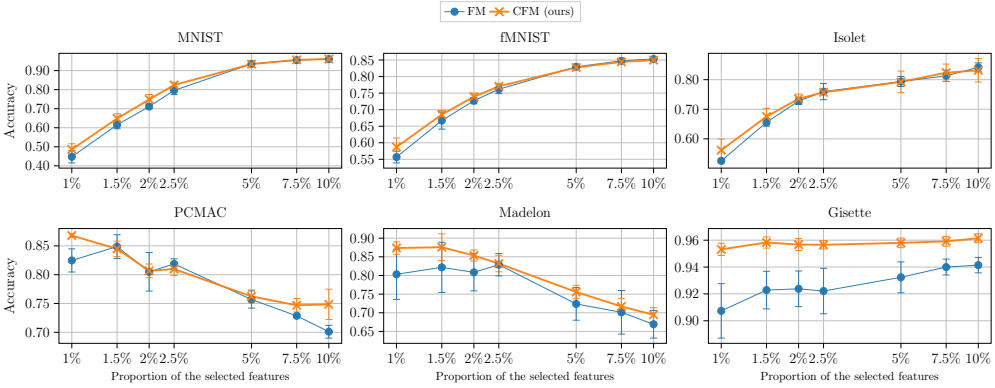
Interestingly, the resulting accuracy for the Madelon dataset decreased with increasing feature subset sizes using all three downstream classifiers for both the FM- and our CFM-method. This is different from other datasets for most cases. The main reason might be that the Madelon dataset was created with only 5 key features (ρ = 1%) that directly correspond to the class label, 15 additional redundant features that were linear combinations of the 5 key features, and 480 distractor features. Thereby, feature subsets with large sizes indicate more misleading features were included and the downstream classification performance can be thereby negatively affected.



(a) Classification accuracy on RF over different selected feature subset sizes.



(b) Classification accuracy on ERT over different selected feature subset sizes.



(c) Classification accuracy on kNN over different selected feature subset sizes.

Figure 5.4: Classification accuracy on the downstream classifiers.

Table 5.1: Comparison of feature selection results obtained by a separately trained random forest on a synthetic dataset for the original FM-method, the CFM-method and the ground truth.

	FM	ACCU	**CFM**	ACCU	**Ground Truth**	ACCU
top-1	x_1	0.335	x_3	**0.348**	x_3	0.348
top-2	x_1, x_2	0.409	x_3, x_1	**0.423**	x_3, x_1	0.423
top-3	x_1, x_2, x_3	0.553	x_3, x_1, x_2	0.553	x_3, x_1, x_2	0.553
top-4	x_1, x_2, x_3, x_4	0.995	x_1, x_2, x_3, x_4	0.995	x_1, x_2, x_3, x_4	0.995

Justification on a Toy Dataset In order to better understand when and why CFM performs better than FM, we created a toy dataset with ground truth for the relevant features. Specifically, five candidate features x_1 to x_5 were independently sampled from a uniform distribution $U(0, 1)$. For each sample vector of length 5, we check for conditions (C_1 to C_4): $C1 : \sin(x_1) < 0.5$, $C2 : \cos(x_2) > 0.85$, $C3 : x_3 > 0.5$ and $C4 : x_4 \notin [0.2, 0.8]$. The class label of the downstream classification task is defined as follows: If a sample meets i ($i \in 1, 2, 3, 4$) out of the four conditions above, this sample is assigned a class label i, otherwise this sample is assigned with a class label of 0. Thereby, we consider a five-class classification as our downstream task. Obviously, in this toy dataset, x_5 is irrelevant and the left four features are relevant. In total, we created 5000 training samples.

Table 5.1 shows the top-1 to top-4 selected features of the FM-method, CFM-method and the ground truth by exhaustive search with their corresponding prediction accuracy (ACCU). It should be noted that we respectively selected top-k features by different methods and used them to train a separate random forest for a 5-class classification task to obtain the prediction performance as the metric. In order to obtain the ground truth of the optimal feature subsets, we exhaustively evaluated all feature combinations for $k \in \{1, 2, 3, 4\}$ and the feature relevance was determined by their prediction accuracy. Although the FM-method can still correctly identify all relevant candidate features, the importance order of the most and the second most candidate features was not correct. On the contrary, the CFM-method successfully assigned the correct importance scores to all relevant features in comparison with the ground truth.

Visual Understanding Similar to Chapter 3, due to the nature of images, a visualization of the learned feature mask on both MNIST and fMNIST datasets can help us understand the selected features in a more intuitive way.

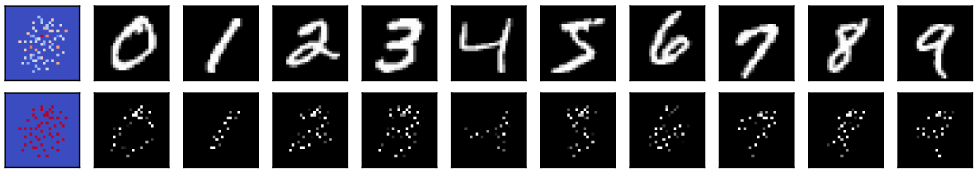
Figure 5.5 demonstrates the learned feature masks and the corresponding top-50 features by the CFM-method on the MNIST and the fMNIST datasets, respectively. In each subfigure, the most top left image shows the learned feature masks normalized into 0 and 1 for better visualization (blue indicates smaller values, while red indicates larger values). The most bottom left image is the top-50 features (red pixels) selected based on the learned feature masks. In addition, the first row from the second column shows exemplary images randomly selected from the ten classes, while the second row from the second column shows the selected 50 pixels correspondingly.

It can be seen that pixels in the center were mostly selected for the MNIST dataset, which makes sense, because digits are mostly located in the center of each image for the MNIST dataset. Furthermore, we can observe that even with the selected 50 pixels only, the digits were still recognizable, meaning that the critical structure of the input data was well maintained even after the selection procedure. The objects in the fMNIST dataset are typically large, so the learned most critical features are located more uniformly than those of MNIST. From the exemplary results of the fMNIST dataset, we can also observe that most critical and discriminative pixels of clothes, shoes and bags were maintained.

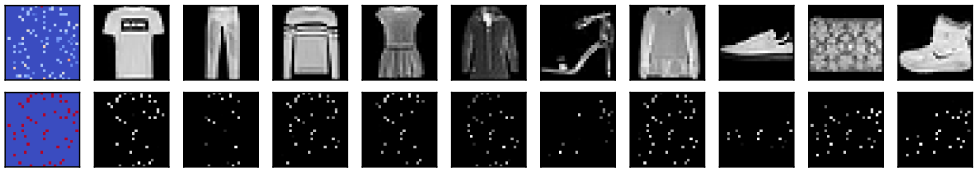
5.5 Design of the Complementary Feature Mask

In Section 5.3, we state that the CFM framework is generic and the concrete form of the complementary feature mask can be specified by users and practitioners, enabling a lot of flexibility for real-world applications.

However, according to our experiments, an improper design of complementary feature masks might lead to difficulty in training. To show this, we use the MNIST dataset as an example and visualize the feature masks to interpret how the CFM defined in Equation 5.3 works.

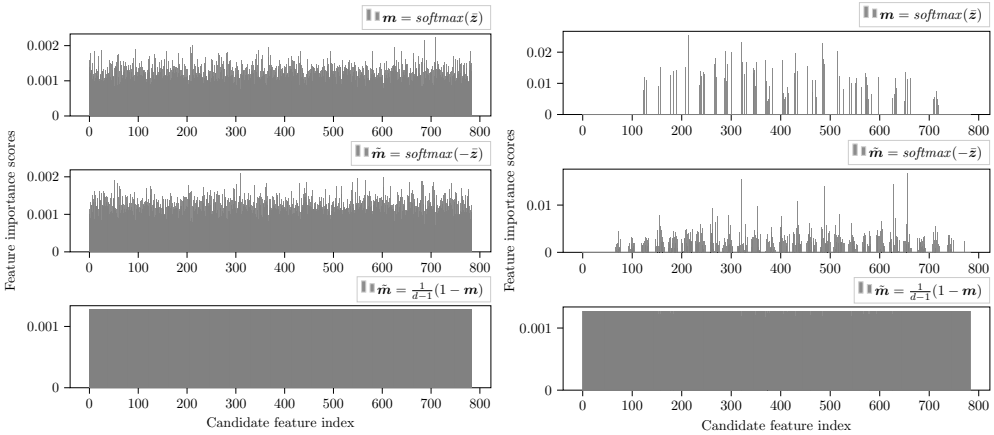


(a) Visualization for the MNIST dataset: the learned continuous-valued feature mask (*most top left*); the top-50 features based on the learned mask (*most bottom left*); exemplary images (*the top row from second column*); the selected top-50 pixels based w.r.t. each image above (*the bottom row from the second column*).



(b) Visualization for the fMNIST dataset: the learned continuous-valued feature mask (*most top left*); the top-50 features based on the learned mask (*most bottom left*); exemplary images (*the top row from second column*); the selected top-50 pixels based w.r.t. each image above (*the bottom row from the second column*).

Figure 5.5: The learned feature masks on the MNIST and fMNIST datasets.



(a) Initialization.

(b) After training.

Figure 5.6: Feature importance for different designs of complementary feature masks. The proposed complementary feature mask $\tilde{\boldsymbol{m}} = softmax(-\bar{\boldsymbol{z}})$ can reliably assign notably different scores, while maintaining similar value ranges as the original feature mask $\boldsymbol{m}$.

Figure 5.6 shows the values of each element in $\boldsymbol{m}$ and $\tilde{\boldsymbol{m}}$ after initialization (Figure 5.6a) and after training (Figure 5.6b), respectively. In each subfigure, the x-axis denotes different entries in $\boldsymbol{m}$ and $\tilde{\boldsymbol{m}}$ with the y-axis denoting the corresponding values. In particular, the feature mask $\boldsymbol{m}$ for the main path is defined as $\boldsymbol{m} = softmax(\bar{\boldsymbol{z}})$ and the complementary feature mask $\tilde{\boldsymbol{m}} = softmax(-\bar{\boldsymbol{z}})$. As can be seen from Figure 5.6a, in our implementation, $\boldsymbol{m}$ (*top*) and $\tilde{\boldsymbol{m}}$ (*middle*) have similar value ranges. This is meaningful for initialization, meaning that all candidate variables can be almost equally considered at the beginning of training without bias. Indeed, other designs such as $\tilde{\boldsymbol{m}} = \frac{1}{d-1}(1 - \boldsymbol{m})$ (*bottom*) can also lead to similar value ranges. This design also meets necessary properties: *i*) $\tilde{\boldsymbol{m}}_i \in (0, 1)$; *ii*) $\tilde{\boldsymbol{m}}_i$ has the opposite ranking as $\boldsymbol{m}_i$; *iii*) $\sum_{i=1}^{d} \frac{1}{d-1}(1 - \boldsymbol{m}_i) = 1$. However, we found that such a design led to indistinguishable importance scores (around 0.001) after training as shown in the bottom of Figure 5.6b and the scores were in a totally different value range than that of $\boldsymbol{m}$ (between around 0.01 and 0.02). On the contrary, our design can not only preserve similar value range for the complementary feature mask $\tilde{\boldsymbol{m}}$ shown in Figure 5.6b (*middle*) but also ensure discriminative complementary scores for individual candidate features.

In summary, our design has two benefits. Firstly, it allows similar value ranges for both $\boldsymbol{m}$ and $\tilde{\boldsymbol{m}}$ during initialization and training. This ensures that all candidate features were equally considered at the beginning of the training. Secondly, value differences between candidate features are obvious and forces the complementary path to focus on the features assigned with small scores by $\boldsymbol{m}$ during training. Despite the superior performance of the current design of the complementary feature mask, it requires future work to explore potentially better forms of $\tilde{\boldsymbol{m}}$.

5.6 Discussion

Actually, as described in the previous sections, the CFM is not only a specific methodology but also a generic framework for feature selection. In other words, the idea of using a complementary path to regularize the main path can be applied to many other existing DL-based feature selection approaches.

To briefly illustrate this, we provide the pseudo-codes of integrating a popular feature selection approach DFS [69] into the overall CFM framework. It should be noted that the DFS method does not contain a function $f_{\boldsymbol{\Theta}_m}(\cdot)$, meaning that the feature mask $\boldsymbol{m}$ for DFS can be considered as a trainable vector which is randomly initialized. Based on this, integrating DFS to the CFM framework is summarized in Algorithm 4. From the pseudo-codes, we can see that the CFM idea can be applied to existing DL-based FS method without much effort by simply adding a complementary path with the complementary loss $\mathcal{L}_c$. Meanwhile, the original loss functions remain the same. This is specially valuable for the use cases where some FS approaches have been already used and the users do not want to significantly change the overall architecture and learning procedure.

Algorithm 4 Applying CFM to DFS

Require: Training data pair $(\boldsymbol{x}, y)$, the original loss function for DFS $\mathcal{L}_{\text{DFS}}$
1: Randomly initialize the DFS with the network parameters $\boldsymbol{\Theta}$ and $\boldsymbol{m}$
2: **for** $i = 1$ to maximal training iterations **do**
3: Calculate the original loss as $\mathcal{L}_{\text{DFS}}(g_m(g_{\text{fea}}(\boldsymbol{x} \odot \boldsymbol{m})), y)$
4: Calculate complementary feature mask as $\tilde{\boldsymbol{m}} = softmax(-\boldsymbol{m})$
5: Calculate the complementary loss as $\mathcal{L}_c(g_c(g_{\text{fea}}(\boldsymbol{x} \odot \boldsymbol{m})))$
6: Update $\boldsymbol{\Theta}$ and $\boldsymbol{m}$ based on gradient $\nabla_{\boldsymbol{\Theta},\boldsymbol{m}} \mathcal{L}_{\text{DFS}} + \gamma \mathcal{L}_c$
7: **end for**

Secondly, the choice or design for the loss function in the complementary path can depend on a given use case. In this chapter, we used categorical cross entropy with respect to a random class label as the loss function to force the complementary feature mask to have uncertain predictive capability and achieved significantly better selection results than the state of the art. Nevertheless, we cannot exclude other possible implementations for loss functions. Some other intuitive ways can include maximizing the distance between the main and the complementary path, which is inspired from contrastive learning [23].

Finally, the complementary path is trained targeting an uncertain prediction. It may raise a question whether redundant relevant features can be selected. Firstly, in terms of the prediction performance in downstream tasks, redundant features are not notably harmful and are typically acceptable, if it is allowed to select a feature subset with a

moderately large size. In addition, as shown in Figure 5.5a and Figure 5.5b, we found that the proposed framework with CFM regularization actually did not select redundant features (neighboring pixels), meaning that the neighboring pixels were assigned with notably different importance scores. This observation indicates that our method can automatically down-weight some redundant candidate features during training.

5.7 Summary

In this chapter, we proposed a generic deep-learning-based feature selection framework based on a novel complementary feature mask. Based on a multi-task learning structure, CFM forced the candidate features with complementary importance to have as uncertain predictive ability as possible and thus enables a correct feature importance order in the main path. As a result, the selection quality of the main path was significantly improved over different selected feature subset sizes. Extensive experiments on six benchmarking datasets and comparison with a recent state-of-the-art method have shown the effectiveness and superiority of our method. In addition, we also demonstrated how to apply CFM to other existing approaches, showing that the CFM idea is generic. In total, this work is expected to inspire other fellow researchers to design new feature selection methods while considering the other side of the feature importance scores by using the complementary feature masks.

