2022

Improving Feature Learning Capability and Interpretability of Unsupervised Neural Networks

Chathurika S. Wickramasinghe Brahmana
Virginia Commonwealth University

IMPROVING FEATURE LEARNING CAPABILITY AND INTERPRETABILITY OF UNSUPERVISED NEURAL NETWORKS

This dissertation is submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at Virginia Commonwealth University.

by

CHATHURIKA S. WICKRAMASINGHE BRAHMANA

Bachelor of Science, University of Peradeniya, Sri Lanka, 2016

Director: Milos Manic,

Professor, Department of Computer Science

Virginia Commonwealth University

Richmond, Virginia

May, 2022

Acknowledgements

First, I would like to thank my advisor and mentor, Prof. Milos Manic, for his help, support, and guidance. Second, I would like to thank Dr. Craig Rieger, Dr. Ronald Boring, Dr. Eyuphan Bulut, and Dr. David C. Shepherd for serving in my dissertation committee and for their valuable feedback. Further, I would like to acknowledge the continuous support given to me by the Idaho National Laboratory (INL), especially for the data provided for the experimentation and testing of methodologies presented in this dissertation. I also want to acknowledge the support provided by the Commonwealth Cyber Initiative (CCI), an investment in the advancement of cyber R&D, innovation and workforce development.

I extend my gratitude to all the great colleagues of the Modern Heuristics Research Group, Daniel, Kasun, Javi, Sandun, Victor, and Morgan for their support and contributions throughout my Ph.D. program. Last but not least, I wish to thank my family and friends, my brother Dhanushka, Sister-in-law Malika, my dearest friends, Buddhini, Pradeepa, Dinendra, Chanaka, Sharon, Sahan, Dumidu, Sam, Kalani, Akila, Shama, Hiran, Dileindree, Jayani, Paolo, for their immense support during my academic journey.

I dedicate my doctoral dissertation to my mother, Anula Jayawardhana, whose guidance, support, and values are the grounds that supported me through this journey.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Abstract

IMPROVING FEATURE LEARNING CAPABILITY AND INTERPRETABILITY OF UNSUPERVISED NEURAL NETWORKS

By Chathurika S. Wickramasinghe Brahmana

A submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at Virginia Commonwealth University.

Virginia Commonwealth University, 2022.

Director: Milos Manic,

Professor, Department of Computer Science

The motivation for this dissertation is two-prong. Firstly, the current state of Machine Learning (ML) imposes the need for unsupervised machine learning. Secondly, once such models are developed, a deeper understanding of ML models is necessary for humans to adapt and use such models.

Why is unsupervised ML needed? Real-world systems generate massive amounts of unlabelled data at rapid speed, limiting the usability of state-of-the-art supervised machine learning approaches. Further, the manual labeling process is expensive and time-consuming as it requires domain experts manually annotate the data. Therefore, the existing supervised learning algorithms are unable to take advantage of the abundance of real-world unlabelled data. Thus, relying on supervised learning alone is not sufficient in many real-world settings. Therefore, improving on existing and developing novel unsupervised machine learning algorithms is necessary.

Once the unsupervised ML models have been developed, these models need to be understood by humans in order to adapt them efficiently. Even with the tremendous

success of ML in many domains, humans are still hesitant to develop, deploy, and use machine learning methods because humans cannot understand the internal decision-making process of machine learning methods (black-box nature). Therefore, it is essential to develop machine learning algorithms that are either explainable or to develop techniques that can apply to ML models to explain their decision-making process. This process is typically referred to as explainable or interpretable machine learning. Therefore, developing novel methodologies for interpreting unsupervised machine learning methods is necessary.

The objectives of this dissertation are: 1) improving the feature learning capability of unsupervised neural networks and 2) interpreting the decision-making process of unsupervised neural networks. We address the objectives using two widely used unsupervised neural networks: Self-organizing Maps (SOM) and Autoencoders. For each of these unsupervised neural networks, we present architectural changes that improve feature learning capabilities. Further, we present novel interpretation methods for SOM based clustering and Autoencoder based anomaly detection. Thus, the contributions of this dissertation are summarized as follows: 1) Improving and Interpreting Self Organizing Neural Network; 2) Improving and Interpreting Autoencoder Neural network.

CHAPTER 1

INTRODUCTION

Why ML: Machine Learning (ML) is a branch of computer science and artificial intelligence. It consists of algorithms that are capable of learning from data to imitate the way that human learns and thinks [1]. The main goal of machine learning is making machines that can learn automatically without human intervention. Machine learning is undoubtedly an essential component in the field of data science. It consists of many components including logistic regression, linear regression, decision trees, K-means, PCA, and Neural Networks. Out of that, neural network-based algorithms have shown remarkable performances during the past couple of decades.

Why NNs: Artificial Neural Networks (ANNs) consist of a set of stacked layers that learn a series of hidden representations hierarchically [2]. Usually, they consist of an input layer, one or more hidden layers, and an output layer. Due to this hierarchical architecture, the higher-level representations contain amplified aspects of input samples that are suitable for discrimination and suppression of irrelevant features. ANNs have improved the state-of-the-art performance in many tasks, including speech recognition, object detection, natural language processing, and pattern recognition. Many neural network architectures belong to two categories of machine learning, namely supervised machine learning and unsupervised machine learning algorithms.

Why UnML: Unsupervised Machine Learning (UnML) has gained significant attention during the last decade. The main reason for this is the availability of a large amount of unlabelled data. Real-world settings bring the challenge of dealing with high volumes of unlabeled data. The manual labeling process is time-consuming,

expensive, and requires the expertise of the data [3]. Further, supervised feature learning is not only unable to take advantage of unlabelled data, but it also can result in biases by relying on labeled data. Due to these limitations, relying on supervised learning alone is not sufficient for data-driven decision making. Therefore, improving on existing and developing novel unsupervised machine learning algorithms is necessary.

Why Explainable AI (XAI): Even with the tremendous success of ML in many domains, humans are still hesitant to develop, deploy, and use machine learning methods because humans cannot understand the internal decision-making process of machine learning methods (black-box nature) [4]. Especially for human-in-the-loop systems, humans need to understand these algorithms such that they can trust these models. By addressing this question, the Explainable Machine Learning (XAI/ Interpretable AI) research area has been introduced and received lot of attention within many domains [5]. The goal of explainable ML is to provide reasoning for ML model outputs, allowing humans to understand and trust ML models' decision-making process. However, in the current literature, very little work has been performed to develop interpretable methods for unsupervised ML algorithms. Real-world systems impose the need for unsupervised Machine Learning. Therefore, developing novel techniques for interpreting unsupervised machine learning methods is necessary. (In this dissertation, transparency, interpretability and explainability are used interchangeably)

1.1 Motivation

1.1.1 Motivation for theoretical domain

As described above, the motivation for this dissertation is two-prong.

1. Real-world systems generate massive amounts of unlabelled data at rapid speed,

limiting the usability of state-of-the-art supervised machine learning approaches. Further, the manual labeling process is expensive, time-consuming, and requires the expertise of the data. Therefore, the existing supervised learning algorithms are unable to take advantage of the abundance of real-world unlabelled data. Thus, relying on supervised learning alone is not sufficient in many real-world settings. Therefore, improving on existing and developing novel unsupervised machine learning algorithms is necessary.

2. Once the unsupervised ML models are developed, these models need to be understood by humans in order to adapt them efficiently. Unfortunately, even with the tremendous success of ML in many domains, humans are still hesitant to develop, deploy, and use machine learning methods because humans cannot understand the internal decision-making process of machine learning methods (black-box nature). Hence, it is essential to develop machine learning algorithms that are either explainable or develop approaches to the decision-making process of existing methods. This process is typically referred to as explainable or interpretable machine learning. Therefore, developing novel techniques for interpreting unsupervised machine learning methods is necessary.

1.1.2 Motivation for application domain

This dissertation focuses on *Cyber-Physical Systems (CPSs)* as the application domain. Modern infrastructure and systems in many domains have become heavily reliant on CPSs, and they can be found in areas ranging from sensor networks [6], intelligent transportation systems, and smart grids to space exploration systems [7, 8, 9, 10, 11]. They typically consist of interconnected computing and physical resources, which enable interactive processing among systems [7]. Such systems integrate computations, communication, control, and physical processes to achieve a specific task

[8]. Due to widespread usage and economic benefits, ensuring the secure, reliable, resilient, and consistent performance of CPSs is crucial. Many independent agencies and national institutes such as the National Science Foundation (NSF), U.S. Department of Homeland Security (DHS), U.S. Department of Transportation (DOT), National Cancer Institute (NCI), and European Commission (E.C.) have recently put their attention towards the advancements of CPS. Their interests include the Internet of Things, Industrial Internet, Smart Cities, Smart Grids, and "smart" anything (Manufacturing, Cars, Buildings) [12, 13, 8, 9, 10].

CPSs produce massive amounts of data, creating opportunities to use predictive Machine Learning (ML) models to improve their operation reliability, improve their performance (in terms of production capacity and cost), performance optimization, preventive maintenance, and threat detection [13, 14, 12]. It has to be noticed that these CPSs produce massive amounts of unlabeled data rapidly. Consequently, relying on supervised learning alone is not sufficient for data-driven decision making in CPSs. If we are to maximize the use of ML in CPSs, it is necessary to have explainable unsupervised ML models. Therefore, in this work, we explore how unsupervised explainable ML could be used within the CPS domain for different applications such as *clustering, unsupervised feature learning, classification, and anomaly detection*. We experimented with these ML tasks on different data types such as image data, sensor readings, financial data, and network communication data.

1.2 Objectives

The objective of this dissertation is improving and interpreting unsupervised neural networks which is divided into two sub-objectives.

1. Improving the feature learning capability of unsupervised neural networks

2. Interpreting the decision making process of unsupervised neural networks

In this dissertation, *improving unsupervised neural networks* refers to improving the feature learning capability of these algorithms. Real-world systems generate a massive amount of unlabelled data. These data are coming through various sources resulting in high-dimensional feature spaces with a lot of data inconsistencies. Therefore, it is essential to extract/learn relevant features to improve the reliability and performance of downstream machine learning tasks such as clustering. This can be achieved by improving the feature learning capability of unsupervised neural networks.

In this dissertation, *interpreting unsupervised neural networks* refers to developing techniques to explain the underline decision-making process of these algorithms on down stream tasks such as clustering. To use unsupervised algorithms efficiently, users need to understand the rationale behind these algorithms. Further, it allows domain experts to understand, trust, debug, diagnose, and adapt them to different applications efficiently.

1.3 Contributions

This dissertation exemplifies the objectives using two widely used unsupervised neural networks: Self Organizing Neural Networks and Autoencoder Neural Networks. Therefore, the main contributions of this dissertation are divided into two sections as follows:

1. *Contribution 1:* Improving and Interpreting Self Organizing Neural Network (SOM)

    (a) A novel unsupervised Self Organizing Neural Network architecture for learning features of different resolutions in parallel layers: improve clas-

sification accuracy and generalizability

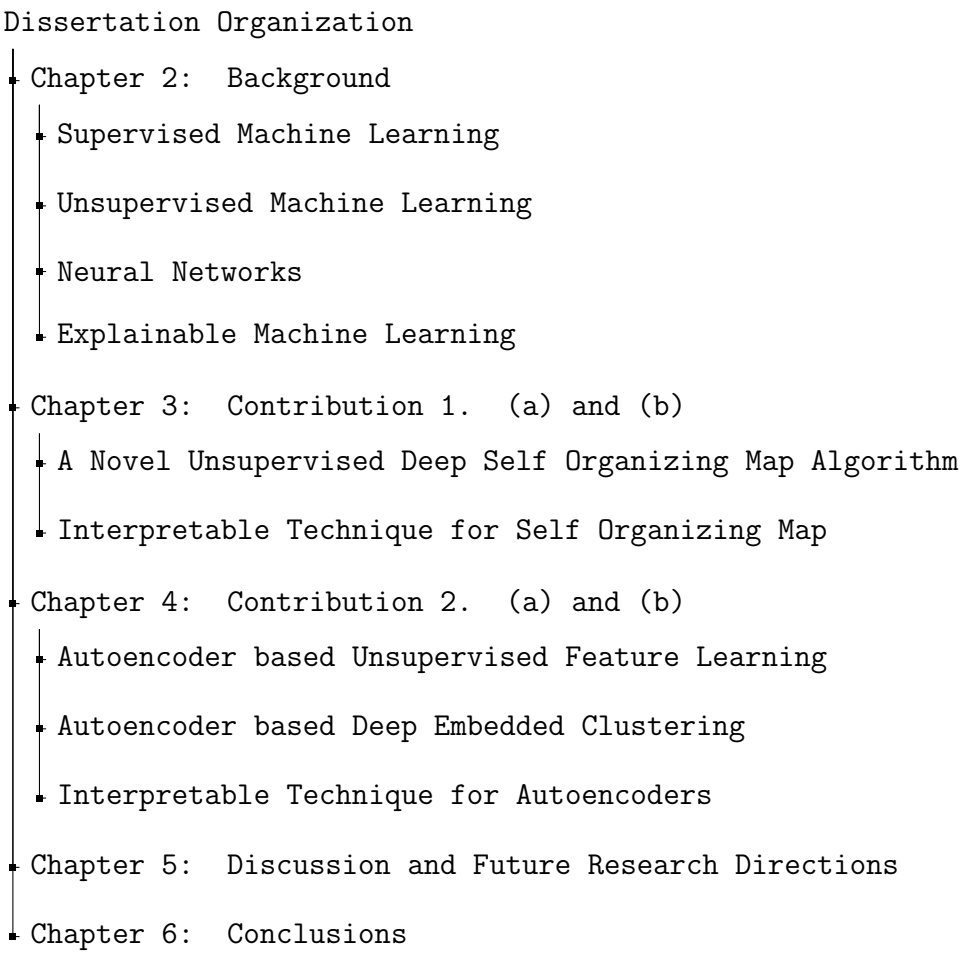
(b) A novel technique for interpreting Self Organizing Neural Network algorithm for unsupervised clustering

2. *Contribution 2:* Improving and Interpreting Autoencoder (AE) Neural Networks

(a) A deep Autoencoder Neural Network based framework for unsupervised feature learning and deep embedded clustering: improve robustness to network depth

(b) A novel technique for interpreting deep Autoencoder based framework for anomaly detection

1.4 Organization



The organization of the rest of the chapters is presented above. Chapter 2 discusses the relevant background and related work; Chapter 3 presents the Contribution 1. (a) and (b), which are the novel Self Organizing Map architecture and technique for interpreting Self Organizing Neural Network algorithm for unsupervised clustering; Chapter 4 presents Contribution 2. (a) and (b). Contribution 2. (a) consists of two sections, presenting the deep AE framework for unsupervised feature learning and AE framework for deep embedded clustering. Contribution 2. (b) presented as the third main section of Chapter 4, presenting the technique for interpreting deep AE based framework for anomaly detection. Chapter 5 presents the discussion and

possible future research directions, Chapter 6 concludes the dissertation objectives, contributions, and future work.

CHAPTER 2

BACKGROUND

In this chapter, we discuss relevant literature briefly. In the first two sections, we discuss two main areas of Machine learning; Supervised Machine Learning (SML), Unsupervised Machine Learning (UnML). Then we discuss Neural Networks and introduce different neural network architectures. Finally, we discuss current literature on XAI and its terminologies which are used within this dissertation.

2.1 Supervised Machine Learning

Supervised Machine Learning (SML) algorithms require prior knowledge of data to train them and make desired outcomes/predictions. SML is frequently used in data science due to its high predictive performance. However, the major drawback of these algorithms is that they can not be trained with unlabelled data. SML algorithms can be categorized into main areas, namely classification algorithms and regression, which are briefly explained below [15].

- Classification: Classification algorithms require class labels as categorical variables. Therefore, it limits the number of possible prediction outcomes to a finite set of categorical variables. Widely used classification algorithms includes Support Vector Machines, Decision Trees, Random Forest, Naive-Bayes, K Nearest Neighbor, and Supervised Neural Networks [16, 17] These algorithms can be further categorized into binary classification and multi-class classification. Binary classification algorithms categorize data samples into two classes, whereas multi-class algorithms can categorize data samples into more than two classes.

- Regression: Regression algorithms can take data labels as real value and predict real value as an output. Hence, the outcomes of regression algorithms can have an infinite number of values. Widely used regression algorithms include linear regression, logistic regression, and polynomial regression [17, 16].

2.2 Unsupervised Machine Learning

Unsupervised Machine Learning (UnML) has gained significant attention during the last decade. The main reason for this is the large amount of unlabelled data generated to the public. To use these data effectively and efficiently, it is crucial to analyze these unlabeled data (exploratory data analysis) to identify hidden patterns within them and reduce the amount of data for high-level tasks such as labeling through dimensionality reduction [18]. In this way, UnML provides initial insight into data allowing domain experts to use them appropriately.

The traditional concept of UnML was mainly limited to the idea of exploratory data analysis and dimensionality reduction. The expansion of deep learning methods and data mining, combined with this era of big data, has given a much broader perspective to traditional unsupervised learning. Therefore, unsupervised learning is used not only for clustering and dimentionality reduction [18], but also for generative modelling [19] [20], auto-regressive modelling [21] [22] and represntation learning (unsupervised feature learning) [23]. Some of the widely used application areas of UnML techniques are discussed below.

- Clustering: Clustering is one of the most common uses of UnML, where it organizes data into sensible groups based on similarities and characteristics of data [24]. This uses the same concepts as in classification tasks. However, this does not depend on labels to identify hidden patterns. Instead, this uses some similarity criteria to group data.

- Pre-trained models in transfer learning: This is the process of learning a machine learning model from a substantial amount of unlabeled data and using these pre-trained models for similar problem domains. These learned representations, have shown improved performance on downstream tasks for which the amount of data is limited, e.g., deep neural networks. [25].

- Unsupervised feature learning: This is the process of learning useful representations of data without manual annotations [26]. When the learned representation has a lower dimension than the input dimension, it is referred to as dimensionality reduction [27].

- Dimensionality reduction: This is the process of learning a low dimensional representation of the data set while preserving topological properties of data [28]. This low dimension can be either in the number of data points or the number of features in each data point.

- Association Rule Mining: This is the process of finding interesting associations (relationships, dependencies) in large sets of data items [29].

- Generative modeling: This is a typical use of unsupervised learning that models the probability distribution of data for generating new samples from the learned distribution [30]. These learned distributions are used to find good representations for large data sets and deal with missing data.

- Auto-regressive modeling: This is a process of time series modeling that uses previous observation from the previous timestamp as input to predict the value of the next timestamp [21].

2.3 Neural Networks

Artificial Neural Networks, also commonly referred to as Neural Networks (NNs), are a subset of machine learning. They are inspired by the human brain, mimicking the process of biological neurons or signaling to one another. In other words, it uses the computational principles of the nervous system to build artificial systems to achieve intelligence in machines. However, the learning or decision-making process of NNs is different from the brain as NNs only learn by extracting structure—statistical regularities -from input data (training examples), whereas the human brain depends on complex processes such as learned and innate mechanisms [31].

Starting from 1944, NNs have evolved over many years. McCulloch Pitt Neuron model is considered to be the first NN design. The building blocks of NN are 'artificial neurons,' which use a somewhat similar but primitive concept of a biological neuron. Like biological neurons, artificial neurons also receive input signals and produce an output signal. The output signal is calculated by calculating a weighted sum of inputs and which is transformed through a non-linear function (activation). These artificial neurons are arranged in multiple layers, making deep structures of NNs. Today, it has advanced to perform exceedingly complex tasks. Sometimes NNs perform better than humans: finding hidden patterns in large volumes of data, revealing complex relationships in data, and making fewer mistakes than humans. However, they also have some disadvantages, such as learning NNs requires massive amounts of data. Training of NNs is computationally expensive with high-dimensional and large volumes of data, making mistakes when fed with incomplete or mislabeled data, and a majority of NNs act as black-box models. Despite the disadvantages, NN application has shown state-of-the-art performance in many domains. Thus, research community is actively work on developing and improving NN models.

NN models consist of hierarchical architectures with many layers of neurons where higher level features are defined in terms of lower level features. They have the capability of extracting features and abstractions from underline raw data with minimal human involvement [32]. Figure 1 illustrates the overall idea of CPS and the use of NNs for CPSs. It shows examples of existing CPSs, what kind of features can be extracted from such systems, possible NN models and advantages of using NNs. Further, the data collected from CPSs is typically high dimensional. NN models are specifically designed to deal with high-dimensional data. Many researchers have experimentally shown that these architectures are capable of yielding outstanding results in many applications in cyber-physical systems domain [33] [34].