Chapter 6

Batch-Wise Attention for One-Class Classification

Chapter 2 gives us a brief overview of different research directions in feature selection. Chapter 3, Chapter 4 and Chapter 5 have presented our novel feature selection approaches based on the novel FM-module. This chapter aims to explore the potential of the FM-module beyond feature selection scenarios.

Literally, feature selection can be understood as to identify which candidate features are more critical, while which candidate features might be less important with respect to a given learning task. In this sense, as mentioned before, feature selection can be interpreted as a task to learn the feature importance.

If we take one step further and not restrict ourselves to a hard selection (i.e. either to maintain or to remove some candidate features), the feature mask or feature importance vector can be actually used as an indicator to tell the downstream neural network, to which parts of the input data it should pay more attention. To provide a more conscious understanding, we can see from Figure 6.1 that each image is covered by a heatmap that tells us that we should focus on the red parts because these parts of the inputs are more informative than other parts. Such a technique is called attention mechanism in deep learning, meaning that neural networks automatically learn to pay attention to certain areas of inputs.

[‡]Contributions presented in this chapter were partially reported and published in our previous works [71, 74, 79]. Some of the author's own formulations from [71, 74, 79] are adopted in this chapter.

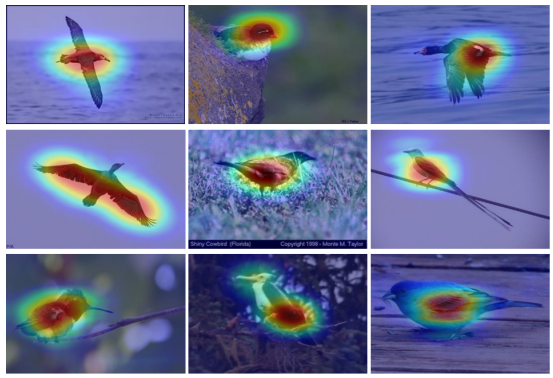


Figure 6.1: Exemplary images with a saliency map. The red parts are the critical area to make a correct detection and recognition. This further confirms our statement: Not all features are created equal. Figure source: [44].

Now, we go back to feature importance learning by feature selection techniques. It becomes attractive to investigate whether our FM-module can be considered as a special attention mechanism and used in some scenarios beyond feature selection? This chapter answers this question by presenting a use case of one-class classification.

6.1 Introduction

In the machine learning community, binary and multi-class classification are the most popular scenarios for various applications such as computer vision and language processing. In these scenarios, we often use the data affiliated with c classes ($c \geq 2$) to train a classifier. During deployment, we want to use the trained classifier to tell which of the c classes an unseen test sample belongs to. Obviously, an implicit requirement for this scenario is that the test sample must be affiliated with one of the given c known classes. Literally, such classifiers do not know what they do not know. For example, supposing that a classifier is trained to discriminate dogs from cats, this classifier still outputs a prediction of either a dog or a cat, even if we feed an image of a car to it.

On the contrary, One-Class Classification (OCC) aims to develop algorithms that can identify the data affiliated with unknown classes. In other words, an OCC algorithm is

expected to know what it does not know. More precisely, OCC algorithms are trained with the data from only one known class, while a trained OCC algorithm must be able to tell whether an unseen test sample belongs to the known class during inference. Based on this requirement, OCC is generally considered more challenging than the ordinary binary or multi-class classification tasks. Moreover, due to the nature of OCC, the resulting OCC algorithms are especially valuable for unsupervised or semi-supervised anomaly detection in wide applications, including but not limited to medical diagnosis [31, 124], fraud detection [5, 51] and concept drift detection [61, 130]. This is because in anomaly detection, we often have access to the normal data only, while the algorithm is expected to discriminate normal samples from abnormal samples or outliers during inference. This exactly matches the OCC scenarios.

Due to its importance, OCC has drawn increasing attention over the last years as shown in some surveys [54, 95, 109]. According to literature review, it is evident that nowadays many popular OCC approaches use autoencoder (AE) as a key backbone algorithm. Representative studies include [2, 11, 25, 35, 76, 80, 94, 96, 104–107], because autoencoders are designed for unsupervised learning and able to learn data-intrinsic structure. More concretely, these OCC approaches typically train an AE on the data from the one given known class only. Then, during inference, it is expected that a test sample affiliated with unknown classes leads to larger reconstruction errors than those of the data from the known class. In this way, test samples affiliated with the unknown classes can be rejected by comparing their reconstruction errors to a predefined threshold. Based on this idea, there are several main streams in designing OCC algorithms. One popular direction is to include adversarial training to minimize the reconstruction errors such as [11, 94, 96, 104]. Some previous works [106, 107] leveraged autoencoders to split the data from the known class into typical and atypical subsets and reformulated OCC into a binary classification problem. One other idea aims to apply additional constraints in the latent space learned by an AE [2, 35].

Despite the success of autoencoders as a backbone algorithm for the OCC problems in literature, we have observed some unexpected behavior of autoencoders in the context of one-class classification, which has not been noticed by prior works. Firstly, as described above, the canonical learning objective of AE is to maximize the similarity between the inputs and the outputs. However, we argue that the maximization of

the similarity (or equivalently, the minimization of the reconstruction errors) cannot guarantee a good OCC performance because an AE automatically tries to learn and preserve all raw information for high reconstruction quality. That is to say, AEs are not created for discriminating among classes. As a consequence, autoencoders may have undesirable generalization capability so that the data affiliated with unknown classes can be still well reconstructed, leading to indistinguishable reconstruction errors between known and unknown classes. Secondly, according to our experiments, we can observe that the OCC performance significantly drops during training, although the loss (i.e. the reconstruction errors) still decreases. One prior work [11] suggests that this phenomenon may be due to overfitting. However, our experiments show that no overfitting exists in such cases because the losses for both training and validation data still decrease, while the OCC performance already degrades instead of an expected increasing.

This chapter investigates the deep reason of the observed discrepancy between self-reconstruction and OCC performance of autoencoders in order to obtain a better understanding of the unexpected behavior of AE in the OCC problems. More importantly, based on the FM-module, this chapter introduces a Batch-Wise Attention (BWA) block between the encoder and decoder to regularize the generalization ability of autoencoders for unknown classes for enhacing the OCC capability.

In summary, the major contributions of this chapter are:

- We have discovered the undesirable generalization ability to unknown classes and performance degradation of autoencoders in the context of one-class classification;
- Comprehensive experiments have been conducted to explain the unexpected behavior described above;
- An effective solution is proposed by applying a novel batch-wise attention block to autoencoders;
- BWA can be considered as a plug-in module for existing autoencoder-based OCC algorithms and is easy to use;

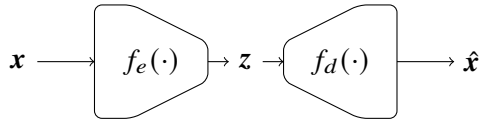


Figure 6.2: The structure of a vanilla autoencoder.

- Experiments on benchmarking datasets have shown the effectiveness of our method for OCC in comparison with the state-of-the-art approaches.

6.2 Background

6.2.1 Autoencoders for One-Class Classification

A vanilla autoencoder is composed of an encoder and a decoder. Let $f_e(\cdot;\boldsymbol{\Theta_e})$ denote the mapping of the encoder parameterized by $\boldsymbol{\Theta_e}$ and $f_d(\cdot;\boldsymbol{\Theta_d})$ be the mapping of the decoder parameterized by $\boldsymbol{\Theta_d}$. The dataset with n samples affiliated with a given one class is denoted as a matrix $\mathbf{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n]^T \in \mathbb{R}^{n\times d}$ with $\boldsymbol{x}_i$ being a d-dimensional column vector.

Based on the notations above, the encoder typically maps the input $\boldsymbol{x}$ into a lower-dimensional representation $\boldsymbol{z}$ as

$$\boldsymbol{z} = f_e(\boldsymbol{x};\boldsymbol{\Theta_e}), \tag{6.1}$$

where $\boldsymbol{z} \in \mathbb{R}^l$ with $l \ll d$. Likewise, the decoder transforms the latent representation $\boldsymbol{z}$ back to the original data space as

$$\hat{\boldsymbol{x}} = f_d(z;\boldsymbol{\Theta_d}), \tag{6.2}$$

where $\hat{\boldsymbol{x}}$ is called the reconstruction of the input sample $\boldsymbol{x}$. The learning objective of AE is to maximize the similarity between $\boldsymbol{x}$ and $\hat{\boldsymbol{x}}$. Hence, a canonical choice is to minimize the MSE loss as

$$\min_{\boldsymbol{\Theta}_e,\boldsymbol{\Theta_d}} \frac{1}{n}\sum_{i=1}^{n} ||\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i||_2^2. \tag{6.3}$$

Due to the bottleneck structure in AE, namely $l \ll d$, the intrinsic structure of training data are expected to be learned. This means that the test data only from the same distribution as the training data can be well reconstructed. During test, given an unseen test sample $\boldsymbol{x}_t$, we calculate its reconstruction error as:

$$\varepsilon_t = ||\boldsymbol{x}_t - \hat{\boldsymbol{x}}_t||_2^2 = ||\boldsymbol{x}_t - f_d(f_e(\boldsymbol{x}_t))||_2^2. \tag{6.4}$$

Correspondingly, with a predefined threshold ε, the known and unknown classes can be discriminated by

$$\boldsymbol{x}_t \in \begin{cases} \text{known class}, & \text{if } \varepsilon_t < \varepsilon \\ \text{unknown class}, & \text{otherwise} \end{cases} \tag{6.5}$$

Note that the threshold ε is typically defined based on the tolerable false negative rate (FNR)[1] by users or practitioners, meaning how many abnormal samples are acceptably allowed to escape from detection.

6.2.2 Undesirable Behavior of Autoencoders

Observation 1: OCC Performance Degradation This toy example considered digit 2 from the MNIST dataset as the given one known class with other digits being from unknown classes and excluded for training. Moreover, we compared autoencoders with three different latent dimensions, i.e. $l \in \{32, 64, 128\}$. Figure 6.3 presents the resulting area under the receiver operating characteristic curve (AUC) [15] on a validation set as well as the training and the validation mean squared errors. Apparently, the AUC notably decreased during training for all three cases, while both training and validation losses were still decreasing. In other words, the OCC performance already significantly degraded in spite of the fact that the AEs were not sufficiently trained in terms of the MSE losses. This observation suggests the discrepancy between minimizing reconstruction errors and obtaining the OCC capability.

[1] In the context of OCC, the known class is considered as negative class, while the unknown classes are considered as positive class.

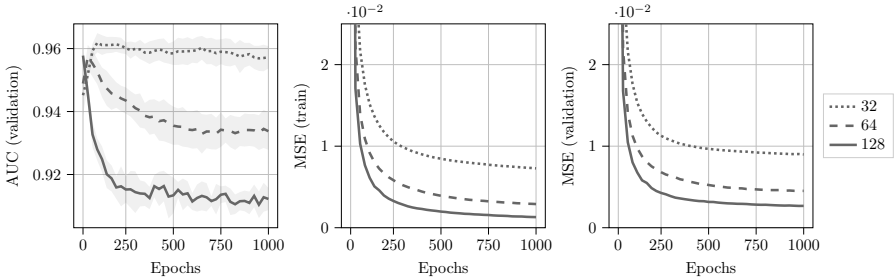


Figure 6.3: AUCs (*left*) degrade while train and validation losses (*middle* and *right*) decrease for different network capacities. This illustrates an unexpected behavior as follows. Although the autoencoder has not been fully trained due to the decreasing losses, the one-class classification performance already decreases.

Observation 2: Unreasonable Generalization Ability Figure 6.4 presents the reconstructed digits obtained by the aforementioned AEs. Although the AEs were trained on digit 2 only, other digits were still well reconstructed in most cases despite their absence during training. This indicates that AEs possess unreasonable generalization ability to totally unknown classes, which is, however, not desirable for the OCC problems. To further confirm this observation, as shown in Figure 6.5, flipped unknown digits (1^{st} *row*) and cracked digits (3^{rd} *row*) were respectively fed to the AE trained on digit 2 only with $l = 64$. Unfortunately, all unknown digits were well reconstructed (2^{nd} *row* and 4^{th} *row*). Furthermore, such undesirable generalization was observed for other datasets as well. Figure 6.6 shows the exemplary reconstructions of autoencoders with $l = 128$ trained on the class "Shirt" for Fashion-MNIST and the class "Frog" for CIFAR-10, respectively. The seventh column in Figure 6.6 shows the samples of the known class during training. It can be seen that almost all other classes can be well reconstructed (2^{nd} *and* 4^{th} *rows*), although the unknown classes were not accessible during training.

Comments The observations mentioned above clearly show an unexpected fact that a vanilla autoencoder does not learn discriminative latent features for a specific class but learns the features which can construct the entire input of all known and unknown classes. Taking digit images as an example, the basic building blocks (e.g. strokes and

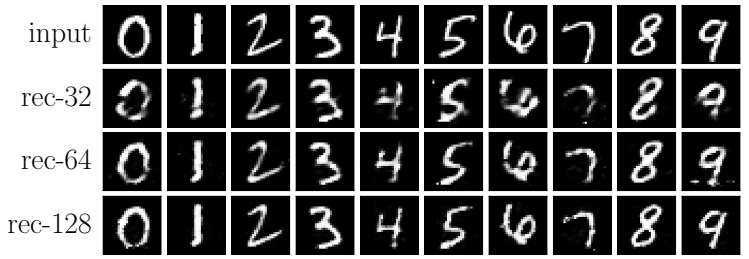


Figure 6.4: Reconstructions obtained by AEs trained on digit 2 only. *input*: the original images. *rec-32/64/128*: reconstructions with different latent dimensions.

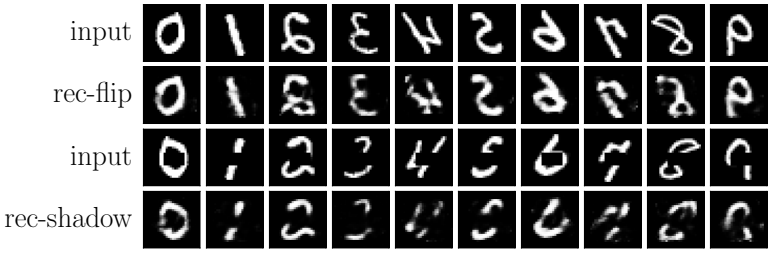


Figure 6.5: Flipped digits (1^{st} *row*) and cracked digits (3^{rd} *row*) were undesirably well reconstructed (2^{nd} *and* 4^{th} *rows*).