Neural Networks can be divided into two main categories; Shallow NNs and Deep NNs. Figure 2 presents important features that characterize the deep NNs and shallow NNs. Typically, Shallow architectures refer to models with only very few (usually one) hidden layers, whereas deep architectures are composed of several hidden layers [35]. These methods are capable of representing more abstract representations of data due to the multi-level architecture. In many practical applications, deep learning models have shown better generalization capability than shallow NNs and maximize unstructured data utilization. Other advantages of deep NNs include: it is computationally cheaper to add layers (deeper) than to add units (shallow), Worry less about feature engineering with deep NN as the hierarchy of concepts allows NNs to learn complicated concepts by building them out of simpler ones and can represent functions with increased complexity. However, the relative simplicity of shallow ANNs translates to a better understanding of shallow architectures compared to deep models. Few neural network architectures are discussed here;

- Deep Feed Forward Neural Networks:

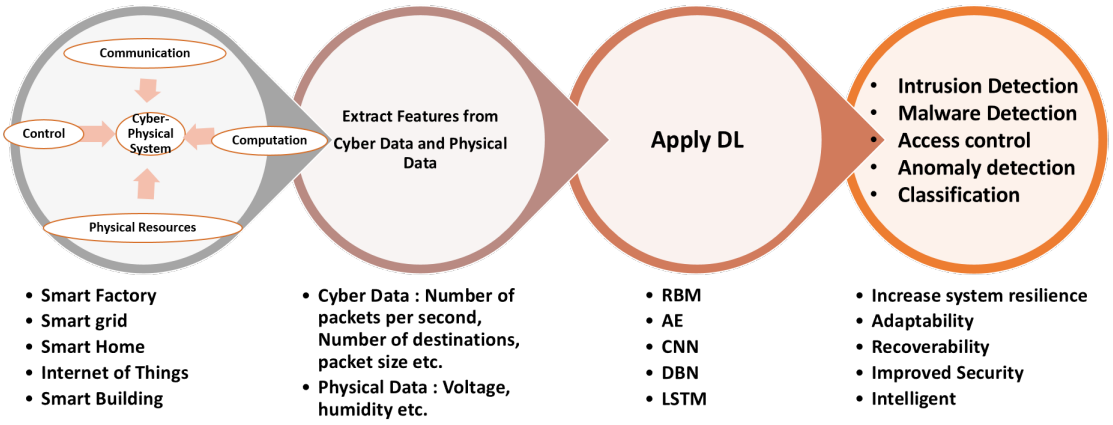


Fig. 1. Application of DL for CPSs

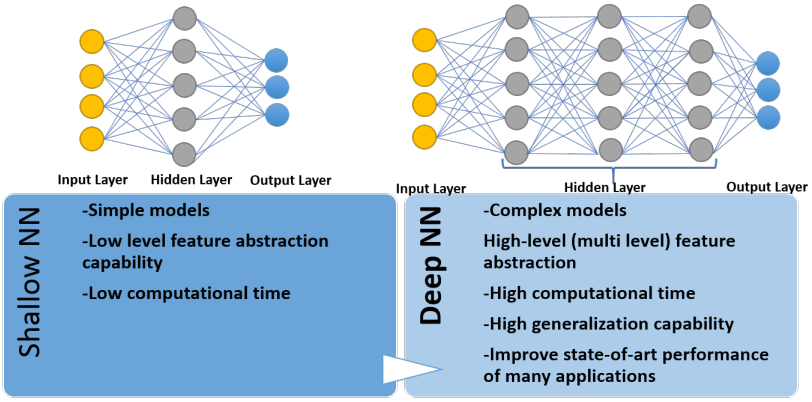


Fig. 2. Shallow vs Deep Neural Networks

These are often called Multilayer Perceptrons (MLPs) as they are made with combining many layers of perceptrons (another type of shallow machine learning algorithms) into a deeper structure. These models are called 'feed forward' because there are no feedback connections where the output of the network is fed back to the network MLPs have been successfully applied in many areas such as malware detection [36], intrusion detection [37] and access control systems

[38].

- Convolutional Neural Network (CNNs):

  CNNs are special kind of neural network for processing data with grid-like topology such as images and videos [39]. It combines three architectural ideas: local receptive fields, shared weights, and spatial subsampling to ensure some degree of shift and distortion invariance [40]. In cyber-security, it has been used for tasks like intrusion detection, classification and detection of malware variants [41] [42].

- Long Short-Term Memory:

  Long Short-Term Memory (LSTM) is a type of Recurrent Neural Network (RNN) proposed to solve the problem of vanishing and exploding gradient problem of conventional RNNs [38]. In cyber-security LSTMs have been used for tasks like classification and detection of malware variants [41] and anomaly detection [43] [42].

- Restricted Boltzmann Machines (RBMs):

  An RBM consist of two-layered undirected graphical models [44]. They are a stochastic model used to learn the underlying probability distribution of the dataset. They are used in many applications including image and speech recognition, dimensionality reduction, classification, feature learning, topic modeling and cyber-security. In cyber-security, it has been used for tasks like intrusion detection [45] , malicious code detection [46] and anomaly detection [43].

- Deep Belief Networks (DBNs):

  DBNs consist of a series of unsupervised multi-layered RBM networks (stacked RBMs) and a supervised back-propagation network [45][46]. DBNs are more

effective compared to other ANNs specially with unlabeled data [44]. They have been successfully used in many areas including image classification, speech recognition and information retrieval, natural language processing and cyber-security. In cyber-security, DBNs have been used for tasks like malicious code detection [46], intrusion detection [45] and anomaly detection [43].

- Autoencoders: Autoencoder (AE) structure is divided into two parts: encoder and decoder. The encoder converts the input data into an abstract representation which is then reconstructed using the decoder. They are widely used for the purpose of dimensionality reduction. In cyber-security, it has been used for tasks like malicious code detection [46], detection of malware variants [41] and anomaly detection [43].

2.4 Explainable Machine Learning

As we discussed in Section I, the effectiveness of AI systems was limited by the inability to explain its decision-making process to human users (black-box behavior) [47, 48, 49]. This has triggered a new research area named Interpretable Machine Learning or Explainable Artificial Intelligence (XAI). XAI focuses on making machine learning models with the ability to explain their rationale, characterize their strengths and weaknesses, and convey an understanding of how they will behave in the future. It allows to produce AI models with high-performance levels while allowing users to understand, trust, and effectively manage machine learning algorithms [47, 50]. Explainable AI research can take two main approaches: 1) developing novel explainable machine learning algorithms, 2) modifying the existing machine learning algorithms to make them understandable to humans.

Based on the literature, the need for XAI consists of four somewhat overlapping reasons [50]; *Explain to Justify, Explain to Control, Explain to Improve, and Explain*

to Discover. *Explain to Justify* refers to the need for reasons/justifications for a particular outcome, rather than providing a description of the inner workings or the logic of reasoning behind the decision-making process in general. It ensures that the AI-based decisions were not made erroneously. *Explain to Control* protect models from making wrongful outcomes by providing visibility of unknowns vulnerabilities, flows, and help to identify and correct errors through debugging. *Explain to Improve* refers to the fact that explainable and understandable models are easier to be improved. Since the user knows why the model produces certain outcomes and flows, users can make models smarter through continuous improvements. *Explain to Discover* refers to explaining to learn new facts, gather information, and gain knowledge. The learned pattern from machine learning models can result in some new and hidden knowledge revealed through explanations. Explainable machine learning is a diverse research area that consists of many components. Figure 3 presents a taxonomy of XAI and a list of common terms used in XAI. They are briefly described below.

- Intrinsic or Extrinsic (post hoc): This distinguishes whether the model itself is interpretable or needs to apply methods that analyze models after training to achieve interpretability [51]. Intrinsic refers to simple, explainable models such as short decision trees. Extrinsic refers to the use of an interpretation method after training to achieve interpretability.

- Model Specific or Model Agnostic: This distinguishes whether the interpretation method is limited to a specific model or not [51]. Model Specific refers to methods and tools which are specific to a model (Ex: regression weights in a linear model, tools only work for neural networks). Model Agnostic refers to methods that can be used on any machine learning model to achieve interpretability. These models do not have access to internal model details such as

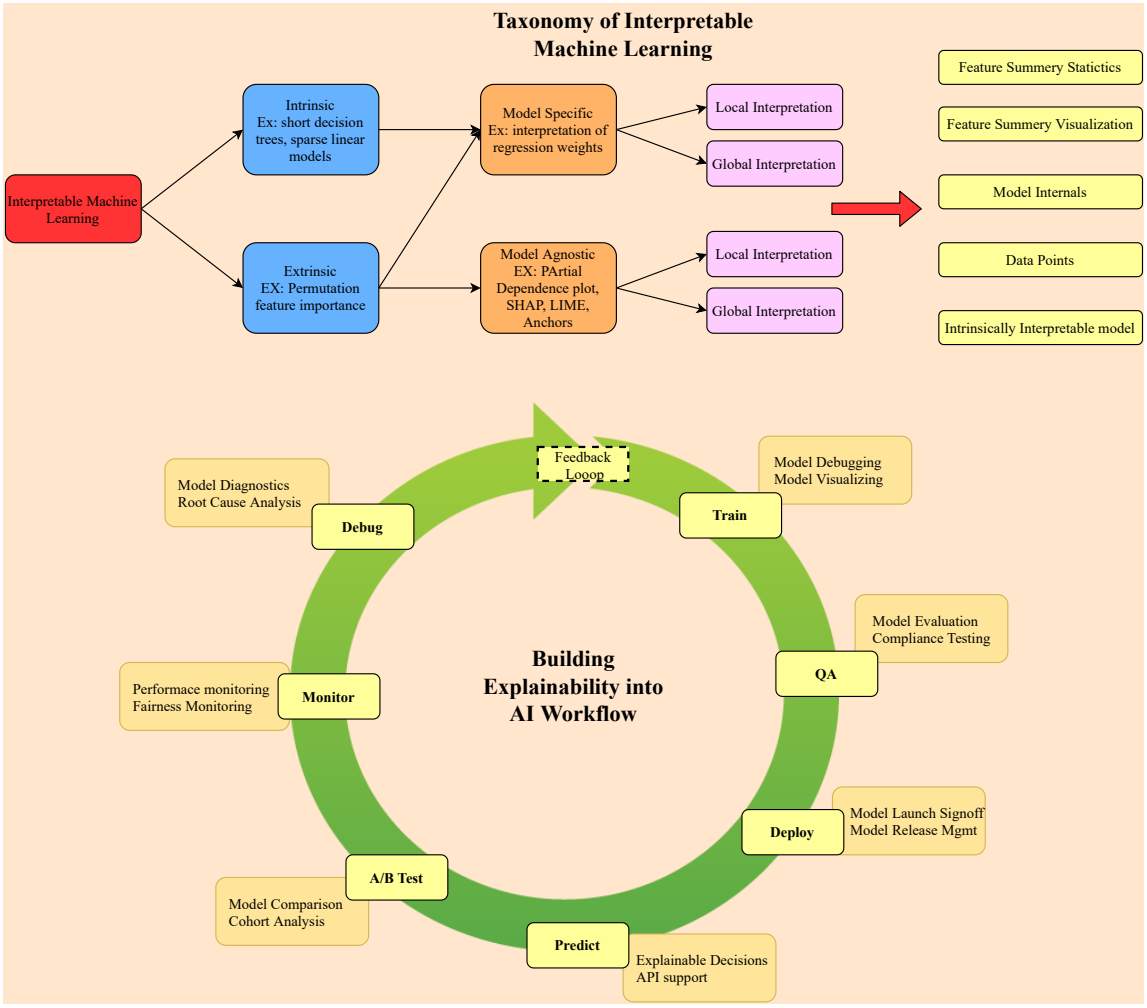


Fig. 3. XAI concept and taxonomy

weights or structural details.

- Local or Global: This distinguishes whether the interpretation method explains a specific data record or the entire behavior of a model [51]. Local refers to methods that explain specific prediction, whereas global refers to methods and tools which provide interpretation for the entire model.

- Result of the interpretation method: The various interpretation methods result in various interpretation outcomes. Some of them are listed below,

- Feature summary statistic: Interpretation methods can result in feature summary statistics for a single feature or multiple features together. For example, it can be a feature importance score for each feature or pair-wise feature importance.
- Feature summary visualization: Some feature statistics are meaningful only when presented visually. For example, partial dependence plots show the dependence between the output of the model and a set of input features. If this result is presented in tabular format, it is difficult to see the dependency between features and the model outcome.
- Model internals (e.g., learned weights): Typically, intrinsic interpretable models result in model internals such as learned weights in linear models and tree structure of decision trees.
- Data point: Some models' output already exists or newly created data points to make the model interpretable. For example, counterfactual explanation methods change the feature values of a data point to flip the class label of the data point.
- Intrinsically interpretable model: Some black box models can be interpreted using interpretable models. The result of this approach can be feature summary statistics or visualizations of the interpretable model.

Today, many companies such as Amazon, Google, NVIDIA, and IBM, and national institutes focus on adding explainability to the AI life cycle to ensure to ethical and fair algorithms for their users. Figure 3 present the idea of incorporating XAI into AI workflow, proposed by Fiddler lab, who is a member of NVIDIA inception. They point out that many people working within companies have no idea how to explain the inner workings of AI to customers. They are working towards bridging the

gap between hardcore data scientists who are building the models and the business teams using these models to make decisions.

CHAPTER 3

IMPROVING AND INTERPRETING SELF ORGANIZING NEURAL NETWORK

3.1 Contribution and Published Papers

This chapter presents the ***Contribution 1, (a) and (b)***;

a. A novel unsupervised Self Organizing Neural Network architecture for learning features of different resolutions in parallel layers: improve classification accuracy and generalizability

b. A novel technique for interpreting Self Organizing Neural Network algorithm for unsupervised clustering

Papers supports this work:

1. ©[2022] IEEE. Reprinted, with permission from **C. S. Wickramasinghe**, K. Amarasinghe, D. L. Marino, C. Rieger and M. Manic, "Explainable Unsupervised Machine Learning for Cyber-Physical Systems", in IEEE Access, vol. 9, pp. 131824-131843, 2021.

2. ©[2022] IEEE. Reprinted, with permission from **C. S. Wickramasinghe**, K. Amarasinghe, Milos Manic, "Deep Self-Organizing Maps for Unsupervised Image Classification", in IEEE Transactions on Industrial Informatics , vol. 15, no. 11, pp. 5837-5845, Nov. 2019.

3. ©[2022] IEEE. Reprinted, with permission from **C. S. Wickramasinghe**, K. Amarasinghe, M. Manic, "Parallalizable Deep Self-Organizing Maps for Image

Classification", in Proc. 2017 IEEE Symposium Series on Computational Intelligence, IEEE SSCI 2017, Honolulu, Hawaii, USA, Nov, 27- Dec 1, 2017.

4. ©[2022] IEEE. Reprinted, with permission from **C. S. Wickramasinghe**, K. Amarasinghe, D. Marino, and M. Manic, "Deep Self-Organizing Maps for Visual Data Mining", in Proc. 11th International Conference on Human System Interaction, IEEE HSI 2018, Gdansk, Poland, July 04-06, 2018.

5. ©[2022] IEEE. Reprinted, with permission from K. Amarasinghe, **C. S. Wickramasinghe**, D. Marino, C.Rieger, M. Manic, "Framework for Data-Driven Health Monitoring of Cyber-Physical Systems", in IEEE Resilience Week (RW) 2018, Denver, CO, USA, Aug 20-23, 2018.

3.2 Introduction

In this era of industrial big data, a massive amount of data is available to the public through various industries such as intelligent transportation [52] [53], power grids [2], cloud computing [54], and finance [55]. Successful mining of these data (classification, clustering) can lead to several advantages including process optimization, fault diagnosis, and improved cyber-security [56], [57, 58, 59, 60, 61]. In classification tasks, deep learning algorithms such as Deep Convolutional Neural Networks (CNN) have shown unprecedented performance [62, 55]. Recent attempts have focused on improving the efficiency of these algorithms by developing light-weight deep neural networks [63]. Despite many advantages, one major drawback of these state-of-the-art classification algorithms is that they are dependent on the availability of large labeled datasets. The scarcity of labeled data in the real world is a major hurdle to deploy supervised models in the real-world [57], [64, 65, 66]. Therefore, unsupervised approaches are ideal to leverage the abundantly available unlabeled data in industrial

applications [57], [65], [66].

Several unsupervised classification methodologies have been explored in literature such as Bayesian hierarchical clustering [67], [68] and Markovian models [69]. In more recent attempts, specialized deep learning methodologies such as Spiking Neural Networks (SNN) [70] and Generative Adversarial Nets (GANs) [71] were proposed. These algorithms have shown comparable performance with supervised algorithms for the MNIST dataset [70]. However, these models have some limitations when it comes to deploying them in the real-world. For example, the complex architecture of SNNs leads to low understandability and requirements of specialized hardware to deploy [72], [73]. Similarly, GANs suffer from poor interpretability and it has been shown that they suffer from high training times [74]. In addition, deep learning methodologies such as stacked convolutional Autoencoders (CAEs) have shown much promise in unsupervised learning for image classification [55], [75].

In this work, we focus on using Self-Organizing Maps (SOMs) based methodology for classification and clustering using unsupervised learning. The Self-Organizing Map is a widely used unsupervised learning algorithm capable of mapping a high-dimensional data distribution onto a low-dimensional grid while preserving important topological, and metric relationships of the input data [76, 77, 78]. It consists of a topological neuron grid (typically 2D or 3D), with each neuron consisting of a weight vector. It adapts its neuron weight vectors to represent topological properties of input data using the unsupervised "winner-take-all" learning algorithm [79, 80]. Since SOMs can represent topological properties of input data, they have been widely used for visual data mining and dimensionality reduction [81, 82]. Other advantages of SOMs include ease of optimization [83], the better capability of revealing overlapping structures in clusters compared to other traditional clustering methods, and suitability for visualizing and mining high dimensional data [84]. SOMs have been

successful in many areas, including speech recognition, robotics, telecommunication, and process optimization [85, 86, 83, 79].

Due to the above-discussed advantages of SOM, the first part of this work focus on improving the feature learning capability of SOM by introducing novel SOM architecture. As we discussed in the introduction, it is also essential to focus on explainable techniques for unsupervised ML methods. Therefore, the second part of this work also focuses on developing a novel interpretable technique for Self Organizing Map (SOM) based clustering. We present a model-specific interpretation method that identifies the most important features used by the decision-making process of the SOM algorithm. Further, it generates global and local interpretations for identified clusters and data records. Therefore, the rest of this chapter presents the first contribution of the dissertation, with its two sub contributions,

a. A novel unsupervised Self Organizing Neural Network architecture for learning features of different resolutions in parallel layers: improve classification accuracy and generalizability (Section 3.3)

b. A novel technique for interpreting Self Organizing Neural Network algorithm for unsupervised clustering (Section 3.4)

The rest of the section is organized as follows. Section 3.3 presents the novel unsupervised deep Self Organizing Map algorithm with improved feature learning capability; Section 3.4 presents the novel interpretable technique for Self Organizing Map based clustering; and finally, Section 3.5 provides a summary of the first contribution of this dissertation.

3.3 Unsupervised Deep Self Organizing Map algorithm

As we discussed in the Introduction, SOMs have many advantages including visual data mining capabilities [87, 88, 89], ease of optimization [90], and better capability of revealing overlapping structures in clusters compared to traditional clustering methods [91]. Thus, SOMs have been successful in a multitude of areas including speech recognition, robotics and process control [92], [93, 94]. The major drawback of SOMs is its limited capability of high-level feature abstraction due to the shallow structure [95].

One of the recent attempts at alleviating this limitation was to explore a deep architecture of SOMs, named Deep Self-Organizing Maps (DSOM) by Liu et al [88]. Since DSOM architecture uses the same learning mechanism as SOMs, it inherits all the advantages of SOMs mentioned above. However, the authors of [88] explored a supervised learning algorithm with DSOM and thus relied on the availability of labeled data. In our work, we explore a parallelized version of an unsupervised DSOM architecture. An accurate unsupervised DSOM architecture has the following main advantages: 1) the ability to leverage unlabeled datasets, 2) hierarchical feature abstraction based unsupervised learning, and 3) the ability to deploy without special hardware.

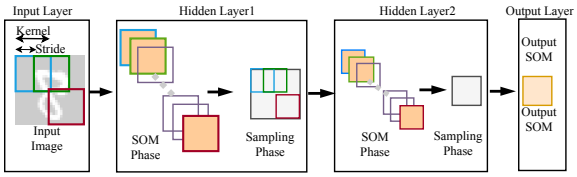


Fig. 4. Two layered DSOM architecture used for handwritten character recognition [88]

This section presents the proposed novel DSOM architecture, referred as Enhanced DSOM (E-DSOM). E-DSOM enhances the DSOM in two ways: 1) the learn-

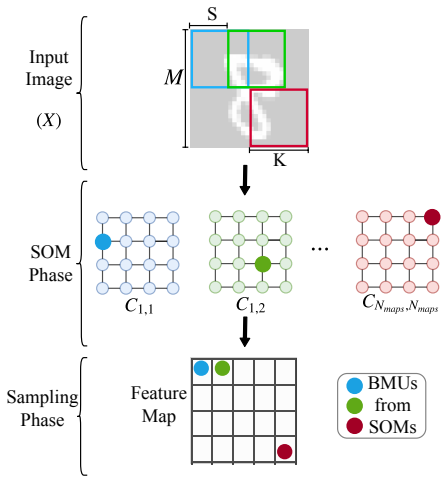


Fig. 5. Sampling layer creation in DSOM

ing algorithm is completely unsupervised and 2) the architecture learns features of different resolutions in parallel in a single hidden layer. Please note that in the rest of the chapter, DSOM refers to the architecture proposed by Liu et al. in [88] and E-DSOM refers to the architecture presented in this chapter. The main contributions of this work are summarized as follows:

1. An unsupervised, easy to understand, easy to implement deep SOM architecture for classification. The goal of this work is not to improve on the accuracies of other supervised learning architectures such as CNN. The goal is to present an unsupervised learning methodology, with high-level feature abstraction capability.