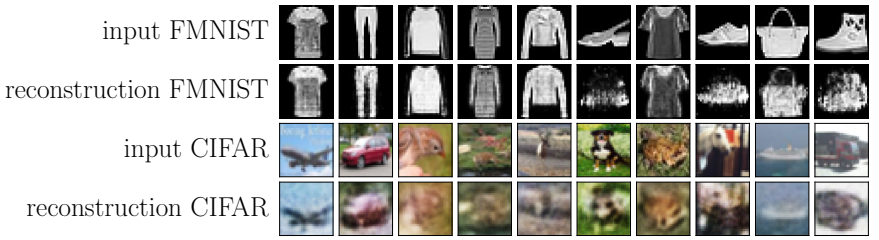


Figure 6.6: The unreasonable generalization ability are also observed on Fashion-MNIST and CIFAR-10. AEs were trained on one class only (7^{th} *column*) but other unknown classes were surprisingly well reconstructed.

circles), writing styles (the stroke width and shapes) as well as the composition rules are learned to reconstruct the input digit images as well as possible. More importantly, the experimental results suggest that digit 2 may already contain adequate information of reconstructing other digits as well, meaning that a vanilla autoencoder can possess unexpected generalization power to unknown digits as well. Therefore, this motivates us to consider an automatic mechanism that helps vanilla autoencoders to identify the most discriminative latent features during training in order to prevent such an undesired generalization capability.

6.3 Methodology

This section presents our solution to reduce the undesirable generalization ability of autoencoders in one-class classification problems.

6.3.1 The Batch-Wise Attention Block

One-class classification naturally assumes that the data belonging to the one given class share common critical latent features, while unknown classes' data can differ from certain latent features. Based on this well accepted assumption, the FM-module is a good candidate to ameliorate vanilla autoencoders in OCC, because it automatically learns the feature importance and thus provides a Batch-Wise Attention (BWA) during training. As can be seen from the previous section, it is critical to intelligently weight the latent dimensions learned by AEs because some latent features contribute to reconstructing the known class only, while other latent features may contain undesirable information for unknown classes. Consequently, the BWA block is naturally added between the encoder and decoder, i.e. in the latent space. Let $\boldsymbol{z}$ be the output of the encoder. Then, the batch-wise attention weight $\boldsymbol{w}$ is calculated as

$$\boldsymbol{w} = softmax\Big(\frac{1}{b}\sum_{i=1}^{b}\big(\mathbf{W}_2 \cdot \sigma(\mathbf{W}_1 \cdot \boldsymbol{z}_i + \boldsymbol{b}_1) + \boldsymbol{b}_2\big)\Big), \tag{6.6}$$

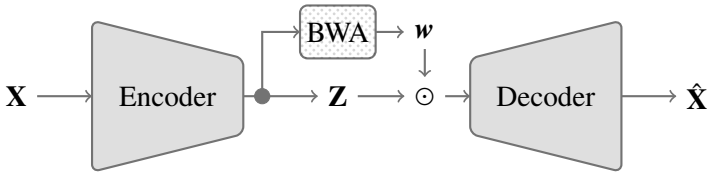


Figure 6.7: The structure of the proposed method based on the batch-wise attention (BWA) block.

where $\mathbf{\Theta}_b = \{\mathbf{W}_1, \boldsymbol{b}_1, \mathbf{W}_2, \boldsymbol{b}_2\}$ are trainable parameters of BWA and $\sigma(\cdot)$ is an activation function, having the same form as the original FM-module. Correspondingly, $\mathbf{W}_1 \in \mathbb{R}^{l' \times l}$, $W_2 \in \mathbb{R}^{l \times l'}$, $\boldsymbol{b}_1 \in \mathbb{R}^{l'}$ and $\boldsymbol{b}_2 \in \mathbb{R}^{l}$. As stated in the previous chapter, although BWA has the same form as the FM-module, BWA does not aim to perform a feature selection in latent space but to intelligently weight the latent dimensions. That is to say, it is expected that the discriminative latent features are assigned with larger scores, while misleading latent features are down-weighted. Obviously, this process can force the decoder to avoid equally considering all latent features for well reconstructions.

6.3.2 Structure and Learning Objective

The architecture of an AE with BWA is illustrated in Figure 6.7 and we call this combination BWA-AE. From the illustration, we can easily see that the BWA-block actually acts as a plug-in module in the latent space for a vanilla autoencoder. This indicates that our idea is generic: The BWA-block is able to be applied to different OCC-algorithms that rely on learning certain latent features.

More concretely, as described in Section 6.2, the encoder can be understood as a non-linear mapping approximated by a multi-layer neural network as $f_e(\cdot; \mathbf{\Theta}_e)$ parameterized by $\mathbf{\Theta}_e$ and the latent representation is defined as $\boldsymbol{z} = f_e(\boldsymbol{x}; \mathbf{\Theta}_e)$. Thereby, the weighted latent representation is defined as

$$\tilde{\boldsymbol{z}} = \boldsymbol{w} \odot \boldsymbol{z}, \tag{6.7}$$

where $\odot$ denotes the Hadamard product. The reconstruction is then obtained by

$$\hat{\boldsymbol{x}} = f_d(\tilde{\boldsymbol{z}}; \boldsymbol{\Theta}_d) = f_d(\boldsymbol{w} \odot \boldsymbol{z}; \boldsymbol{\Theta}_d). \tag{6.8}$$

Finally, the entire training and inference process follows the one for a vanilla AE in the OCC problems as introduced in Section 6.2.

This also reveals another advantage of our proposed BWA-AE: We do not change anything in training procedure and avoid introducing additional loss terms, making our approach easy to use. Specifically, we follow the same learning objective as a vanilla autoencoder for OCC as in Section 6.2. That is to say, the BWA-AE is trained by minimizing the MSE loss as

$$\underset{\boldsymbol{\Theta}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} ||\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i||_2^2 = \frac{1}{n} \sum_{i=1}^{n} ||\boldsymbol{x}_i - f_d(\boldsymbol{w} \odot f_e(\boldsymbol{x}_i))||_2^2, \tag{6.9}$$

where $\boldsymbol{\Theta} = \{\boldsymbol{\Theta}_e, \boldsymbol{\Theta}_d, \boldsymbol{\Theta}_b\}$ consists of all trainable parameters of the encoder, decoder and BWA block (defined in Eq. 6.6).

6.4 Evaluations

This section evaluates the OCC performance of our novel BWA-AE on two benchmarking datasets with the following three focuses:

- Can BWA-AE alleviate the performance degradation during training?
- Does BWA-AE yield better OCC results than a vanilla AE?
- Can BWA-AE outperform the contemporary and state-of-the-art DL-based OCC approaches?

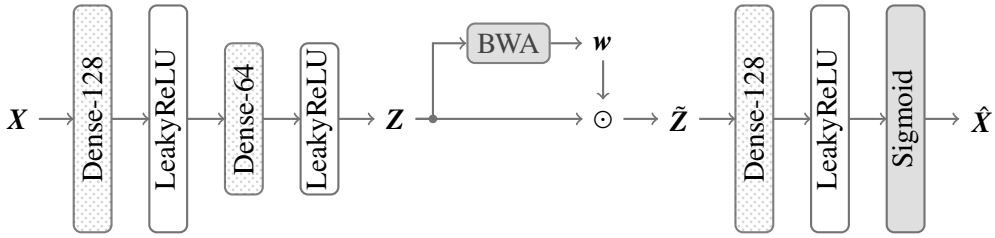


Figure 6.8: The architecture of the BWA-AE for the MNIST dataset.

6.4.1 Setup

Settings In each individual experiment, the data from one known class only were used for training different OCC algorithms. During test, the data from both known and unknown classes were fed to the trained algorithms to evaluate their OCC performance. We repeated each experiment for ten times and this section reports the averaged results with standard deviations only. AUC was used as the evaluation metric in order to be consistent with prior works.

Benchmarking Datasets The MNIST [66] and CIFAR10 [62] datasets were selected to benchmark different OCC algorithms because they were frequently used in the prior studies such as [2, 94, 103]. Although both MNIST and CIFAR10 originally consist of 10 different classes (i.e. designed for multi-class classification), in each experiment, we used one class only to train OCC algorithms and the rest nine classes were considered as unknown classes to be used in test only.

Implementation On MNIST, as shown in Figure 6.8, BWA was applied to an AE of 3 hidden dense layers with 128, 64 and 128 neurons, respectively. The CIFAR10 dataset is more complex so we used an AE of 5 hidden dense layers with 512, 512, 256, 512 and 512 neurons, respectively. Thereby, the BWA-block was applied to the output of the hidden layer of 256 neurons, which was considered as the latent representation. As a rule of thumb, $l' = l/2$ for all cases in the BWA block. All hidden layers used a LeakyReLU [84] with $\alpha = 0.02$ as the activation function. The output layer of all networks used *sigmoid* to guarantee that the values of reconstructions to be bounded

between 0 and 1. Adam [56] with a learning rate of 0.001 was used to optimize BWA-AE. Inspired by [13], we used structural similarity [123] as the distances between the original and reconstructed data to determine the affiliation of test samples during evaluation.

6.4.2 Main Results

Performance Degradation Figure 6.9 presents the recorded AUCs of both a vanilla AE and our novel BWA-AE on the MNIST dataset as an example. Firstly, our approach (*black solid line*) indeed resulted in stable training curves for all ten cases; i.e. there was no performance degradation anymore for BWA-AE, matching our expectation and confirming the success of applying the BWA block in the latent space. Secondly, we also observe that the performance degradation generally existed in AE (*gray dashed line*) for different cases (i.e. considering different digits as the only one known class), which was, however, rarely discussed in literature. For example, for digit 8, there was a dramatic drop in AUC during training, which was significantly more serious than other digits. We suggest that the basic building blocks of digit 8 can be understood as two small circles or two round tiny objects. Unfortunately, this pattern may occur in many different digits such as digits 3, 6 and 9. In rare cases, even the digit 2 and digit 5 can have similar shape to digit 8. The significant overlapping of basic building blocks of digit 8 and other digits may lead to the performance degradation of a vanilla AE in this case. In comparison with these highly similar digits, the main characteristic of digit 8 should be two *closed* circles, which seems not be captured by the vanilla autoencoders, while our approach successfully learned such latent feature. Moreover, interestingly, we can see that a vanilla AE typically led to more significant deviations (large standard deviation) during training, while our approach showed a neat convergence property with extremely small deviations. This shows that BWA-AE also provides a stable training and is resistant to different neural network initialization.

OCC Ability Enhancement Figure 6.9 additionally shows that the novel BWA-AE outperformed the vanilla AE with notable margin in all cases on the MNIST dataset. To further confirm this, Table 6.1, Table 6.2 and Table 6.3 summarized the AUC

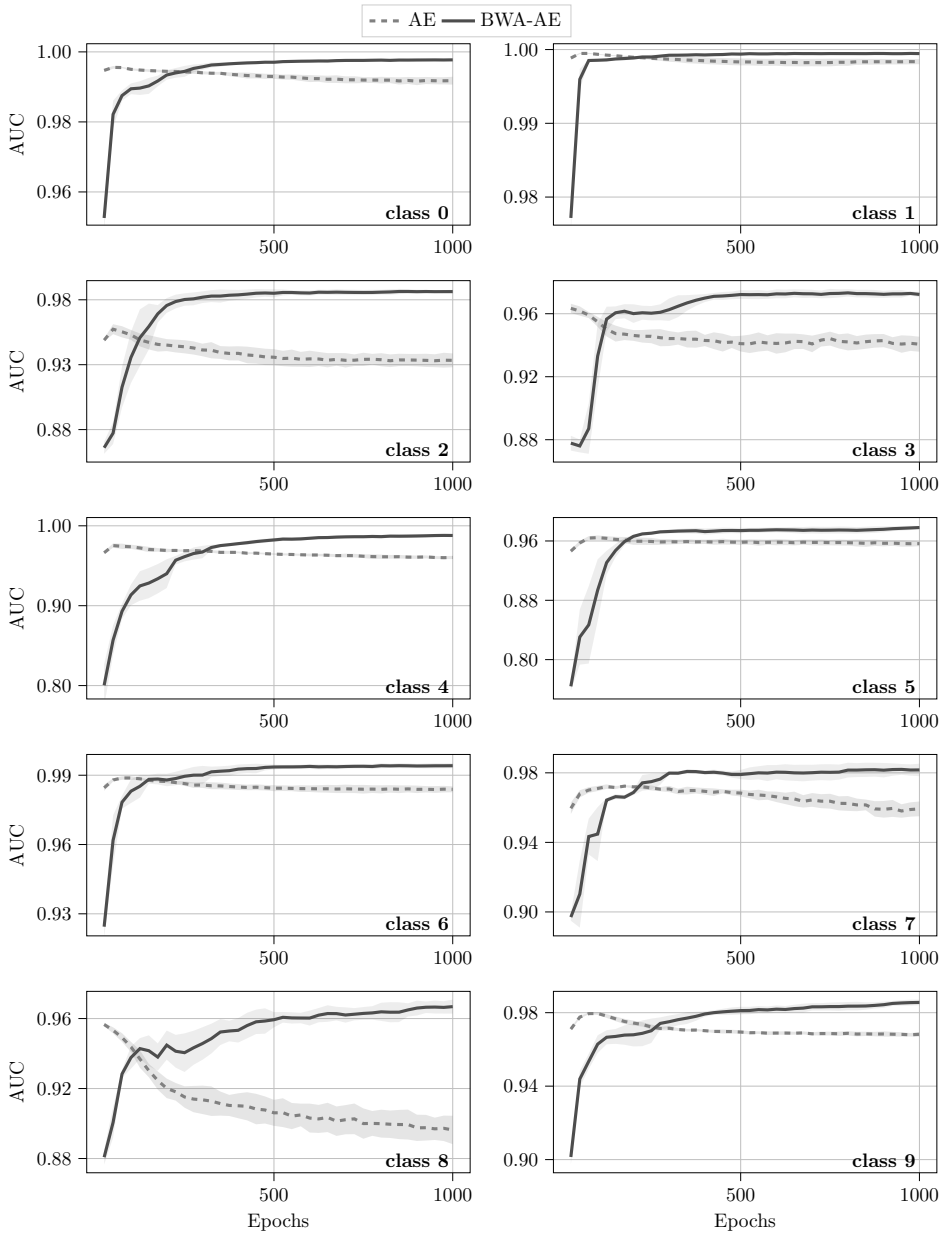


Figure 6.9: Performance degradation (*gray dashed line*) of a vanilla autoencoder is significantly reduced by leveraging the proposed novel BWA-AE framework (*black solid line*).

comparison between the BWA-AE and the vanilla AE on both the MNIST and the CIFAR10 datasets. Without surprise, our novel approach outperforms the vanilla AE in all cases. It is worth noting that on the more challenging CIFAR10 dataset, BWA-AE achieved a significant improvement of about 13.5% in terms of AUC averaged over all ten cases. This dramatic enhancement in OCC performance indirectly confirms that the BWA-block indeed correctly weights different latent dimensions for better OCC capability.

Table 6.1: Part 1: Performance comparison with vanilla autoencoders.

		class 0	class 1	class 2	class 3	class 4
MNIST	AE	0.995 (± 0.001)	0.999 (± 0.000)	0.945 (± 0.002)	0.948 (± 0.004)	0.969 (± 0.001)
	Ours	0.999 (± 0.000)	0.999 (± 0.000)	0.986 (± 0.001)	0.973 (± 0.003)	0.988 (± 0.002)
CIFAR10	AE	0.726 (± 0.002)	0.600 (± 0.005)	0.607 (± 0.002)	0.615 (± 0.003)	0.557 (± 0.001)
	Ours	0.830 (± 0.003)	0.715 (± 0.004)	0.674 (± 0.006)	0.620 (± 0.001)	0.698 (± 0.002)

Table 6.2: Part 2: Performance comparison with vanilla autoencoders.

		class 5	class 6	class 7	class 8	class 9
MNIST	AE	0.964 (± 0.003)	0.988 (± 0.001)	0.974 (± 0.002)	0.931 (± 0.004)	0.977 (± 0.001)
	Ours	0.981 (± 0.003)	0.994 (± 0.001)	0.987 (± 0.005)	0.967 (± 0.004)	0.988 (± 0.001)
CIFAR10	AE	0.707 (± 0.002)	0.484 (± 0.004)	0.669 (± 0.002)	0.776 (± 0.002)	0.513 (± 0.003)
	Ours	0.724 (± 0.003)	0.659 (± 0.001)	0.709 (± 0.002)	0.822 (± 0.004)	0.644 (± 0.007)

Table 6.3: Part 3: Performance comparison with vanilla autoencoders averaged over all classes.

	AE	Ours	Enhancement
MNIST	0.969	0.986	1.8%↑
CIFAR10	0.625	0.710	13.5%↑

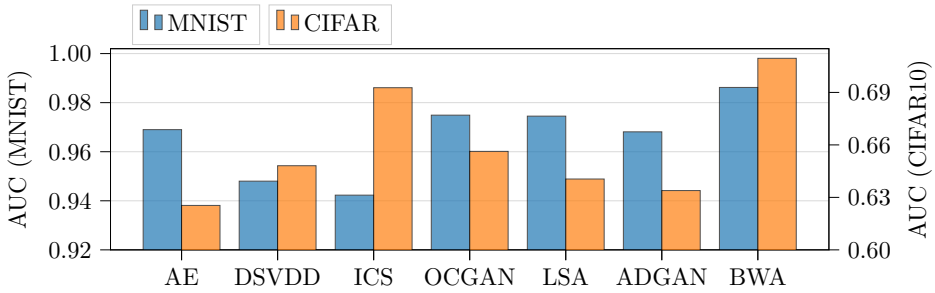


Figure 6.10: OCC performance comparison between our method (BWA) and other state-of-the-art OCC methods. BWA outperforms other methods on both datasets with a notable margin.

Comparison with SOTA Finally, we compared the BWA-AE with several contemporary or state-of-the-art OCC approaches in literature. In particular, we include: *i*) Deep Support Domain Description (DSVDD) [103]; *ii*) Intra-Class Splitting (ICS) [106]; *iii*) One-Class Generative Adversarial Network (OCGAN) [94]; *iv*) Latent Space Autoregression (LSA) [2]; *v*) Anomaly Detection Using Generative Adversarial Networks (ADGAN) [26]. All the above mentioned reference methods were evaluated on the same benchmarking datasets and thus easy for us to compare. Figure 6.10 presents the averaged AUCs over the ten classes on the MNIST and CIFAR10 datasets, respectively. Based on the results, we surprisingly found that the simple integration of the BWA-block to an AE already yields better performance than the other contemporary approaches, indicating the importance of intelligent weighting in latent dimensions. Furthermore, it is worth highlighting that our approach only used shallow simple dense layers for the implementation, while some other reference methods used complex and modern neural architectures including deep convolutional neural networks or residual connections. Thereby, we expect that our approach can achieve an even better performance by leveraging modern architectures and we leave it as a future work.

6.5 Insights into Batch-Wise Attention

Despite the success of BWA-AE in the OCC problems comparing to other reference methods, this section goes one step further and investigate the properties of BWA-AE

with the following aspects:

- Whether BWA just equals to an AE with small latent dimensions?
- Can we show that BWA really helps an AE to focus on the class-discriminative latent features only?
- Is our BWA-block usable for autoencoders with different capacities?
- How does batch size play a role in training BWA-AE?

6.5.1 BWA-AE is Not AE with Small Latent Dimensions

Taking digit 2 as the only one known class as an example, we recorded how the weighting vector $\boldsymbol{w}$ behaved during the training procedure. Figure 6.11 plots the values of all entries of $\boldsymbol{w}$ right after initialization (*top*), during training (*middle*) and after training (*bottom*) for the case of $l = 64$. Apparently, only 12 out of 64 latent features were finally assigned with significantly large weights by the jointly learned $\boldsymbol{w}$, while the rest latent features were clearly down-weighted. Visually, it seems that only the 12 latent features with high importance scores were really relevant for OCC. This observation raises a question: Is BWA-AE just equal to a vanilla autoencoder with small latent dimensions?

We first give a definition below:

Definition 6 (Effective Latent Dimension) *The latent dimensions are effective only if the corresponding learned importance scores are significantly larger (e.g. 100 times) than at least one of the rest dimensions, or all dimensions are almost equally important (e.g. differences are within the same order of magnitude).*

It should be noted that this is not a rigorous definition and the motivation is to enable better readability for the following discussions. According to this definition, for example, Figure 6.11 shows that the BWA-AE finally learned 12 effective latent dimensions, notated as $l_e = 12$. Then, in order to answer the aforementioned question, we

can design an iterative experiment as follows with the superscript being the iteration index:

Require: Let $l^{(1)} = 64$ and $i = 1$ be the start iteration index
while True **do**
  Respectively train AE and BWA-AE with $l^{(i)}$ latent dimensions
  Record AUCs for both AE and BWA-AE
  Identify $l_e^{(i)}$ of BWA-AE according to Definition 6
  if $l^{(i)} = l_e^{(i)}$ **then**
    End experiment
    break
  else
    $l^{(i+1)} \leftarrow l_e^{(i)}$
    i ← i+1
  end if
end while

Figure 6.12 shows the overall results of the iterative experiment described above. Apparently, BWA-AE outperformed the corresponding AE with small latent dimensions with clear margin. For example, with $l = \{9, 12, 64\}$, BWA-AE resulted in AUCs of $0.987 \sim 0.988$, while the vanilla AE with the same latent dimensions achieved AUCs with only about $0.945 \sim 0.966$. This clearly confirms that applying the novel batch-wise attention block to autoencoders does not equal to an AE with small latent dimensions. Furthermore, it is also interesting to see that our BWA-AE provided consistent AUCs over different l, while a vanilla AE was extremely sensitive to the choice of latent dimensions. Moreover, Figure 6.13 illustrates the learned $\boldsymbol{w}$ for the cases of $l = 12$ and $l = 9$. Obviously, for our experiment on digit 2, the minimal effective latent dimensions should be nine.

Nevertheless, it is still worthy studying the cases if we keep on decreasing l. In Figure 6.12, we can see two cases for $l = 2$ and $l = 5$, respectively. Even in such cases, BWA-AE still outperformed a vanilla AE with the same latent dimensions, indicating the effectiveness of the batch-wise attention. More interestingly, for an impractical case $l = 2$, BWA-AE still showed much better performance than most cases for AE. This suggests that the intelligent weighting in latent space exactly contribute to OCC performance.

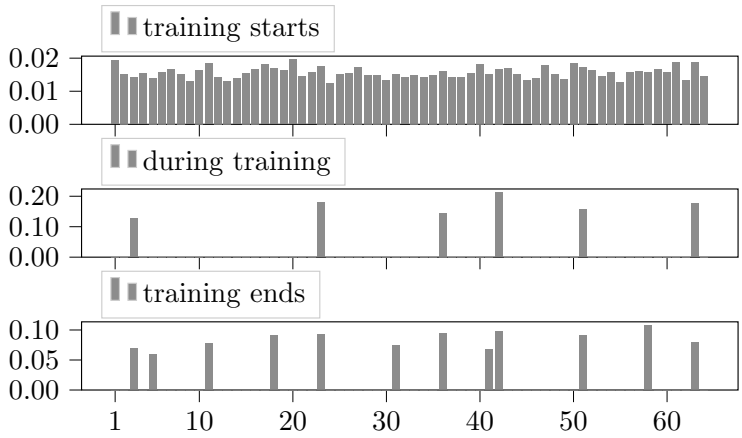


Figure 6.11: The values of all elements of the weighting vector $\boldsymbol{w}$ at the three different phases of training our BWA model. After training, only 12 of 64 latent dimensions were assigned with large weights. x-axis denotes the index of the element in $\boldsymbol{w}$.

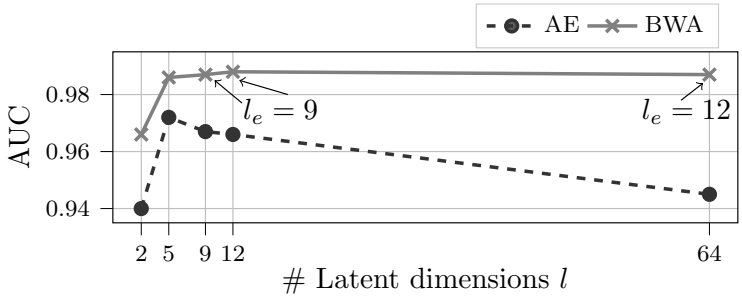


Figure 6.12: BWA always outperforms a vanilla autoencoder with the same latent dimension. This indicates that the training of the BWA model does not equal to training an autoencoder with small latent dimensions.

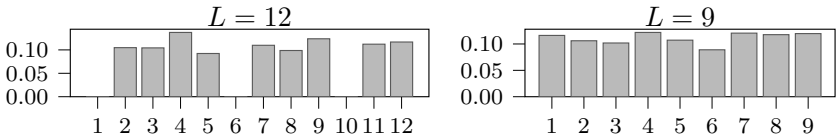


Figure 6.13: The learned $\boldsymbol{w}$ for BWA with 12 latent dimensions (*left*) and 9 latent dimensions (*right*), respectively.

6.5.2 Reconstruction

Section 6.2 introduces an observation that a vanilla autoencoder seems to learn basic building blocks for the entire data without focusing on the most class-discriminative latent features, because all unknown digits were also well reconstructed. In order to have a visual proof whether our BWA-AE approach can really correctly weight the latent features, we show the reconstructed digits by our BWA-AE for the ten cases on the MNIST dataset in Figure 6.14, where each row named with "digit x" indicates that BWA-AE was trained on digit x only and the first row presents the unseen digit images from a separate test set that were fed into the trained models as inputs, respectively. Clearly differing from a vanilla autoencoder (see Figure 6.4 as an example), BWA-AE tends to reconstruct all different inputs into the same class. For example, if BWA was trained on digit 2 (i.e. the 4^{th}-row in Figure 6.14), all input images were reconstructed into some kind of digit 2. For example, if digit 1 was given, the reconstruction was a digit 2 but looked like a digit 1 (4^{th} row and 2^{nd} column); if digit 7 was given, the reconstruction was again a digit 2 but looked like a digit 7 (4^{th} row and 8^{th} column). This observation holds for all ten cases, suggesting that BWA really focused on the class-discriminative basic building blocks as expected.

This property is desirable for OCC because in this way the difference between the reconstructions and the inputs becomes large for unknown classes but limited for the known class, explaining how BWA-AE significantly outperformed other modern state-of-the-art OCC approaches. Overall, the experimental results above imply that the novel batch-wise attention helps autoencoders to regularize its generalization ability and to learn more class-discriminative latent features despite the absence of unknown classes' data during training.

6.5.3 BWA for Different Capacities

One major drawback of vanilla autoencoders is that the OCC performance is sensitive to network capacities. For example, Figure 6.3 shows that large capacity led to significant performance degradation, while small capacity had less degradation. According to our experiments, BWA-AE worked well with different capacities as shown in Figure 6.15.

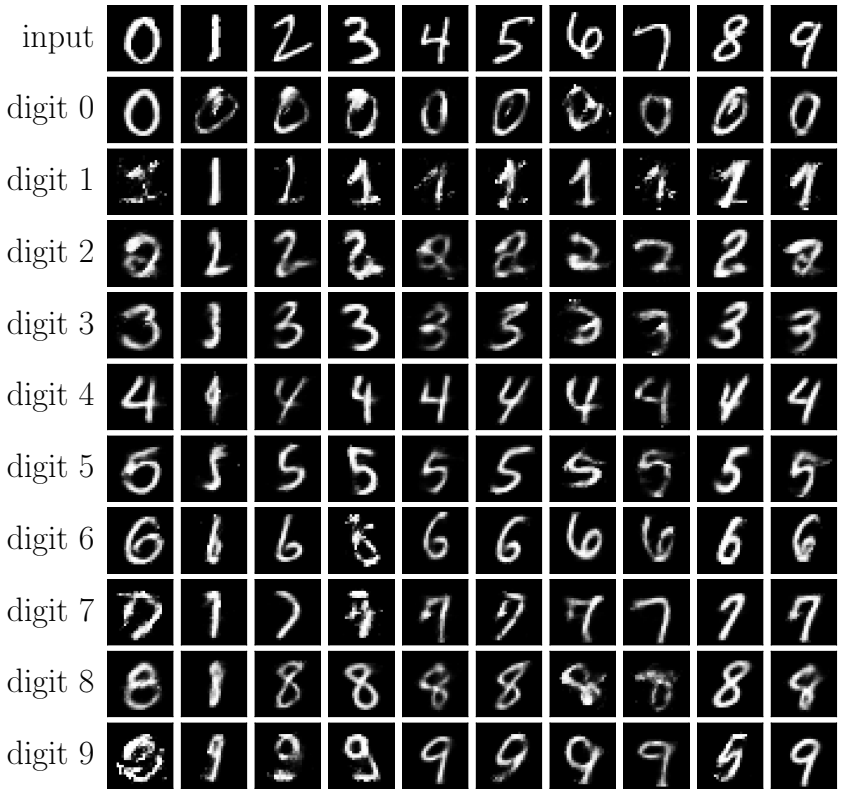


Figure 6.14: *input*: The original test images. *digit x*: Reconstructions obtained by BWA trained on digit x only.

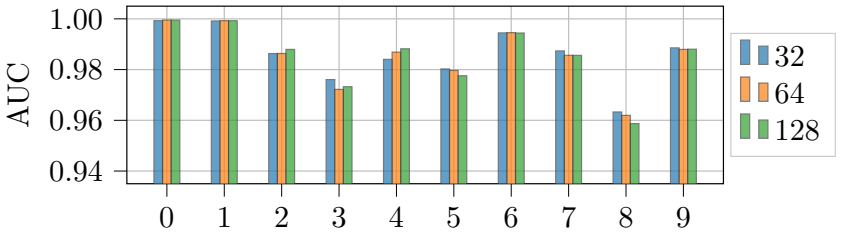


Figure 6.15: AUCs obtained by the proposed BWA w.r.t. three different network capacities ($l \in \{32, 64, 128\}$) for ten cases on MNIST. Generally speaking, in contrast to vanilla autoencoders, BWA works reliably for different network capacities.

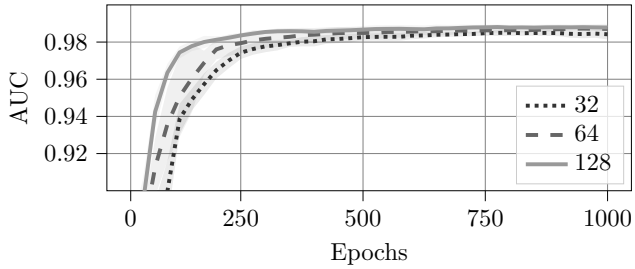


Figure 6.16: AUCs of BWA during training w.r.t different network capacities ($l \in \{32, 64, 128\}$) for digit 2 as the known class only. No performance degradation exists for all cases.

Here we trained BWA-AE with three capacities, i.e. $l \in \{32, 64, 128\}$. BWA-AE showed similar performance in terms of AUC for all ten cases. Moreover, Figure 6.16 additionally presents the AUCs during training for digit 2 as the known class as an example. In comparison with Figure 6.3, BWA had a stable training behavior with no performance degradation. This is valuable for using BWA-AE on new datasets without enough prior knowledge and the practitioners can easily have a good start point for performing OCC on new datasets.