2. A deep SOM architecture capable of learning features of different resolutions simultaneously. We hypothesize that this capability improves classification accuracy and the generalization capability of the model. Further, we hypothesize that this will result in a shallower model compared to the DSOM and lead to reduced training times.

3.3.1 Background

This section provides the background information needed to present the E-DSOM algorithm. First, the single-layered SOM is introduced. Then, the DSOM algorithm proposed by Liu et al. in [88] is presented.

Table 1. Algorithm for training the Self-Organizing Map

Algorithm I: SOM Training	
Inputs: Training set of images (X)	
Outputs: Trained SOM	
1:	Random Weight initialization
2:	**for** each epoch e do
3:	**for** number of training samples do
4:	$x \leftarrow$ pick random input record from X
5:	$md \leftarrow$ initialize to the largest float
6:	**for** number of neurons in SOM do
7:	$d_i \leftarrow \|X - W_i\|$
	% find the BMU
8:	**if** $d_i < md$ do
9:	$BMU_x \leftarrow W_i$ % weight of BMU
10:	$BMUIndex_x \leftarrow i$ % index of BMU
11:	$md \leftarrow d_i$
12:	**end if**
13:	**end for**
	% update weights
14:	**for** number of neurons in SOM neighborhood do
15:	$n \longleftarrow e^{-\left(\frac{BMU_x - w}{2\delta t^2}\right)}$
16:	$\triangle W_i \longleftarrow W_i \times \alpha \times \eta \times (x - w)$
17:	$W_i \longleftarrow W_i + \triangle W_i$
18:	**end for**
	% Decay the neighborhood and learning rate
19:	**end for**
20:	**end for**

3.3.1.1 Self-Organizing Maps

SOMs consist of a topological neuron grid (typically 2D) with each neuron consisting of a weight vector. The SOM adapts itself to the topological properties of input data using the unsupervised "winner-take-all" learning algorithm. Both DSOM and E-DSOM use this as the underlying learning mechanism in the hidden layers.

The learning algorithm for a SOM is given in Algorithm I (Table 1). For each input pattern, the SOM selects the neuron that best matches the pattern in terms

of Euclidean distance. This neuron is called the Best Matching Unit (BMU). Then, the SOM updates weights of the neurons in the neighborhood of the BMU so that they move closer to the BMU (line 14-18 in **Algorithm I**). The learning rate and the radius of the BMU neighborhood are used as the controlling hyper-parameters.

The learning rate and the neighborhood radius is decayed with time. The neighborhood radius is halved at each epoch. The learning rate is decayed as follows:

$$\eta(t) = 0.49\Big(1 - \frac{e}{epochs}\Big) + 0.01 \tag{3.1}$$

Where e is the current epoch and $epochs$ is the total number of epochs. The most important hyper-parameter of the SOM is the size of the map. If the map size is too small, it will lead to the model not capturing the feature space adequately (under-fitting). Conversely, if the map size is too large, it will lead to over-fitting the training data and unnecessary computations.

3.3.1.2 Deep Self-Organizing Maps

The DSOM is a multi-layered SOM architecture, which consists of an input layer, hidden layers, and an output layer. The initial design of DSOMs merged the concepts of SOMs and Convolution Neural Networks (CNNs). SOMs provided the underlying learning mechanism to DSOM while CNNs inspired the high-level feature abstraction process.

In CNNs, in each hidden layer, each unit (neuron) receives inputs from a subset of units in the preceding layer (local receptive field/patch) [96], [97]. The lower-level features learned in the preceding layer are combined in the current hidden layer to generate higher-level features. This idea was incorporated into the DSOM architecture so that higher-level layers are capable of learning more abstract information than lower-level layers.

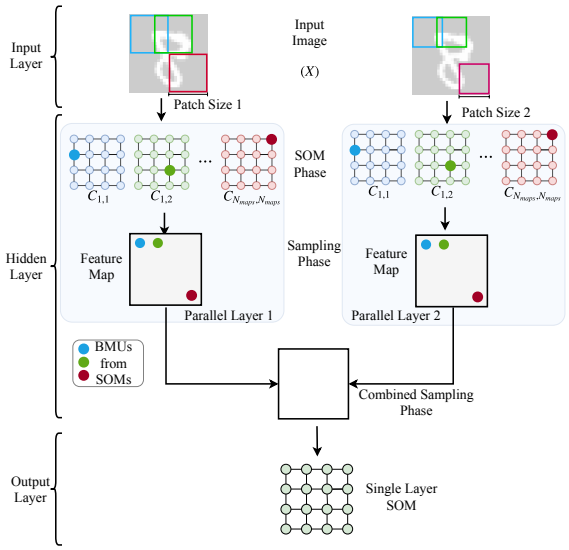


Fig. 6. E-DSOM architecture with one hidden layer (two parallel layers in the hidden layer)

Although DSOM uses local receptive fields, the function of hidden layers is completely different from CNN. In CNN hidden layers, a convolution operation followed by a pooling step generates the feature map for the next layer. Conversely, hidden layers in DSOM process the patches with SOMs and aggregate the BMUs to generate the feature map. Therefore, the only similarity between DSOM and CNN is the notion of the local receptive field. Figure 4 shows a DSOM architecture with two hidden layers, which was used for MNIST classification by Liu et al [88].

The function of each layer in the DSOM can be summarized as follows:

Input layer: Forwards the input images to the DSOM

Hidden Layer: Each hidden layer consists of two phases: 1) SOM phase and 2) sampling phase. In the SOM phase, each input image is segmented into smaller local regions (patches). Then, each patch is sent to its own SOM unit in this layer, i.e. each patch is processed by its own SOM. Each SOM finds the BMU for the input patch using the Algorithm I. In the sampling phase, the BMUs of the hidden SOM

units are combined to generate a single 2D grid (see Figure 5). This 2D grid acts as the input image (feature map) to the next hidden layer. This process is repeated for all hidden layers.

Output layer: The output layer contains a single SOM. It receives the feature map generated by the last hidden layer. The output SOM extracts abstract and pertinent information for classification.

3.3.2 Novel Deep Self-Organizing Maps

This section presents the novel DSOM (E-DSOM) architecture, its unsupervised learning algorithm, classifier implementation and a discussion on computational complexity.

3.3.2.1 E-DSOM Architecture

Similar to the DSOM architecture, the E-DSOM consists of an input layer, hidden layers, and an output layer. The main differences in the E-DSOM architecture are in the hidden layers. As opposed to the DSOM, E-DSOM hidden layers contain several parallel layers (See Figure 6). Each parallel layer has its SOM phase and sampling phase. In the sampling phase, first, a feature map for each parallel layer is created. Then, those feature maps are combined to generate one feature map.

E-DSOM uses different sized patches (multi-scale patches) in parallel SOM layers of the hidden layers. It has been shown that multi-scale patch approaches help to improve classification accuracy by extracting complementary information [98, 99]. Figure 6 shows an E-DSOM architecture with two parallel layers with different patch sizes.

The above architecture modification enables the algorithm to learn feature spaces of different size and resolution by using different map sizes and patch sizes in the

parallel layers. We hypothesize that this ability will 1) improve classification accuracy, 2) improve generalization capability, and 3) reduce the need for sequential hidden layers (reduce training time). In this work, more emphasis is laid on changing the patch size, i.e., changing the size of the local region of focus to enable the learning features of different resolutions.

Table 2. Algorithm for training the E-DSOM

Algorithm II: E-DSOM Training

Inputs: Training set of images (X)
Outputs: Trained E-DSOM

Random Weight initialization
for each epoch e do
  for number of training samples do
    $x \leftarrow$ pick random input record from X
    for each hidden layer l do
      $featureMapList \leftarrow empty\ list\ of\ length\ P$
      for each parallel SOM layer p do
        $featureMapList[p] \leftarrow ParallelLayer(x)$
      end for
      $x \leftarrow CombinedSampling(featureMapList)$
    end for
    OutputSOM $\leftarrow$ Algorithm I (x)
    % Find the BMU for (x) using SOM algorithm
  end for
end for

Procedure I: ParallelLayer

Inputs: Input record (x), Number of patches (p)
Outputs: Sampled $featureMap$

$featureMap \leftarrow$ empty list of length p
for each patch $x^{'}$ do:
  $index_x \leftarrow$ the location of $x^{'}$ w.r.t. x
  $BMU_{x^{'}} \leftarrow$ get BMU index for $x^{'}$ on corresponding $SOM_{x^{'}}$
  $featureMap[index] \leftarrow BMU_i ndex$
end for

Procedure II: CombinedSampling

Inputs: List of feature maps from each parallel layer ($featureMapList$)
Outputs: Combined Feature Map

$comFeatureMap \leftarrow$ Append $featureMapList$ to a single list
$l \leftarrow$ length of $comFeatureMap$
if $\sqrt{l} \notin \mathcal{N}$ **for**; $\mathcal{N} = \{1, 2, 3, 4, ...\}$
  Use zero-padding on $comFeatureMap$ until $\sqrt{l} \in \mathcal{N}$
$CFM \leftarrow$ Reshape $comFeatureMap$ to a 2D vector of size $\sqrt{l} * \sqrt{l}$
return CFM

3.3.2.2 Training the E-DSOM

Training algorithm of the E-DSOM is presented in Algorithm II (Table 2). Similar to the SOM and DSOM, weights of the network are randomly initialized. In a hidden layer, the SOM phase consists of P parallel SOM layers with P different patch sizes (Algorithm II, lines 7-9). For each patch size, the number of patches along one dimension is calculated as follows:

$$N_{map} = ceil\Big(\frac{M-K}{S}\Big) + 1 \tag{3.2}$$

where $ceil(\cdot)$ calculates the smallest integer upper, M is the pixel width/height of the input image X ($M \times M$ image) K is the width/height of the patch ($K \times K$ patch) and S is the stride of the patch. Therefore, $N_{map} \times N_{map}$ number of patches are created form the input image for each patch size (see Figure 5), i.e. $N_{map} \times N_{map}$ number of SOMs are created for each parallel layer.