6.5.4 Sensitivity to Batch Size

As its name shows, our novel batch-wise attention learns the feature importance scores based on minibatches during each training iteration. Naturally, the batch size should be critical for the final OCC performance. Specifically, we evaluated BWA-AE on the MNIST dataset using 8 different batch sizes with $b \in \{1, 2, 4, 64, 128, 256, 512, 1024\}$, including both extremely small and large batch sizes. The resulting AUCs averaged over the ten cases are shown in Figure 6.17. Generally speaking, batch sizes from a common choice (i.e. $b = 64$) led to similar and high OCC performance. In contrast, extremely small batch sizes such as $b = 1, 2, 4$ led to slightly worse performance. This makes sense because the feature importance learned from an extremely small batch size is not representative enough and might be easily misled due to potential outliers or noise in the data. On the other hand, extremely large batch sizes cannot provide further performance enhancement. One feasible reason is that overly large batch sizes lose

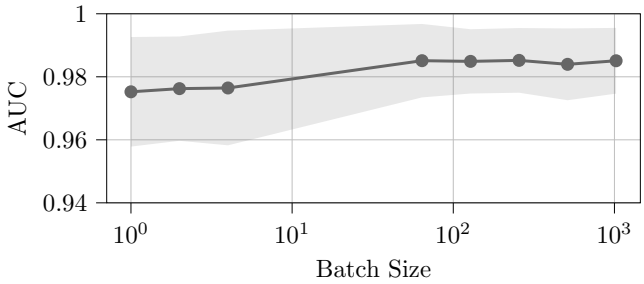


Figure 6.17: The AUCs w.r.t. different batch sizes averaged over the ten cases on MNIST. BWA works well for typical batch sizes (e.g. starting from 64) with consistent performance. Although the performance with extremely small batch sizes was slightly less satisfied, it was still comparable to other considered approaches. Note that the x-axis is in log scale.

the advantage of using mini-batch gradient descent algorithms, meaning that training might more easily get stuck to some local minima. Nevertheless, it is empirically shown that BWA-AE works well with common batch sizes and are not sensitive to large batch sizes.

6.6 Case Study: Resistive Open Defect Identification

The previous sections have shown the effectiveness of the proposed BWA-AE on the two popular benchmarking datasets in comparison with several contemporary approaches for one-class classification. This section targets a case study for a real-world application, namely unsupervised resistive open defect identification for combinational circuits.

6.6.1 Background

The past decades have witnessed the rapid growth in the semiconductor industry, with applications in various aspects of daily life, from smart phones and personal computers to autonomous driving and Internet of Things (IoT). In order to meet the dramatically increasing demand of semiconductor products, testing has been of great interest for both academic and industrial communities [43, 48]. In particular, effectively and efficiently

testing devices at early stages can help producers determine whether the manufactured devices are within the given specifications as well as promptly discover potential design flaws and systematic faults in order to reduce costs and time consumption. Furthermore, testing is also required for high reliability in the subsequent mass production. Among different levels of testing, one important case is to detect resistive open defect because resistive open defects often degrade already during the early stage of the lifetime for circuits and the degradation can soon lead to catastrophic and irrecoverable faults for the devices or even the entire systems.

To efficiently identify the defects, recent studies such as [90, 91] have introduced data-driven approaches to detect resistive open defects in combinational circuits with process variation in presence. In particular, they trained supervised classifiers using the delays (obtained under a few different voltages) of both defective and defect-free circuits. In other words, they considered it as a binary classification task. However, this may raise a twofold concern. Firstly, in practice, we cannot easily collect all possible defect types, which is time consuming and costly. Secondly, supervised classification algorithms typically can work well with balanced datasets, which might be difficult in the real world because defective devices are generally rare.

To address these concerns, we propose applying the BWA-AE to the resistive open defect identification problem, where we can consider the delays obtained by defect-free circuits only as the given known class (i.e. the normal class) and our approach is expected to discriminate between defects and non-defects during inference.

6.6.2 The Simulation Data

This case study used the data obtain by [91], which was a high-fidelity simulation dataset. Since the data generation is not the focus of this dissertation, we only briefly give readers an overview on it and encourage interested readers to refer to [91] for more details. In particular, an NAND cell was connected to a chain of λ inverters. The delay of each individual transistor was independent and followed a Gaussian distribution as $\tau \sim N(\mu_\tau, \sigma_\tau^2)$. To mimic the real-world manufacturing, two process variations, i.e. the channel length l and the gate width w were obtained separately by a Monte Carlo

simulation on SPICE [89]. The simulated data were defect-free and used for training. Moreover, in order to justify the detection performance, we also simulated defective data by inserting resistive opens of different sizes to the embedded NAND cells. It should be noted that the defective data were not used for training our BWA-AE and only used for evaluation. In total, 802 defect-free training samples were simulated and we augmented[2] the data to 4010. 5% of the augmented data were used for validation. The test set was composed of 193 defect-free samples and 205 defective samples. Similar to the previous experiments, AUC was the metric to measure the defect detection performance.

6.6.3 Comparison with Unsupervised Methods

We firstly compared the BWA-AE with three popular unsupervised anomaly detection algorithms:

- One-Class Support Vector Machine (OCSVM) [108]: We chose the optimal hyperparameters combinations γ and ν from $\{0.001, 0.01, 0.1, 0.5, 1, 10., 100, 500\}$ and $\{0.01, 0.1, 0.5, 0.75\}$, respectively;
- Isolation Forest [82]: The optimal number of estimators was determined from the set $\{5, 10, 15, 25, 50, 75, 100, 150, 200, 250, 500\}$;
- Local Outlier Factor (LOF) [17]: We searched different numbers of neighbors from $\{5, 10, 25, 50, 75, 100, 150, 200, 250, 500\}$.

Table 6.4 presents the unsupervised defect identification performance in terms of AUC for our method as well as the reference approaches. Without surprise, our BWA-AE outperformed the other three reference approaches with significant margin, namely with about $5.3\% \sim 7.0\%$ higher AUC, confirming the effectiveness of the BWA-AE in the domain of resistive open defect identification.

In order to give a visual understanding of the effectiveness of our approach, we compared the reconstructed delays for both defective and defect-free circuits. Figure 6.18

[2] We have proposed a novel method to augment delay data but it is beyond the scope of this chapter, so we recommend the interested readers to refer to our original paper [79] for more details.

Table 6.4: Comparison with unsupervised methods.

	OCSVM	Isolation Forest	LOF	Ours
AUC	0.895 (± 0.000)	0.880 (± 0.008)	0.889 (± 0.000)	**0.942** (± 0.005)

shows the results of non-defective circuits and we can see that the reconstructed delays (*orange solid line*) well matched the original input delays (*dashed blue line*), corresponding to small reconstruction errors for the known class. On the contrary, in Figure 6.19 shows the results of defective circuits, we can see an clear discrepancy between the original delays and their reconstructions, meaning notably larger reconstruction errors. It is interesting to notice that delays at larger voltages typically led to larger reconstruction errors. Based on this observation, it is expected to further enhance the overall detection performance by using a target mean squared error (tMSE) loss proposed in our previously published work [80] to BWA-AE. In particular, the tMSE loss can focus on the reconstructions of larger voltages during training and lead to better discrimination capability. We leave this as a future work.

6.6.4 Comparison with Supervised Methods

Although it is well known that unsupervised anomaly detection (i.e. an OCC problem) is typically more challenging than a supervised anomaly detection (i.e. a binary or even multi-class classification problem), we are still curious about whether our approach can already have comparable detection power to supervised classifiers. Thereby, we included three common and easy-to-use supervised classification algorithms:

- k-Nearest Neighbor (kNN) [39]: we used Euclidean distance as metric and selected the number of neighbors from $\{5, 10, 15, 20, 25, 50, 100\}$;
- Support Vector Machine (SVM) [24]: we used radius basis function as the kernel and selected C from $\{0.01, 0.1, 1, 10, 20, 50\}$;
- Random Forest (RF) [16]: we selected the number of estimators from $\{5, 10, 25, 50, 100\}$ and depth from $\{5, 10, 20, 50, 100\}$.

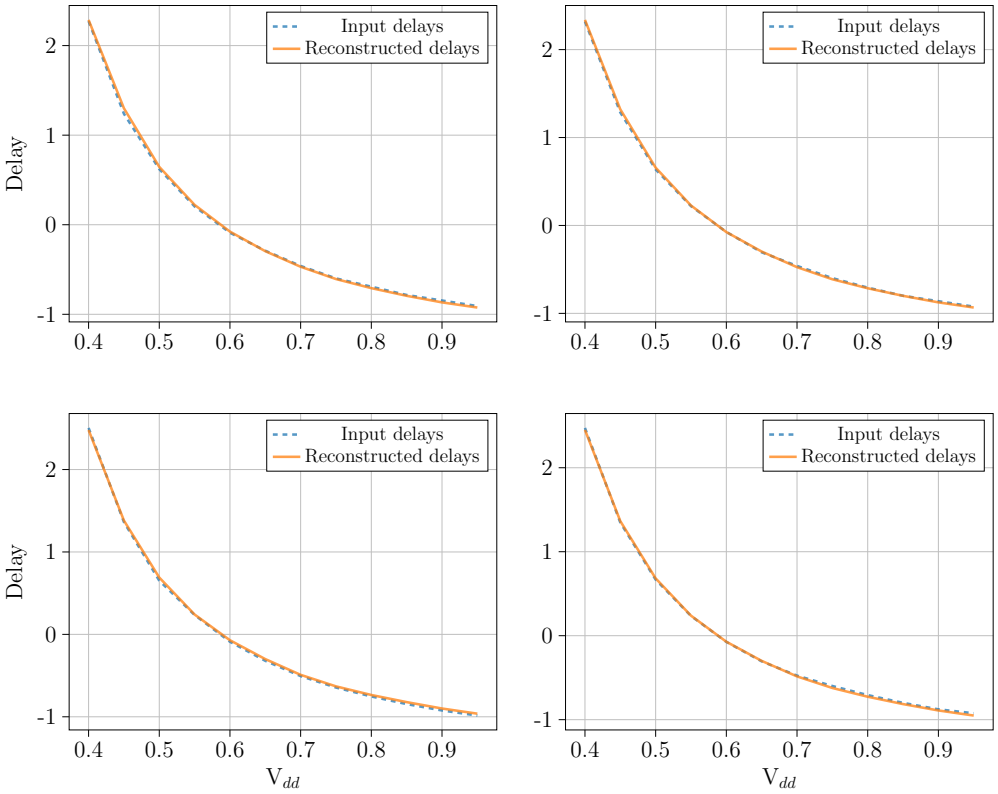


Figure 6.18: The delays and reconstructions for non-defective data. The delays were well reconstructed with invisible differences. Each subplot denotes an individual simulation.

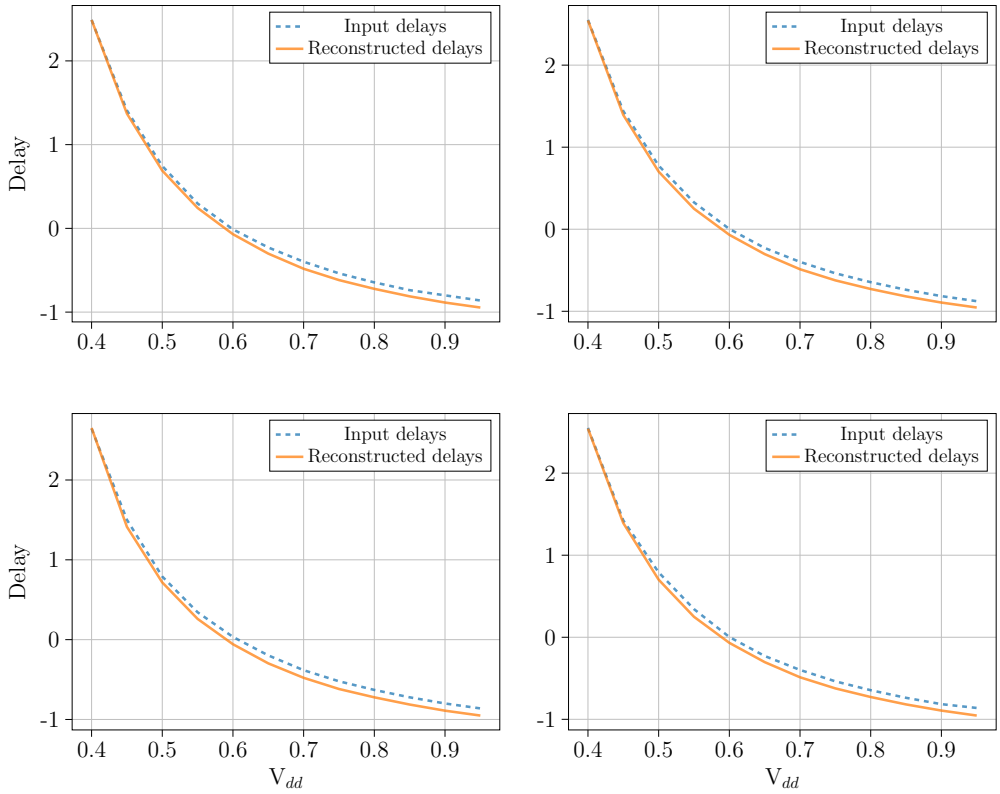


Figure 6.19: The delays and reconstructions for defective data. The delays were poorly reconstructed with notable differences. Each subplot denotes an individual simulation.

Table 6.5: Comparison with supervised methods.

	kNN	SVM	Random Forest	Ours
AUC	0.948 (± 0.000)	0.947 (± 0.000)	0.949 (± 0.002)	0.942 (± 0.005)

Table 6.5 summarizes the resulting AUCs on a separate test set for BWA-AE and the reference approaches. It should be noted that we fed a balanced dataset to the supervised approaches in our experiments so that they can be trained in an almost ideal environment. All three supervised classifiers outperformed our method without surprise. Nevertheless, the performance gap between our BWA-AE and the reference approaches is extremely minor. More precisely, our method has achieved more than 99.3% defect identification capability of the reference supervised methods in terms of AUC. This application further confirmed that our BWA-AE indeed captured the class-discriminative latent features for the non-defective data, although no label information was used for training. Our method is therefore valuable for practitioners because using our method only requires a few non-defective simulation samples but can already provide comparable defect identification performance as using both defective and non-defective simulation samples. This can save much money and time in the real world.

6.6.5 Further Investigations

The previous two sections have clearly shown the effectiveness of our novel BWA-AE for the use case of detecting resistive open defects. This section discusses two further advantages of using BWA-AE by presenting two experimental results.

Lack of Defect Simulations

This experiment compared BWA-AE to the aforementioned supervised classifiers in such a case that the reference methods were trained with limited defective data. In particular, we considered 11 cases: The ratio between defective and non-defective data was defined as ρ with $\rho \in \{1\%, 2\%, 5\%, 10\%, 15\%, 20\%, 25\%, 50\%, 60\%, 75\%, 100\%\}$. That is to say, for example, $\rho = 20\%$ indicates that the number of defective samples is

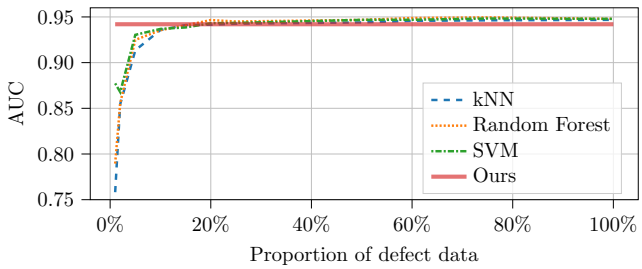


Figure 6.20: The comparison of the defect identification ability under different amounts of defect data during training. Our method (*red solid line*) was independent of defect data and had consistent AUC. On the contrary, supervised methods had poor performance when limited defect data were available.

only 20% of that for the non-defective data. Figure 6.20 illustrates the resulting AUCs with respect to different ρ. Our method is unsupervised and did not use any defective data so it had a consistent performance of 0.942 as before. However, interestingly, the identification performance of the supervised reference methods was significantly affected by the amounts of the available defective data. The experimental results show that the reference approaches required at least about 20% defective data in order to have a comparable identification capability to our method, indicating that using BWA-AE can save at least about 16.7% simulation time in practice. This further implies saving much more money in real-world manufacturing.

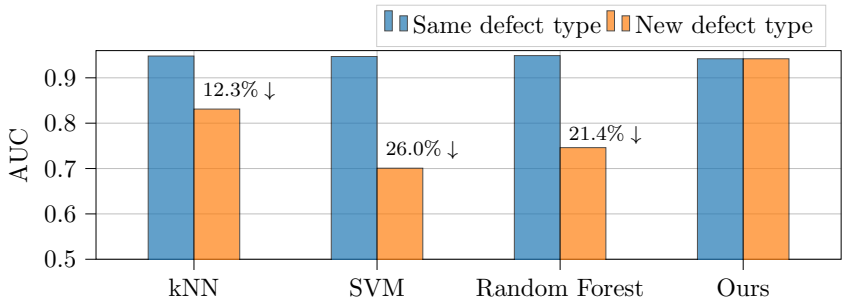


Figure 6.21: The performance comparison for detecting non-target defects. Blue bars demonstrate the case where same defect type was encountered during test, while orange bars shows the performance for non-target defects during test. Notable performance drops can be easily observed by the supervised methods, while our method had consistent AUCs.

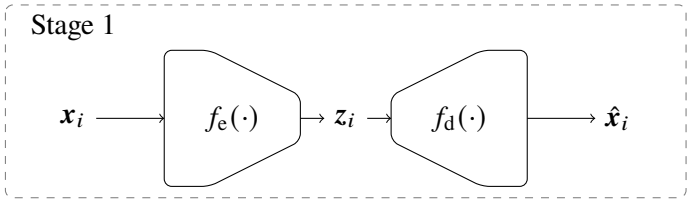


Figure 6.22: An arbitrary trained autoencoder dedicated to a given anomaly detection problem.

Non-Target Defects

It is well known that data-driven approaches can provide poor performance for the test data distribution shifts [98, 113]. This holds true for anomaly detection as well since we cannot cover all potential defect types during training in practice and non-target defects can occur in the real world. This section aims to show the consistent detection performance of our BWA-AE in comparison with the supervised reference approaches. As shown in Figure 6.21, the blue bars show the resulting AUCs if all defect types are known and fixed for the supervised methods. Clearly, in this case, the reference approaches may slightly outperform our approach. However, when we include non-target defect types during test, the detection performance of the supervised reference methods significantly dropped as shown in orange bars by up to 26%. In contrast, our method did not include any defective data during training and thus avoid potential bias of the possibly available defective data. As a result, BWA-AE showed robust performance towards novel defects because our approach as a one-class classifier can learn a closed decision boundary around the given non-defective data.

6.7 Batch-Wise Attention as a Plug-In

The previous sections have demonstrated an end-to-end solution for one-class classification based on our BWA-block. Nevertheless, we suggest that the BWA-block can be considered as an off-the-shelf plug-in component for existing autoencoder-based anomaly detectors.

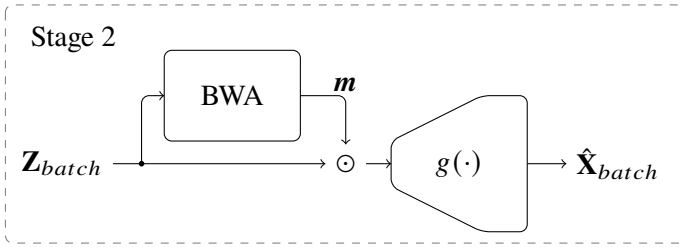


Figure 6.23: BWA as a plug-in to enhance the anomaly detection performance.

This section briefly demonstrates how to use the BWA-block to further enhance the anomaly detection performance of a *trained* autoencoder (Figure 6.22) in a semi-supervised setting. In particular, the latent representations calculated by the trained encoder are used as inputs to train a combination of the BWA-block and the decoder as shown in Figure 6.23. As a result, the learned feature mask $\boldsymbol{m}$ is able to tell which learned latent dimensions are more critical and which dimensions are meaningless or even misleading for the subsequent anomaly detection.

However, it might be difficult to detect anomalies if we simply retain some latent features and remove the rest features, if the latent features are obtained from a separately trained autoencoder. This is because the encoding procedure of the autoencoder takes into account the information of entire data, i.e. all original features. Therefore, even the non-selected features of the latent representations can still possess certain information about the normal class. This also suggests that trivial multiplying $\boldsymbol{m}$ to the learned latent representation $\boldsymbol{z}$ cannot yield reliable results in this case. Therefore, we define a weighting process as

$$\tilde{z}_{i,j} = \begin{cases} z_{i,j} & \text{, if } z_{i,j} \in \text{top-}k \text{ features} \\ \tau \cdot z_{i,j} & \text{, otherwise} \end{cases} \tag{6.10}$$

where $\tau \in (0, 1)$ is a weighting factor and $z_{i,j}$ is the j-th element of the learned latent representation $\boldsymbol{z}_i$ of the sample $\boldsymbol{x}_i$. In this way, the resulting weighted latent representation is denoted as $\tilde{\boldsymbol{z}}_i = [\tilde{z}_{i,1}, \tilde{z}_{i,2}, \ldots, \tilde{z}_{i,l}]^\top$. Consequently, for each test instance $\boldsymbol{x}_i$, the refined reconstruction error is calculated as:

$$\varepsilon_i = ||\boldsymbol{x}_i - f_{\mathrm{d}}(\tilde{\boldsymbol{z}}_i)||_2^2. \tag{6.11}$$

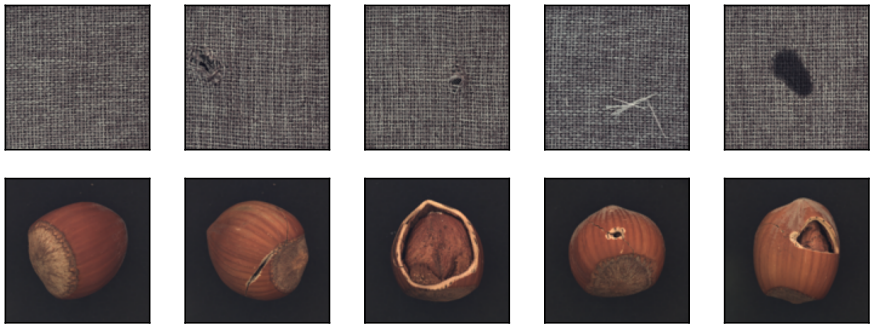


Figure 6.24: Exemplary MVTec images. The first column from the left shows the normal images, while the rest columns illustrate the abnormal samples. The first row shows some images of carpet and the second row shows some images of hazelnuts.

Finally, as introduced before, a test sample is rejected as anomaly if $\varepsilon_i > \varepsilon_0$. It should be additionally mentioned that k and τ are two additional hyperparameters which can be fine-tuned based on a few available anomalies in a semi-supervised setting.

We evaluated the proposed method on the popular MVTec-AD dataset [13], which is a real-world dataset created for anomaly detection. Some exemplary images are shown in Figure 6.24. The original MVTec-AD dataset contains high-resolution images (1024×1024 pixels) for 15 different categories. To save training time, all images were downscaled to 120×120 pixels. Moreover, as suggested in [13], we used data augmentation to increase the training data size to 4000 for each category by randomly flipping, rotating and mirroring images. Specifically, for each category, defect-free products were considered to be normal, while the products affiliated with the same category but with defects were denoted as abnormal samples.

Figure 6.25 shows the AUC for each individual category of the MVTec-AD dataset. Using the BWA as a plug-in can outperform a vanilla autoencoder with more than 27% improvement averaged over the 15 cases. In 10 out of 15 cases, the plug-in idea significantly enhanced the anomaly detection performance by more than 5%, where the largest improvement was about 152% for the category *carpet*. The overall significant improvement confirms the effectiveness of the BWA block as a plug-in for a trained

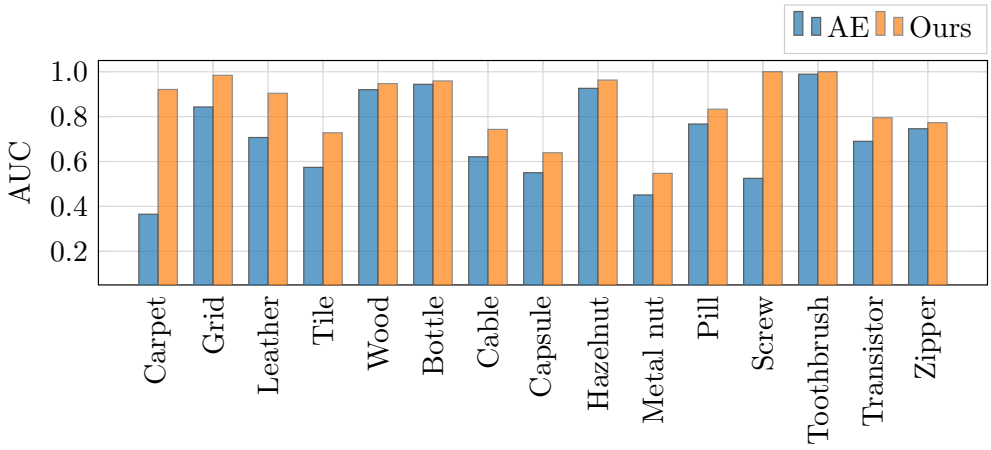


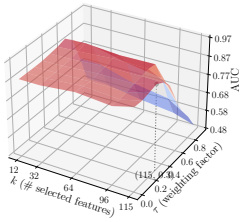
Figure 6.25: AUC for each individual category on the MVTec dataset.

autoencoder in the context of anomaly detection.

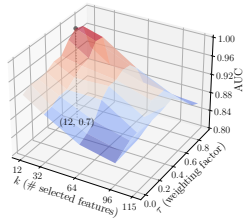
To further investigate the effectiveness of the proposed plug-in for the refinement, we plot the performance versus different k and τ for each individual category in Figure 6.26. As can be seen, the originally learned latent representation ($k = 128$ and $\tau = 1.0$) generally does not guarantee an optimal anomaly detection performance. This observation further confirms that a vanilla autoencoder does learn all basic building blocks instead of class-discriminative latent features. By introducing our BWA block as a plug-in, we can easily find out a better latent representation dedicated to anomaly detection.

6.8 Summary

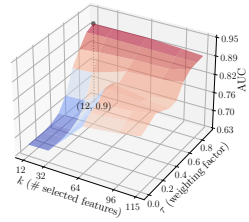
This chapter has demonstrated the novel Batch-Wise Attention mechanism to autoencoders in the context of one-class classification. Importantly, we have discovered some unexpected behavior of vanilla autoencoders for the OCC problems, namely unreasonable generalization ability to unknown classes as well as the OCC performance degradation during training. To solve these problems, we have proposed a novel variant of a vanilla autoencoder which is called BWA-AE, where the batch-wise attention



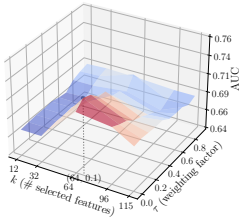
(a) Carpet



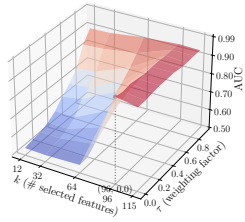
(b) Grid



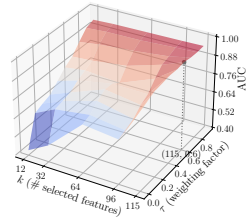
(c) Leather



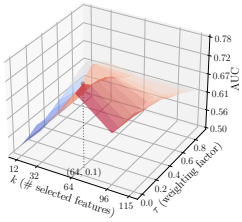
(d) Tile



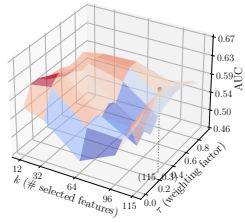
(e) Wood



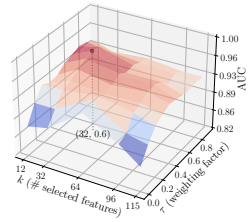
(f) Bottle



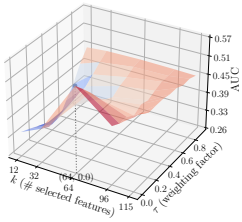
(g) Cable



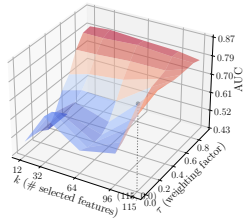
(h) Capsule



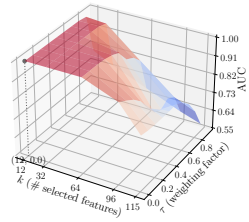
(i) Hazelnut



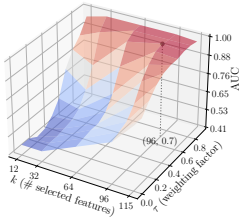
(j) Metal nut



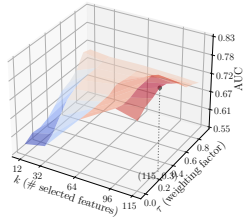
(k) Pill



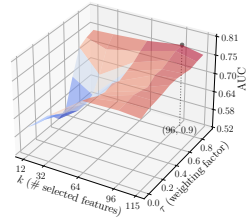
(l) Screw



(m) Toothbrush



(n) Transistor



(o) Zipper

Figure 6.26: AUC with respect to k and τ. The highest AUC is marked with a black point with dotted lines.

block was applied to the latent space. In this way, the BWA-block can intelligently regularize the training procedure, forcing the entire neural network to generalize to the known class only, while limiting such generalization ability to the unknown classes. As a result, the performance degradation has been alleviated and reduced. In addition, the novel BWA-AE has achieved better performance than the state of the art on the two benchmarking datasets. In-depth experiments have also demonstrated the difference between our approach and a vanilla AE with small latent dimensions. Last but not least, we have successfully applied the BWA-AE to a real-world application, i.e. unsupervised resistive open defect identification, showing the great potential of our approach in other research fields.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

This dissertation dealt with feature selection using deep learning concepts, which is of high relevance in both academic and industrial scenarios. Firstly, the nature of deep learning enables high scalability with respect to data volume and dimensionality for users. Secondly, feature selection itself is a critical tool to investigate big data for different real-world scenarios to improve efficiency and reliability.