In all the parallel SOM layers, the BMU selection for its respective patches is carried out followed by its sampling process(see Procedure I). Therefore, P feature maps are created (see Algorithm II) [88]. All feature maps are converted to one-dimensional arrays and concatenated into a single array. Then, the resultant array is reshaped to a 2D grid which acts as the input image to the next hidden layer (see Procedure II).

After processing the hidden layers, the combined feature map generated from the last hidden layer acts as the input to the output SOM. The output layer SOM is trained using Algorithm I.

3.3.2.3 Classifier

A classifier is implemented based on the trained output layer SOM to assign the class labels to the input records. It has to be noted that E-DSOM algorithm is trained purely unsupervised, without using any prior knowledge about class labels.

The classifier requires some labeled data. Each neuron in the output layer SOM is assigned a class label using the labeled dataset. First, the labeled dataset is processed through the trained E-DSOM. For each neuron j in the output layer, the number of times it was selected as the BMU for each class is stored as, $N_{BMU}^{j,c}$ where c is the class label. The class label with the highest BMU frequency is assigned as the neuron label:

$$label_j = \underset{c}{\operatorname{argmax}}\, N_{BMU}^{j,c} \tag{3.3}$$

In case of a tie one of the tied labels of maximum $N_{BMU}^{j,c}$ values, is assigned at random.

Once each output SOM neuron is assigned a class label, test data can be classified using the E-DSOM. When an input data record (image) is processed through the E-DSOM, the label of its BMU in the output layer is assigned to the data record.

3.3.2.4 Computational Complexity

As mentioned, each hidden layer consists of multiple parallel layers where each patch is processed by a separate SOM (Procedure I), i.e., for each patch, **Algorithm I** is used to find the BMU and the BMU_{index} is stored in its feature map (Algorithm II steps 7 to 9). The Algorithm I executes in two phases. Phase 1 calculates the Euclidean distances between the input vector x and the SOM units and finds the best matching unit (BMU). Phase 2 updates the neuron weights in the BMU neighborhood.

The computational complexity of each phase in the E-DSOM hidden layer can be expressed as follows:

$$O(K^2N^2N_{map}^2) \tag{3.4}$$

Where K^2 is the number of elements in a single patch, N^2 is the number of units in a SOM and N_{map}^2 is the number of patches/SOMs. Both phases are highly parallelizable. Therefore, traversing the SOM for distance calculation and weight update can be reduced to $O(1)$ in the ideal case. Therefore, for a highly parallelized ideal implementation, the above computational complexity can be reduced to:

$$O(K^2N_{map}^2) \tag{3.5}$$

Table 3. The DSOM Architecture

	Hidden Layer 1			Hidden Layer 2			Output Layer
Dataset	Map Size	Patches (K)	Stride	Map Size	Patches (K)	Stride	Map Size
MNIST	4–24	10—20	2	15	6	1	8
GSA	4–24	3-–7	2	14–16	3–5	1	5–8
SP-HAR	4–24	5-–17	2	14–16	3–7	1	5–8

Table 4. The E-DSOM Architecture

Dataset	Hidden Layer 1			Output Layer
	Map Size	Patches (K)	Stride	Map Size
MNIST	4-24	10-20	2	8
GSA	4-24	3-7	2	5-8
SP-HAR	4-24	5-17	2	5-8

In the combined sampling phase of a hidden layer, all the feature maps from P parallel layers are combined into one (Procedure II). Concatenating these arrays take linear computational complexity. Therefore, the computational complexity of creating the combined sampling map can be expressed as follows:

$$O(N_{map}^2P) \tag{3.6}$$

The P parallel layers can be executed in parallel. Therefore, increasing the number of parallel layers very little effect on the computational time is given in Eq.

(5). However, it does affect Eq. (6). When the number of parallel layers (p) increase, the time taken to combine them into a combined sampling layer increase linearly.

When considering the space complexity of the E-DSOM model, the number of parameters that need to be stored per hidden layer can be approximated as follows:

$$|\theta| = PK^2N^2N_{map}^2 \tag{3.7}$$

The number of parameters that need to be stored linearly grows with the number of hidden layers. This can be used to infer the space complexity of the model. Unlike time complexity, space complexity grows with the increase of parallel layers.

3.3.3 Experiments and Discussion

This section discusses the experiments and results. The experimental setup is presented followed by a comparative analysis against DSOM and other state-of-the-art unsupervised algorithms.

3.3.3.1 Datasets

Three datasets were used for experimentation: 1) MNIST [100], 2) Gas Sensor Array Drift (GSAD) dataset [101], and 3) Smart Phone dataset for Human Activity Recognition (SP-HAR) [100] . All the datasets were normalized to zero mean and unit variance. For all datasets, balanced subsets of the data records were selected to alleviate the class imbalance problem. Further, data records in numerical datasets (GSAD and SP-HAR) were converted into 2D square images.

The MNIST dataset contains images of hand-written characters (digits from 0-9), each 28×28 pixels in size. The complete MNIST dataset contains 55000 train images and 10000 test images. In this work, a significantly smaller training set of 3000 images was used to reduce the classifier training time. The complete testing set

(10000 images) was used to test the accuracy of the algorithms.

The GSAD dataset contains 13910 records collected from 16 chemical sensors from a gas delivery facility. The dataset contains data about 6 gases and the classifier's goal is to discriminate between the gasses. The dataset contains data collected for 36 months. In this work, only the first 21 months were used to avoid concept drift in data. A balanced dataset of 4500 records was selected and the train/test split was chosen as 2400/2100. The sensor data were rearranged to a 2D grid and is processed as an image. Since each data record consists of 121 dimensions, each data record was arranged to an 11×11 image.

SP-HAR consists of 10299 smartphone sensor records of 30 subjects performing six different daily living activities. A balanced dataset of 4792 records was selected and the train/test split of 3300/1492 was chosen. Similar to the GSAD dataset, data were rearranged to a 2D grid. Since the dataset contained 561 dimensions, the features were reduced to the closest square number (529) using information gain based feature selection. Then, each record was re-arranged into a 23×23 image.

Table 5. Classification Accuracy comparison between DSOM and E-DSOM

Dataset	Model	Layer1						Test Accuracy for Different Noise level (%)						
		Patch Scale1	Patch Scale2	Map Size1	Map Size2	Train Acc	Test Acc	2	5	10	20	40	50	60
MNIST	DSOM	10	-	24X24	-	85.06± 1.94	83.47± 2.85	83.37± 3.04	83.14± 2.81	83.14± 2.68	82.39± 2.63	74.46± 2.72	62.00± 3.25	20.37± 1.55
	E-DSOM	10	20	24X24	24X24	**88.04± 1.96**	**87.12± 2.41**	**87.12± 2.35**	**87.15± 2.14**	**86.88± 2.17**	**86.51± 1.87**	**79.91± 1.77**	**69.34± 1.98**	**23.63± 1.71**
GSAD	DSOM	3	-	24X24	-	83.35± 2.73	57.24± 8.63	49.76± 6.19	45.20± 3.18	38.08± 1.93	32.59± 1.45	27.12± 0.83	23.84± 0.45	21.88± 1.27
	E-DSOM	3	9	24X24	24X24	**91.24± 1.19**	**72.73± 6.78**	**66.82± 5.06**	**61.19± 3.90**	**50.01± 4.89**	**37.86± 3.34**	**28.59± 3.64**	**24.12± 2.03**	**22.45± 2.41**
SP-HAR	DSOM	11	-	24X24	-	62.85± 1.08	57.88± 2.31	56.90± 3.28	55.60± 2.37	52.58± 2.32	44.81± 2.12	**27.17± 2.92**	19.52± 2.19	**17.78± 0.76**
	E-DSOM	11	17	22X22	22X22	**67.39± 1.12**	**64.36± 0.28**	**63.22± 1.22**	**61.90± 0.83**	**58.22± 1.57**	**48.51± 2.69**	24.14± 1.46	**19.69± 0.88**	17.35± 1.01

Table 6. Generalization error comparison between DSOM and E-DSOM

Dataset	Model	Generalization Errror (%) for different Noise Levels								Computational Time (s)
		0	2	5	10	20	40	50	60	
MNIST	DSOM	1.59	1.69	1.92	1.92	2.67	10.60	23.06	64.69	3788
	E-DSOM	**0.92**	**0.92**	**0.89**	**1.16**	**1.53**	**8.12**	**18.69**	**64.41**	**3114**
GSAD	DSOM	26.10	33.59	38.15	45.27	**50.76**	**56.23**	**59.51**	**61.46**	390
	E-DSOM	**18.51**	**24.42**	**30.05**	**41.23**	53.38	62.65	67.12	68.78	**313**
SP-HAR	DSOM	4.97	5.95	7.25	10.27	**18.04**	**35.68**	**43.32**	**45.07**	3098
	E-DSOM	**3.04**	**4.18**	**5.49**	**9.18**	18.88	43.25	47.70	50.04	**2602**

3.3.3.2 Hyper-parameter and Model Architecture Selection

As mentioned in Section I, we hypothesize that due to the parallel architecture of E-DSOM, a shallower model compared to the DSOM can be used. This results in a reduction of serial operations, resulting in reduced training time. In order to test this, for all the tests, a DSOM with two hidden layers and an E-DSOM with only one hidden layer were implemented. In the E-DSOM hidden layer, two parallel layers were implemented.

Table 3 summarizes the architecture and the hyper-parameters chosen for DSOM for the three datasets. For the MNIST dataset, the set of hyper-parameters were selected based on the experiments done by Liu et al [88] and our previous work [102]. For the other datasets, the hyper-parameters were selected experimentally through cross-validation. For each dataset, different experiments were conducted by changing the map size and the patch size within the ranges shown in Table 3.

Table 4 presents details of the E-DSOM architecture. Different combinations of patch sizes and map sizes were tested within the presented ranges. Across parallel layers of the same model, different patch sizes were used, but the map size was kept the same. The shallower model of E-DSOM enabled the use of bigger patch sizes than the DSOM. In order to ensure a fair comparison of classification accuracies, the

output layer of DSOM and E-DSOM was implemented with a SOM of the same size.

3.3.3.3 Experimental Results: MNIST

Classification accuracies for the MNIST dataset is presented in **Table 5**. The DSOM was able to achieve the best test accuracy of 83.468% while E-DSOM was able to achieve 87.118 % (A 3.65% improvement).

Generalization capability was analyzed with noisy test data. **Table 5 and Fig. 4 (a)** show the performance of the two algorithms. There was no significant difference in classification accuracy for both models until the noise level increased beyond 20%. Despite the drop in accuracy beyond 20% noise, it was observed that E-DSOM consistently outperformed the DSOM. Further, E-DSOM showed a lower generalization error at all the noise levels **(see Table 6)**.

When **computational time** was compared **(Table 6)**, it was observed that the E-DSOM was able to reduce the training time by more than 670 seconds compared to the DSOM (17% improvement).