One of the key contributions of the dissertation is the novel Feature Mask approach. It is based on neural networks and can easily scale to high-dimensional data of large volumes. One important superiority of the FM-method in comparison with the existing DL-based FS methods is that the FM-method does not introduce additional loss terms to the ordinary learning objective and is therefore easy to use. Intensive experiments were conducted on synthetic, benchmarking and real-world datasets to confirm the superior selection quality of our method. Generally speaking, it had notably better performance than the contemporary or state-of-the-art reference methods for most cases. In addition, comprehensive experiments towards the structure of the FM-module (i.e. depth of the non-linear transformation block, network initialization, batch sizes) show that our method is insensitive to those hyperparameters, which is valuable in practice. Although the FM-method was designed mainly in a supervised feature selection environment, we also extended it to unsupervised feature selection use cases and demonstrated its performance against the state-of-the-art approach. Last but not least, the FM-method

can automatically remove redundant candidate features, while all the reference methods failed in our experiments. This property allows a higher selection quality.

Motivated by the use case of post-silicon validation, the second novel contribution of this dissertation is the framework called Conditional Feature Selection. We have for the first time proposed a framework that can easily and efficiently integrate expert knowledge or prior information into the training procedure of a DL-based feature selection model (i.e. the FM-method in our case). Experiments on both synthetic and real-world data have shown that integrating expert knowledge as conditions can help to enhance the feature selection quality. Overall, this novel idea demonstrates a promising way of grounding artificial intelligence techniques in the areas where plenty of expert knowledge is available and must be considered.

The third novel idea of this dissertation is the Complementary Feature Mask, aiming to further enhance the feature selection performance of the original FM-method. One important contribution of CFM is that it can significantly enhance the feature selection quality in terms of small subset sizes, which has been observed as a major performance bottleneck for many existing DL-based FS methods. Experiments on the benchmarking datasets have shown that the novel design of CFM also led to extremely small deviations in terms of the accuracy in downstream classifiers, indicating the stability of the selection results. Although the design of an optimal complementary feature mask remains an open question, we provided an easy implementation with reliable performance, which reached state-of-the-art performance. Finally, the overall CFM framework is not restricted to the FM-method and we demonstrated applying CFM to other existing DL-based FS methods as well, showing its genericness .

Furthermore, we abstracted the FM-module to a Batch-Wise Attention mechanism and applied it to one-class classification and unsupervised anomaly detection. A novel contribution is that we discovered two unexpected behavior of autoencoders in the context of OCC, namely performance degradation during training and unreasonable generalization ability to unknown classes. BWA was applied to the latent space of autoencoders and solved the aforementioned undesirable behavior. An autoencoder with a BWA-module easily achieved state-of-the-art OCC performance on benchmarking datasets in comparison with the other reference methods. Comprehensive experiments

also show that the BWA-module successfully learned more meaningful latent features, which did not equal a vanilla autoencoder with small latent dimensions. Based on the success on the benchmarking datasets, this idea was evaluated on a use case of resistive open defects identification, where our method outperformed other popular unsupervised anomaly detection algorithms and even reached comparable performance to the considered supervised methods. More importantly, our method has shown its superiority in the situations, where extreme data imbalance or non-target defects occurred. Finally, the BWA-module was used as a plug-in to a trained autoencoder to improve the anomaly detection performance as a post-processing technique. Experiments on real-world industrial datasets confirmed its effectiveness.

7.2 Future Work

This dissertation also provides several promising future research directions based on the accomplished achievements of the proposed methodologies.

Uncertainty Measure for Feature Selection Most existing feature selection methods including the proposed methodologies in this dissertation are deterministic, meaning that they output fixed feature importance scores once they are trained. That is to say, we cannot know how certain the learned importance is. This can be risky in real-world applications because we typically want to make reliable decisions. To solve this problem, it is natural to introduce an uncertainty measure to our FM-method. There are several potential solutions. Firstly, we can consider ensemble models, meaning that we train our FM-method multiple times but with slightly different network architectures. After training, we can obtain the feature importance scores based on different models and calculate their average values and standard deviations to quantify the uncertainty. Another way is to leverage the idea of Bayesian Neural Network [59]. To enable this, we can replace the fully connected layers within the FM-module with Bayesian dense layers [116]. However, both research directions may face the problem of heavy computational consumption, which requires further study and investigation.

FM-Module for Neural Network Pruning Pruning a neural network [70] often means removing neurons of intermediate layers of the neural network, which can be considered as a feature selection problem for each intermediate layer. Therefore, it is interesting to see whether and how we can apply the FM-module to individual layers of an existing neural network to find out the most important neurons (features) and correspondingly prune the neural network. The challenge here is the trade-off between computational efficiency and numerical stability. On one hand, for each individual hidden layer, we can train a separate FM-module on it to prune it, but it is of high computational efficiency because we have to train the model n times for a network with n hidden layers. On the other hand, we can apply the FM-module on all hidden layers and train the model only once, but this may lead to numerical issues. This is because we introduce overly many *softmax* functions in a neural network and stacking of them can lead to severe gradient vanishing problems. Dealing with this trade-off or looking for an alternative requires much further research.

General Learning Objective for the FM-Method Currently, the learning objective for the FM-method is either the minimization of an MSE loss for regression tasks, or the minimization of a cross entropy loss for classification tasks. Despite the success in the conducted experiments, both learning objectives are not general enough. Therefore, one future research direction is to find out a more general learning objective for our FM-method to enhance the selection quality. One promising candidate is to maximize the mutual information between the candidate features and the targets, because relevant candidate features often have the largest mutual information with the given target variable by definition [42]. However, state-of-the-art approaches for estimating mutual information [60] are typically not differentiable and hardly applied to DL-based FS frameworks. Recently, a study [12] proposed to use neural networks to estimate mutual information and it is therefore straightforward to consider integrating it to our FM-method to provide a more general solution.

FM-Module for Different Data Forms Throughout this dissertation, we always assume that raw data are stored in a tabular form, where each row represents a sample

and each column denotes a feature. This assumption makes sense because most real-world data in the industry are exactly stored in this way. However, sometimes we may encounter other data types such as images or time series and want to perform feature selection on them. Naturally, we can vectorize images and time series into tabular forms and feed them to our FM-method. Nevertheless, it is inefficient due to the transformation process and may lose some information of hidden patterns during this transformation. In this case, conducting feature selection directly on the raw data form may be a more reasonable solution. Therefore, one other future work is to explore other forms for the FM-module, including replacing the fully connected layers with convolutional layers for dealing with images and with recurrent neural layers for dealing with time series. A more challenging research direction would be looking for a universal FM-module that can easily handle tabular data, image data and time series.

Bibliography

[1] Martín Abadi et al. “Tensorflow: A system for large-scale machine learning”. In: *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*. 2016, pp. 265–283.

[2] Davide Abati et al. “Latent space autoregression for novelty detection”. In: *The IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 481–490.

[3] Shigeo Abe. “Modified backward feature selection by cross validation.” In: *ESANN*. 2005, pp. 163–168.

[4] Abubakar Abid, Muhammed Fatih Balin, and James Zou. “Concrete Autoencoders for Differentiable Feature Selection and Reconstruction”. In: *Proceedings of the 36th International Conference on Machine Learning, PMLR*. 2019.

[5] Ehsan Aghaei and Gursel Serpen. “Host-based anomaly detection using Eigentraces feature extraction and one-class classification on system call trace data”. In: *arXiv preprint arXiv:1911.11284* (2019).

[6] Prachi Agrawal et al. “Metaheuristic algorithms on feature selection: A survey of one decade of research (2009-2019)”. In: *Ieee Access* 9 (2021), pp. 26766–26791.

[7] Hussein Almuallim and Thomas G Dietterich. “Learning boolean concepts in the presence of many irrelevant features”. In: *Artificial intelligence* 69.1-2 (1994), pp. 279–305.

[8] Jun Chin Ang et al. “Supervised, unsupervised, and semi-supervised feature selection: a review on gene selection”. In: *IEEE/ACM transactions on computational biology and bioinformatics* 13.5 (2015), pp. 971–989.

[9] Anton P Barten. “The coefficient of determination for regression without a constant term”. In: *The Practice of Econometrics: Studies on Demand, Forecasting, Money and Income* (1987), pp. 181–189.

[10] MohammadHossein Bateni et al. “Sequential Attention for Feature Selection”. In: *arXiv preprint arXiv:2209.14881* (2022).

[11] Laura Beggel, Michael Pfeiffer, and Bernd Bischl. “Robust Anomaly Detection in Images Using Adversarial Autoencoders”. In: *Machine Learning and Knowledge Discovery in Databases - European Conference, ECMLPKDD, 2019*. 2019, pp. 206–222.

[12] Mohamed Ishmael Belghazi et al. “Mutual information neural estimation”. In: *International conference on machine learning*. PMLR. 2018, pp. 531–540.

[13] Paul Bergmann et al. “MVTec AD–A comprehensive real-world dataset for unsupervised anomaly detection”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 9592–9600.

[14] Vadim Borisov, Johannes Haug, and Gjergji Kasneci. “CancelOut: A Layer for Feature Selection in Deep Neural Networks”. In: *International Conference on Artificial Neural Networks*. Springer. 2019, pp. 72–83.

[15] Andrew P Bradley. “The use of the area under the ROC curve in the evaluation of machine learning algorithms”. In: *Pattern recognition* 30.7 (1997), pp. 1145–1159.

[16] Leo Breiman. “Random forests”. In: *Machine learning* 45.1 (2001), pp. 5–32.

[17] Markus M Breunig et al. “LOF: identifying density-based local outliers”. In: *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*. 2000, pp. 93–104.

[18] Jie Cai et al. “Feature selection in machine learning: A new perspective”. In: *Neurocomputing* 300 (2018), pp. 70–79.

[19] Rich Caruana and Virginia R de Sa. “Benefitting from the variables that variable selection discards”. In: *Journal of machine learning research* 3.Mar (2003), pp. 1245–1264.

[20] Lester Lik Teck Chan et al. “Just-in-time modeling with variable shrinkage based on Gaussian processes for semiconductor manufacturing”. In: *IEEE Transactions on Semiconductor Manufacturing* 31.3 (2018), pp. 335–342.

[21] Girish Chandrashekar and Ferat Sahin. “A survey on feature selection methods”. In: *Computers & Electrical Engineering* 40.1 (2014), pp. 16–28.

[22] Argon Chen and Amos Hong. “Sample-efficient regression trees (SERT) for semiconductor yield loss analysis”. In: *IEEE transactions on semiconductor manufacturing* 23.3 (2010), pp. 358–369.

[23] Ting Chen et al. “A simple framework for contrastive learning of visual representations”. In: *International conference on machine learning*. PMLR. 2020, pp. 1597–1607.

[24] Corinna Cortes and Vladimir Vapnik. “Support-vector networks”. In: *Machine learning* 20.3 (1995), pp. 273–297.

[25] Xiaonan Cui et al. “Robust Randomized Autoencoder and Correntropy Criterion-Based One-Class Classification”. In: *IEEE Transactions on Circuits and Systems II: Express Briefs* 68.4 (2020), pp. 1517–1521.

[26] Lucas Deecke et al. “Image anomaly detection with generative adversarial networks”. In: *Joint european conference on machine learning and knowledge discovery in databases*. Springer. 2018, pp. 3–17.

[27] Jacob Devlin et al. “Bert: Pre-training of deep bidirectional transformers for language understanding”. In: *arXiv preprint arXiv:1810.04805* (2018).

[28] Chris Ding and Hanchuan Peng. “Minimum redundancy feature selection from microarray gene expression data”. In: *Journal of bioinformatics and computational biology* 3.02 (2005), pp. 185–205.

[29] Mark Fanty and Ronald Cole. “Spoken letter recognition”. In: *Advances in Neural Information Processing Systems*. 1991, pp. 220–226.

[30] Francesc J Ferri et al. “Comparative study of techniques for large-scale feature selection”. In: *Machine Intelligence and Pattern Recognition*. Vol. 16. Elsevier, 1994, pp. 403–413.

[31] Long Gao et al. “Handling imbalanced medical image data: A deep-learning-based one-class classification approach”. In: *Artificial intelligence in medicine* 108 (2020), p. 101935.

[32] Pierre Geurts, Damien Ernst, and Louis Wehenkel. “Extremely randomized trees”. In: *Machine learning* 63.1 (2006), pp. 3–42.

[33] Xavier Glorot and Yoshua Bengio. “Understanding the difficulty of training deep feedforward neural networks”. In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. 2010, pp. 249–256.

[34] José L Gómez-Sirvent et al. “Optimal feature selection for defect classification in semiconductor wafers”. In: *IEEE Transactions on Semiconductor Manufacturing* 35.2 (2022), pp. 324–331.

[35] Dong Gong et al. “Memorizing normality to detect anomaly: Memory-augmented deep autoencoder for unsupervised anomaly detection”. In: *The IEEE/CVF International Conference on Computer Vision*. 2019, pp. 1705–1714.

[36] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[37] Ning Gui, Danni Ge, and Ziyin Hu. “Afs: An attention-based mechanism for supervised feature selection”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 2019, pp. 3705–3713.

[38] Cheng Guo and Felix Berkhahn. “Entity embeddings of categorical variables”. In: *arXiv preprint arXiv:1604.06737* (2016).

[39] Gongde Guo et al. “KNN model-based approach in classification”. In: *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*. Springer. 2003, pp. 986–996.

[40] Isabelle Guyon and André Elisseeff. “An introduction to variable and feature selection”. In: *Journal of machine learning research* 3.Mar (2003), pp. 1157–1182.

[41] Isabelle Guyon et al. “Competitive baseline methods set new standards for the NIPS 2003 feature selection benchmark”. In: *Pattern recognition letters* 28.12 (2007), pp. 1438–1444.

[42] Isabelle Guyon et al. *Feature extraction: foundations and applications*. Vol. 207. Springer, 2008.

[43] Friedrich Hapke et al. “Defect-oriented test: Effectiveness in high volume manufacturing”. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 40.3 (2020), pp. 584–597.

[44] Xiangteng He, Yuxin Peng, and Junjie Zhao. “Fine-grained discriminative localization via saliency-guided faster R-CNN”. In: *Proceedings of the 25th ACM international conference on Multimedia*. 2017, pp. 627–635.

[45] Taewon Heo, Youngju Kim, and Chang Ouk Kim. “A Modified Lasso Model for Yield Analysis Considering the Interaction Effect in a Multistage Manufacturing Line”. In: *IEEE Transactions on Semiconductor Manufacturing* 35.1 (2021), pp. 32–39.

[46] Wei Bin How et al. “Significance of the chemical environment of an element in nonadiabatic molecular dynamics: Feature selection and dimensionality reduction with machine learning”. In: *The Journal of Physical Chemistry Letters* 12.50 (2021), pp. 12026–12032.

[47] Jie Hu, Li Shen, and Gang Sun. “Squeeze-and-excitation networks”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 7132–7141.

[48] Wei Hu et al. “An overview of hardware security and trust: Threats, countermeasures, and design tools”. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 40.6 (2020), pp. 1010–1038.

[49] Phillip Isola et al. “Image-to-image translation with conditional adversarial networks”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1125–1134.

[50] Jon Paul Janet and Heather J Kulik. “Resolving transition metal chemical space: Feature selection for machine learning and structure–property relationships”. In: *The Journal of Physical Chemistry A* 121.46 (2017), pp. 8939–8954.

[51] SK Kamaruddin and Vadlamani Ravi. “Credit card fraud detection using big data analytics: use of PSOAANN based one-class classification”. In: *Proceedings of the international conference on informatics and analytics*. 2016, pp. 1–8.

[52] Seokho Kang, Dongil Kim, and Sungzoon Cho. “Efficient feature selection-based on random forward search for virtual metrology modeling”. In: *IEEE Transactions on Semiconductor Manufacturing* 29.4 (2016), pp. 391–398.

[53] Kittisak Kerdprasop and Nittaya Kerdprasop. “Feature selection and boosting techniques to improve fault detection accuracy in the semiconductor manufacturing process”. In: *World Congress on Engineering 2012. July 4-6, 2012. London, UK.* Vol. 2188. International Association of Engineers. 2010, pp. 398–403.

[54] Shehroz S Khan and Michael G Madden. “A survey of recent trends in one class classification”. In: *Irish conference on artificial intelligence and cognitive science*. Springer. 2009, pp. 188–197.

[55] Kyung-Jun Kim et al. “Variable selection under missing values and unlabeled data in semiconductor processes”. In: *IEEE Transactions on Semiconductor Manufacturing* 32.1 (2018), pp. 121–128.

[56] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[57] Kenji Kira and Larry A. Rendell. “The Feature Selection Problem: Traditional Methods and a New Algorithm”. In: *AAAI Conference on Artificial Intelligence*. 1992.

[58] Ron Kohavi and George H John. “Wrappers for feature subset selection”. In: *Artificial intelligence* 97.1-2 (1997), pp. 273–324.

[59] Igor Kononenko. “Bayesian neural networks”. In: *Biological Cybernetics* 61.5 (1989), pp. 361–370.

[60] Alexander Kraskov, Harald Stögbauer, and Peter Grassberger. “Estimating mutual information”. In: *Physical review E* 69.6 (2004), p. 066138.

[61] Bartosz Krawczyk and Michał Woźniak. “One-class classifiers with incremental learning and forgetting for data streams with concept drift”. In: *Soft Computing* 19.12 (2015), pp. 3387–3400.

[62] Alex Krizhevsky. *Learning multiple layers of features from tiny images*. Tech. rep. Citeseer, 2009.

[63] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Communications of the ACM* 60.6 (2017), pp. 84–90.

[64] Thomas Navin Lal et al. “Embedded methods”. In: *Feature Extraction: Foundations and Applications* (2006), pp. 137–165.

[65] Ken Lang. “Newsweeder: Learning to filter netnews”. In: *Proceedings of the Twelfth International Conference on Machine Learning*. 1995, pp. 331–339.

[66] Yann LeCun et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[67] Jundong Li et al. “Feature selection: A data perspective”. In: *ACM Computing Surveys (CSUR)* 50.6 (2017), pp. 1–45.

[68] Xin Li. “Post-silicon performance modeling and tuning of analog/mixed-signal circuits via Bayesian model fusion”. In: *2012 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE. 2012, pp. 551–552.

[69] Yifeng Li, Chih-Yu Chen, and Wyeth W Wasserman. “Deep feature selection: theory and application to identify enhancers and promoters”. In: *Journal of Computational Biology* 23.5 (2016), pp. 322–336.

[70] Tailin Liang et al. “Pruning and quantization for deep neural network acceleration: A survey”. In: *Neurocomputing* 461 (2021), pp. 370–403.

[71] Yiwen Liao, Alexander Bartler, and Bin Yang. “Anomaly detection based on selection and weighting in latent space”. In: *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*. IEEE. 2021, pp. 409–415.

[72] Yiwen Liao, Raphaël Latty, and Bin Yang. “Feature selection using batch-wise attenuation and feature mask normalization”. In: *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2021, pp. 1–9. DOI: 10.1109/IJCNN52387.2021.9533531.

[73] Yiwen Liao, Raphaël Latty, and Bin Yang. *Experts in the Loop: Conditional Variable Selection for Accelerating Post-Silicon Analysis Based on Deep Learning*. 2022. DOI: 10.48550/ARXIV.2209.15249. URL: https://arxiv.org/abs/2209.15249.

[74] Yiwen Liao and Bin Yang. “To generalize or not to generalize: Towards autoencoders in one-class classification”. In: *2022 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2022, pp. 1–8.

[75] Yiwen Liao et al. “A Deep-Learning-Aided Pipeline for Efficient Post-Silicon Tuning”. In: arXiv, 2022. DOI: 10.48550/ARXIV.2207.00336. URL: https://arxiv.org/abs/2207.00336.

[76] Yiwen Liao et al. “A Learning-Aided Generic Framework for Fault Detection and Recovery of Inertial Sensors in Automated Driving Systems”. In: *IEEE Systems Journal* 15.2 (2021), pp. 3001–3011. DOI: 10.1109/JSYST.2020.3004805.

[77] Yiwen Liao et al. *Conditional Variable Selection for Intelligent Test*. 2022. DOI: 10.48550/ARXIV.2207.00335. URL: https://arxiv.org/abs/2207.00335.

[78] Yiwen Liao et al. *Deep Feature Selection Using a Novel Complementary Feature Mask*. 2022. DOI: 10.48550/ARXIV.2209.12282. URL: https://arxiv.org/abs/2209.12282.

[79] Yiwen Liao et al. “Efficient and Robust Resistive Open Defect Detection Based on Unsupervised Deep Learning”. In: *2022 IEEE International Test Conference (ITC)*. IEEE. 2022, pp. 185–193.

[80] Yiwen Liao et al. “Unsupervised fault detection and recovery for intelligent robotic rollators”. In: *Robotics and Autonomous Systems* (2021), p. 103876.

[81] Yiwen Liao et al. “Wafer Map Defect Classification Based on the Fusion of Pattern and Pixel Information”. In: *2022 IEEE International Test Conference (ITC)*. IEEE. 2022, pp. 1–9.

[82] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. “Isolation Forest”. In: *2008 Eighth IEEE International Conference on Data Mining*. 2008, pp. 413–422. DOI: 10.1109/ICDM.2008.17.

[83] Huijuan Lu et al. “A hybrid feature selection algorithm for gene expression data classification”. In: *Neurocomputing* 256 (2017), pp. 56–62.

[84] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. “Rectifier nonlinearities improve neural network acoustic models”. In: *Proceedings of the 30th International Conference on Machine Learning (ICML)*. 2010.

[85] Leland McInnes, John Healy, and James Melville. “Umap: Uniform manifold approximation and projection for dimension reduction”. In: *arXiv preprint arXiv:1802.03426* (2018).

[86] Prabhat Mishra and Farimah Farahmandi. *Post-Silicon Validation and Debug*. Springer.

[87] Prabhat Mishra et al. "Post-Silicon Validation in the SoC Era: A Tutorial Introduction". In: *IEEE Design Test* 34.3 (2017), pp. 68–92. DOI: 10.1109/MDAT.2017.2691348.

[88] Hassan Ghasemzadeh Mohammadi, Pierre-Emmanuel Gaillardon, and Giovanni De Micheli. "Efficient statistical parameter selection for nonlinear modeling of process/performance variation". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 35.12 (2016), pp. 1995–2007.

[89] Laurence W. Nagel and D.O. Pederson. *SPICE (Simulation Program with Integrated Circuit Emphasis)*. Tech. rep. UCB/ERL M382. EECS Department, University of California, Berkeley, 1973. URL: http://www2.eecs.berkeley.edu/Pubs/TechRpts/1973/22871.html.

[90] Zahra Paria Najafi-Haghi, Marzieh Hashemipour-Nazari, and Hans-Joachim Wunderlich. "Variation-aware defect characterization at cell level". In: *2020 IEEE European Test Symposium (ETS)*. IEEE. 2020, pp. 1–6.

[91] Zahra Paria Najafi-Haghi and Hans-Joachim Wunderlich. "Resistive open defect classification of embedded cells under variations". In: *2021 IEEE 22nd Latin American Test Symposium (LATS)*. IEEE. 2021, pp. 1–6.

[92] Adam Paszke et al. "Pytorch: An imperative style, high-performance deep learning library". In: *Advances in neural information processing systems* 32 (2019).

[93] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[94] Pramuditha Perera, Ramesh Nallapati, and Bing Xiang. "Ocgan: One-class novelty detection using gans with constrained latent representations". In: *The IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 2898–2906.

[95] Pramuditha Perera, Poojan Oza, and Vishal M Patel. "One-class classification: A survey". In: *preprint arXiv:2101.03064* (2021).

[96] Stanislav Pidhorskyi et al. "Generative probabilistic novelty detection with adversarial autoencoders". In: *The 32nd International Conference on Neural Information Processing Systems*. 2018, pp. 6823–6834.

[97] Kemal Polat and Salih Güneş. "A new feature selection method on classification of medical datasets: Kernel F-score feature selection". In: *Expert Systems with Applications* 36.7 (2009), pp. 10367–10373.

[98] Joaquin Quiñonero-Candela et al. *Dataset shift in machine learning*. Mit Press, 2008.

[99] Alec Radford et al. "Improving language understanding by generative pre-training". In: (2018).

[100] Beatriz Remeseiro and Veronica Bolon-Canedo. "A review of feature selection methods in medical applications". In: *Computers in biology and medicine* 112 (2019), p. 103375.

[101] Jiangtao Ren et al. "Forward semi-supervised feature selection". In: *Advances in Knowledge Discovery and Data Mining: 12th Pacific-Asia Conference, PAKDD 2008 Osaka, Japan, May 20-23, 2008 Proceedings 12*. Springer. 2008, pp. 970–976.

[102] Robin Rombach et al. "High-resolution image synthesis with latent diffusion models". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 10684–10695.

[103] Lukas Ruff et al. "Deep one-class classification". In: *International conference on machine learning*. 2018, pp. 4393–4402.

[104] Mohammad Sabokrou et al. "Adversarially learned one-class classifier for novelty detection". In: *The IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 3379–3388.

[105] Mohammad Sabokrou et al. "Deep end-to-end one-class classifier". In: *IEEE transactions on neural networks and learning systems* 32.2 (2020), pp. 675–684.

[106] Patrick Schlachter, Yiwen Liao, and Bin Yang. "Deep one-class classification using intra-class splitting". In: *2019 IEEE Data Science Workshop (DSW)*. IEEE. 2019, pp. 100–104.

[107] Patrick Schlachter, Yiwen Liao, and Bin Yang. "One-class feature learning using intra-class splitting". In: *2019 27th European Signal Processing Conference (EUSIPCO)*. IEEE. 2019, pp. 1–5.

[108] Bernhard Schölkopf et al. "Estimating the support of a high-dimensional distribution". In: *Neural computation* 13.7 (2001), pp. 1443–1471.

[109] Naeem Seliya, Azadeh Abdollah Zadeh, and Taghi M Khoshgoftaar. "A literature review on one-class classification and its potential applications in big data". In: *Journal of Big Data* 8.1 (2021), pp. 1–31.

[110] Swati Shilaskar and Ashok Ghatol. "Feature selection for medical diagnosis: Evaluation for cardiovascular diseases". In: *Expert Systems with Applications* 40.10 (2013), pp. 4146–4153.

[111] Saúl Solorio-Fernández, J Ariel Carrasco-Ochoa, and José Fco Martínez-Trinidad. “A review of unsupervised feature selection methods”. In: *Artificial Intelligence Review* 53.2 (2020), pp. 907–948.

[112] Nitish Srivastava et al. “Dropout: a simple way to prevent neural networks from overfitting”. In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.

[113] Masashi Sugiyama and Motoaki Kawanabe. *Machine learning in non-stationary environments: Introduction to covariate shift adaptation*. MIT press, 2012.

[114] Joseph G Taylor et al. “Learning using unselected features (LUFe)”. In: *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*. 2016, pp. 2060–2066.

[115] Robert Tibshirani. “Regression shrinkage and selection via the lasso”. In: *Journal of the Royal Statistical Society: Series B (Methodological)* 58.1 (1996), pp. 267–288.

[116] Dustin Tran et al. “Bayesian layers: A module for neural network uncertainty”. In: *Advances in neural information processing systems* 32 (2019).

[117] Andrii Trelin and Aleš Procházka. *Binary Stochastic Filtering: feature selection and beyond*. 2020. arXiv: 2007.03920 [cs.LG].

[118] Ashish Vaswani et al. “Attention is all you need”. In: *Advances in neural information processing systems* 30 (2017).

[119] Jorge R Vergara and Pablo A Estévez. “A review of feature selection methods based on mutual information”. In: *Neural computing and applications* 24 (2014), pp. 175–186.

[120] Dimitrios Ververidis and Constantine Kotropoulos. “Sequential forward feature selection with low computational cost”. In: *2005 13th European Signal Processing Conference*. IEEE. 2005, pp. 1–4.

[121] Junliang Wang, Jie Zhang, and Xiaoxi Wang. “A data driven cycle time prediction with feature selection in a semiconductor wafer fabrication system”. In: *IEEE Transactions on Semiconductor Manufacturing* 31.1 (2018), pp. 173–182.

[122] Qian Wang et al. “Attentional neural network: Feature selection using cognitive feedback”. In: *Advances in neural information processing systems*. 2014, pp. 2033–2041.

[123] Zhou Wang et al. “Image quality assessment: from error visibility to structural similarity”. In: *IEEE transactions on image processing* 13.4 (2004), pp. 600–612.

[124] Qi Wei et al. “Anomaly detection for medical images based on a one-class classification”. In: *Medical Imaging 2018: Computer-Aided Diagnosis*. Vol. 10575. SPIE. 2018, pp. 375–380.

[125] Sanghyun Woo et al. “Cbam: Convolutional block attention module”. In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 3–19.

[126] Han Xiao, Kashif Rasul, and Roland Vollgraf. “Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms”. In: *arXiv preprint arXiv:1708.07747* (2017).

[127] Ying Xue et al. “Effect of molecular descriptor feature selection in support vector machine classification of pharmacokinetic and toxicological properties of chemical agents”. In: *Journal of chemical information and computer sciences* 44.5 (2004), pp. 1630–1638.

[128] Yiming Yang and Jan O Pedersen. “A comparative study on feature selection in text categorization”. In: *Icml*. Vol. 97. 412-420. Citeseer. 1997, p. 35.

[129] Fangming Ye et al. “Information-theoretic syndrome evaluation, statistical root-cause analysis, and correlation-based feature selection for guiding board-level fault diagnosis”. In: *IEEE transactions on computer-aided design of integrated circuits and systems* 34.6 (2015), pp. 1014–1026.

[130] Yang Zhang, Xue Li, and Maria Orlowska. “One-class classification of text streams with concept drift”. In: *2008 IEEE International Conference on Data Mining Workshops*. IEEE. 2008, pp. 116–125.

[131] Wei Zheng et al. “Feature Selection Boosted by Unselected Features”. In: *IEEE Transactions on Neural Networks and Learning Systems* (2021).

[132] Cheng Zhuo, Bei Yu, and Di Gao. “Accelerating chip design with machine learning: From pre-silicon to post-silicon”. In: *2017 30th IEEE International System-on-Chip Conference (SOCC)*. 2017, pp. 227–232. DOI: 10.1109/SOCC.2017.8226046.

[133] Hui Zou and Trevor Hastie. “Regularization and variable selection via the elastic net”. In: *Journal of the royal statistical society: series B (statistical methodology)* 67.2 (2005), pp. 301–320.