3.3.3.4 Experimental Results: GSAD

Classification accuracies for the GSAD dataset are presented in Table 5. DSOM achieved 57.24% as its best classification accuracy while E-DSOM achieved 72.73% (A 15.49% improvement).

Generalization capability: Classification accuracies with noisy data are presented in Table 5 and Figure 7 (b). E-DSOM outperformed DSOM at all noise levels. Further, the E-DSOM showed a lower generalization error at noise levels of 0%-20% (see Table 4).

When **computational time** was considered (Table 6), it was observed that the E-DSOM was able to finish training more than 70 seconds faster than the DSOM

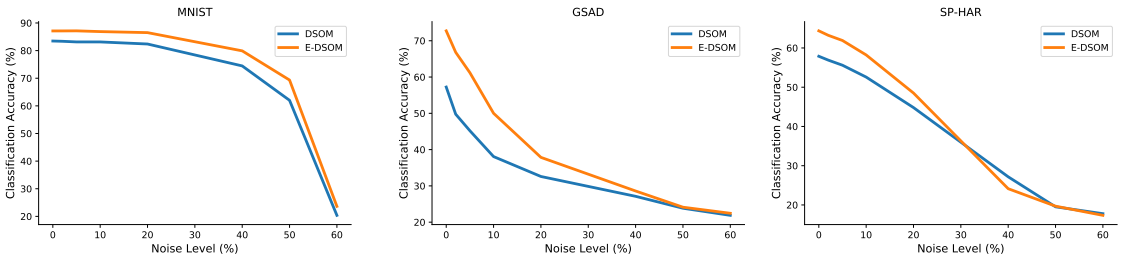
with GSAD dataset (19% improvement).

3.3.3.5 Experimental Results: SP-HAR

Classification accuracies for the SP-HAR dataset are presented in Table 5. DSOM was able to achieve a maximum test accuracy of 57.88% while E-DSOM was able to achieve 64.36 (6.48% improvement).

In terms of **generalization capability**, E-DSOM outperformed DSOM at all noise levels except at 40% and 60% (Table 5 and Figure 7(c)). Further, E-DSOM showed a lower generalization error for 0%-10% noise levels (see Table 6).

Table 6 shows the **computational times** of the two models. E-DSOM was able to finish training over 490 seconds faster than the DSOM for SP-HAR dataset (around 16% improvement).



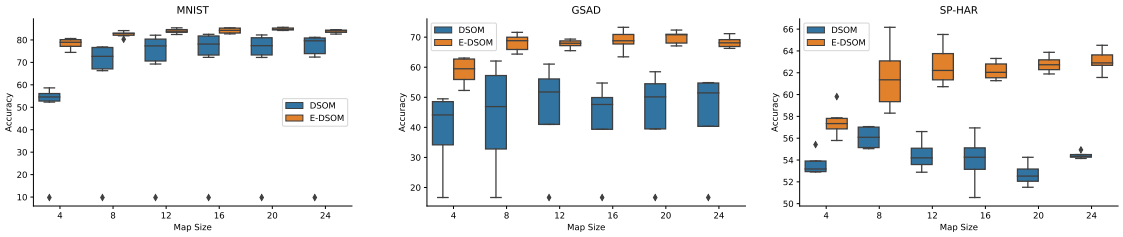
(a) (b) (c)

Fig. 7. Change in accuracy with different noise levels for E-DSOM and DSOM. (a) MNIST, (b) GSAD and (c) SP-HAR

3.3.3.6 Overall Results Discussion

For MNIST and GSAD datasets, E-DSOM showed superior performance in classification accuracy, generalization capability and computational time.

For the SP-HAR dataset, the E-DSOM achieved superior classification performance and reduced computational time. However, the E-DSOM failed to outperform



(a) (b) (c)

Fig. 8. Effect of patch size and map sizes on classification accuracy for E-DSOM and DSOM: (a) MNIST, (b) GSAD, (c) SP-HAR

the DSOM at noise levels beyond 40%, but scores remained comparable. SP-HAR dataset contains data from smartphone sensors, which can be less precise than the industrial grade sensors in GSAD. The noisy data could be the reason for the lower classification accuracies shown by both algorithms.

The overall classification accuracy results support our hypothesis that the E-DSOM architecture with parallel layers is able to achieve better/higher accuracy with a fewer number of serial layers compared to DSOM, i.e., using less computational time.

3.3.3.7 Analysis: Effect of patch size and map sizes on classification accuracy

The analysis studied the effect of the two most important hyper-parameters—patch size and map size—on the classification accuracy. We used square maps of the size range 4–24 and different square patch sizes with each map size. For simplicity, the parameters were changed only in the first layer of both algorithms.

For MNIST and GSAD datasets, patch sizes in the range of 10–20 pixels (per dimension) and 3–7 pixels were used, respectively. The patch size was incremented in two pixels between tests. Patch sizes for SP-HAR were kept within the range of 5-17 and incremented in four pixels between tests (See Table 5 and 6).

The results from the analysis are given in box and whisker graphs (See Figure 8(a) –(c)). Each box plot relates to specific map size. The height of the box plot indicates the variability of classification accuracy for the different patch sizes, i.e., a shorter box plot indicates low variability classification accuracies and vice-versa.

Effect of Map Size: Classification accuracies were observed for different map sizes. For all tests, E-DSOM outperformed the DSOM in classification accuracy. Further, for all datasets, if the map size wasn't very low, E-DSOM's classification accuracies remained consistent across map sizes. Conversely, DSOM showed significant variations in its classification accuracies across map sizes with the exception of MNIST. Further, the smallest map size yielded the smallest classification accuracy for both models. This is expected as a small map can be inadequate to capture the feature space.

Therefore, from these datasets, it can be inferred that for the E-DSOM, as long as the map size is not too small, the classification accuracies will not change much with the map size. However, with the DSOM, in order to find the optimal map size, a thorough cross-validation process is needed.

Effect of Patch Size: As mentioned, for each map size, several patch sizes were tested. The E-DSOM consistently outperformed the DSOM despite different configurations. With the exception of SP-HAR dataset, the classification accuracies remained fairly consistent across different patch sizes with E-DSOM (shorter box plots). However, in DSOM, the results varied significantly across patch sizes for a single map size (taller box plots). In the SP-HAR dataset, the E-DSOM algorithm showed some variability for the patch sizes when the map size was 8 and 12. Therefore, it can be inferred that generally, the E-DSOM algorithm shows less dependency on the patch sizes when compared to the DSOM. This leads to an easier process of hyper-parameter selection. This could be a result of E-DSOM balancing out the effect of

patch size by detecting complementary features of different resolutions in the parallel layers.

3.3.3.8 Comparison of E-DSOM with other unsupervised algorithms

The proposed E-DSOM architecture was compared against three other unsupervised algorithms: 1) single layer SOM, 2) stacked Autoencoder and 3) stacked Convolutional Autoencoder. A single layer SOM with an 8×8 neuron grid was implemented for the completeness purpose.

Stacked Autoencoders (AE) are deep unsupervised learning architectures [55]. AE consists of two functions, an encoder, and a decoder. Encoder learns a compressed representation of the input data and decoder reconstructs the input data using the compressed representation. AEs are widely used for dimensionality reduction [103], feature learning and data denoising [75]. In this work, AEs was implemented with a SOM (8×8) connected to the last hidden layer. The SOM was trained with the encoded data, and the same classifier as E-DSOM was implemented. AEs with up to three hidden layers were tested and the best classification results are reported.

Stacked Convolutional Autoencoders (CAEs) are a variant of AEs that contains convolutional layers. CAEs are unsupervised learning algorithms, which use the building blocks—convolution layers and max-pooling layers—of supervised CNNs [103]. Similar to AE, CAE was implemented with up to three hidden layers followed by a SOM classifier. The number of filters was changed within the range 4–30. The kernel size was set to 3×3 as it resulted in the best classification accuracies. ReLU activation function was used for the non-linear transformations.

Table 7 presents the test accuracy comparison between algorithms. For **MNIST**, E-DSOM achieved the best accuracy while AE came in second. Single layers SOM showed the lowest accuracy for the MNIST. For **GSAD**, E-DSOM yielded the best

Table 7. Comparison of test accuracies of unsupervised algorithms

Dataset	Test Accuracy (%)				
	SOM	DSOM	E-DSOM	Stacked AE	Stacked CAE
MNIST	71.26	83.47	**87.12**	84.24	81.93
GSAD	66.62	57.24	**72.73**	63.59	70.12
SP-HAR	62.80	57.88	64.36	**67.41**	66.47

accuracy while CAE showed the second best accuracy. DSOM showed the lowest accuracy for the GSAD dataset. For the **SP-HAR** dataset, the AE and CAE came in first and second respectively, in terms of classification accuracy. DSOM showed the lowest accuracy for the SP-HAR dataset.

3.3.3.9 Comparing Visual Data Mining Capabilities SOM and DSOM

Data mining methodologies have become almost indispensable with the increase of amount and complexity in data in almost every domain. Data mining is an interactive process which requires intuition and human knowledge coupled with modern machine learning techniques. Visual data mining (VDM) is the process of exploration, interaction, and reasoning with abstract data in human perceivable way [2]. Thus, it allows humans to incorporate human intelligence in the data mining process, and it has been shown that human involvement increase the effectiveness of data mining processes. Visual data mining facilitates the involvement of domain experts in the data mining processes.

The effectiveness of visual data mining is especially dominant when paired with unsupervised methods due to the abundance of unlabeled data. Therefore, in this work, we analyzed the effectiveness of using novel DSOMs for visual data mining. DSOM's visual data mining capability was evaluated using the following visual data explorations methodologies: 1) U-Matrix, 2) hit maps and 3) data histograms.

- U-matrix: (Unified Distance Matrix) is one of the most widely used methods

for visualizing the cluster structure of SOMs [28], [29]. It shows the distance between weight vectors of neighboring neurons (immediate neighbors) using color codes [30]. If distances between neighboring units are small, then they represents a cluster pattern with similar characteristics. If neighboring units are far apart, then these units are located on low dense input space with few patterns. They can be considered as separation between clusters.

- HitMaps: This shows how often a neuron is chosen as the BMU. Hit map information can be utilized in clustering the SOM by using zero-hit units to indicate cluster boarders [28] .

- Data Histograms: These represent how many data items are represented by a specific unit. This is also a slightly different representation of hitmap representation.

The MNIST data set is used for the comparing the VDM capabilities of SOM and DSOM. The ratio of training to test data set was used as 3:10. For this experiment, a significantly smaller training set of 3000 images were used to reduce the classifier training time. The complete test set of 10000 images were used to test the accuracy of algorithms. Building an efficient classifier using a small training data will be advantageous in cases where there is only limited amount of training data to training a supervised classifier and to for implementing classifiers which are time and cost efficient. The training data set was selected randomly while maintaining the balance class labels. Since the classifications of all the DSOM models were performed on a 2D neuron map of size 8X8 (last SOM layer), the SOM model with 8X8 was selected for the comparison.

Hit map representation which shows how often a unit is chosen as a BMU. Figure 9 (a) and (b) represents the hit map observed for SOM and DSOM, respectively. Hit

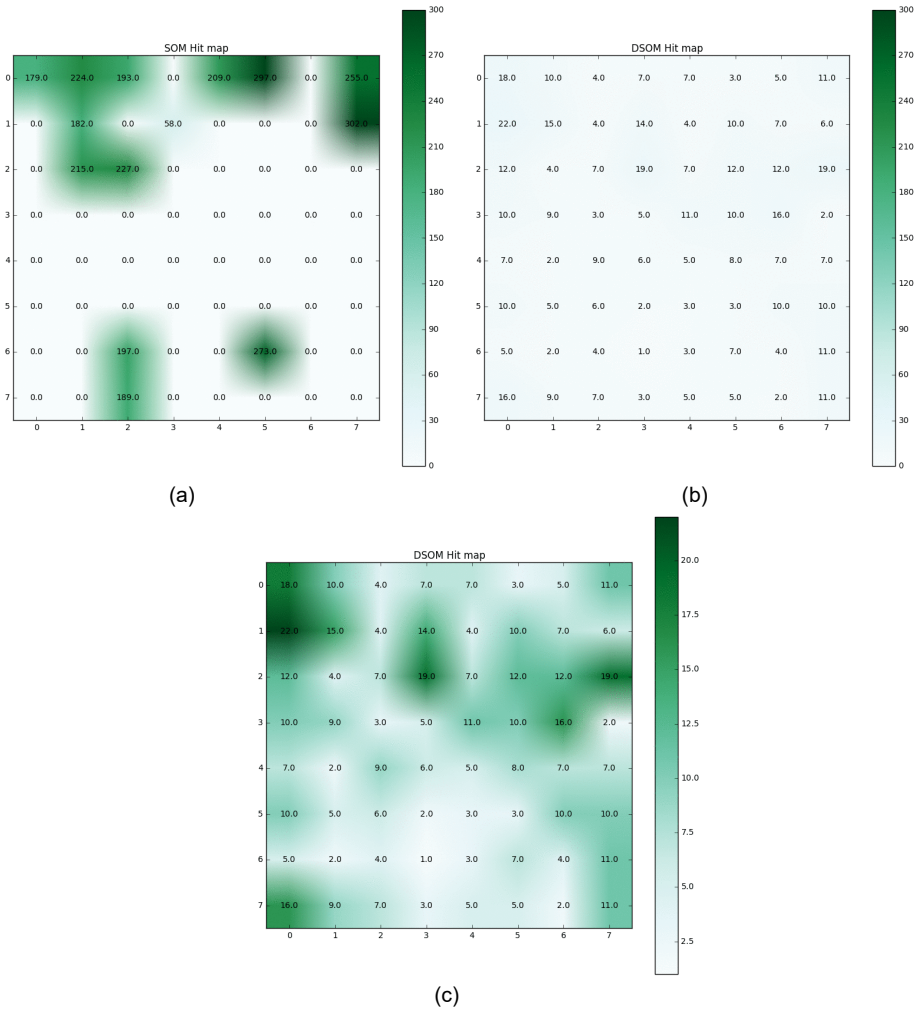


Fig. 9. Hitmap representations for (a) SOM, (b) DSOM-scaled, and (c) DSOM-unscaled)

maps SOM and DSOM were mapped to the same scale for comparison purposes. It was observed that only a few units of the SOM were activated and most units showed 0 hit value, whereas in DSOM, all units showed a hit value greater than 0 (all units were active). When comparing the neuron hits, active SOM units showed very high neuron hits compared to DSOM. In Fig 9 (b), it appears as if the DSOM neurons do not show a difference in operation. However, In Figure 9(c), which represents an unscaled hit map of DSOM, it can be seen that some units show higher activity

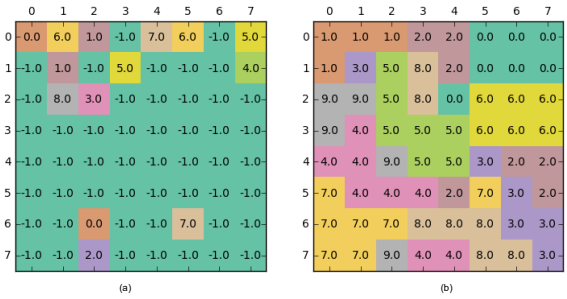


Fig. 10. Hitmap representations for (a) SOM, (b) DSOM-scaled, and (c) DSOM-unscaled)

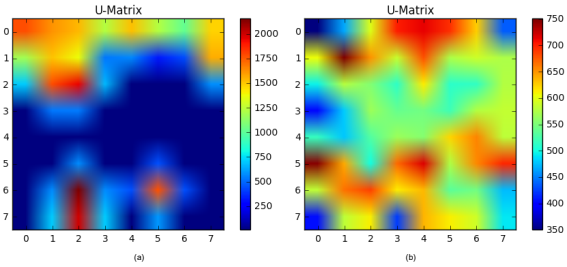


Fig. 11. U-Matrix representations for (a) SOM, (b) DSOM)

compared to the other units.

Figure 10 represents the special hit map which we used for implementation of the unsupervised classifier. Using that we generated a new hit map representation where each unit represents the class label which it activated as BMU at the highest frequency (See Figure 10). It was observed that (8*8) SOM model doesn't show proper clusters whereas DSOM hit map shows better clusters, where neighboring units act as one cluster to represent one class. The '-1' value is assigned for neurons with 0 BMU hits.

Figure 11 (a) represents the U-matrix obtained for SOM architecture whereas Figure 11 (b) represents the U-matrix observed for DSOM architecture. It was observed that SOM U-matrix doesn't show any large clusters or cluster separations for both SOM and DSOM but for DSOM it showed some cluster separations. Large blue color area in the SOM U-matrix corresponds to the area with inactive units. Further

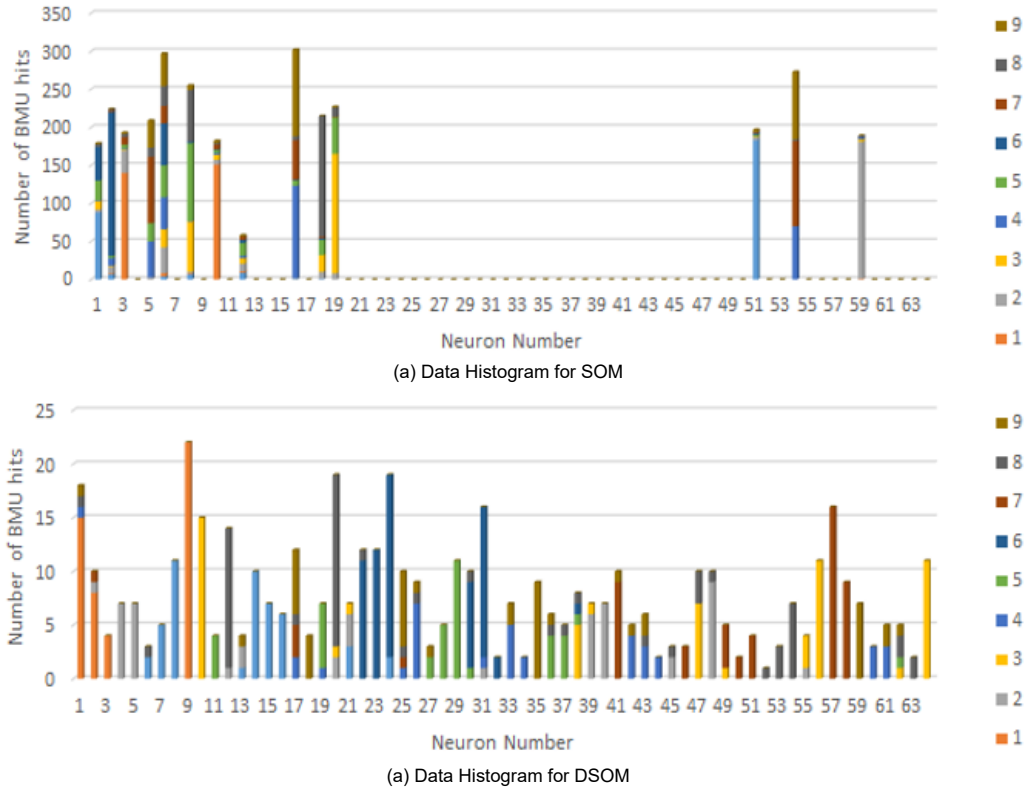


Fig. 12. Data Histogram representations for (a) SOM, (b) DSOM)

analysis of the U-Matrix is needed to improve the visualization on this front. There are several methodologies proposed in literature on ways of calculating the U-Matrix. These methodologies will have to be explored. Furthermore, as alternatives of the u-matrix, other weight vector visualization techniques such as t-distributed Stochastic Neighbor Embedding (t-SNE) can be evaluated [33].

Figures 12 (a) and (b) represent the data histograms obtained for SOM and DSOM architectures respectively. Data histograms visualize which units are activated and how often it became BMU compared to other units in the map. It also represents the amount of each unit acted as the BMU for each class label. According to the data histogram of SOM, it was observed that all the activated units were activated for more than one class label. There were very few units which were activated for only

a single class (See neuron number 51, for class label 0). Further, number of hits per unit was significantly higher for SOM units compared to DSOM units. According to the data histogram for DSOM, it can be seen that all the units have been activated to some degree. However, it can be seen that most of the neuron has been activated only for a specific class label. In the ones that has many class labels, one class label has dominated the others in terms of frequency. The better cluster separations observed in hit map and data histograms, make it easier for the domain expert to label the neuron and group the neurons based on classes. Hence, hit map and data histograms improve the visual data mining process.

In comparison with traditional single layered SOM architectures, experimental results showed that DSOMs produced more accurate visual representations of the underlying data distributions. Therefore, DSOM is a viable method for generating easily understandable visual representations of high-dimensional complex datasets. These visual representations can be powerful tools in the real world, leading to better understanding of systems and thus enabling the design of better algorithms for control and monitoring.

3.3.4 Contribution 1 (a): Findings, Discussion, and Future Work

This section presented a deep self-organizing map architecture (E-DSOM) for unsupervised image classification. The E-DSOM extended the originally proposed Deep Self-Organizing Maps (DSOM) in two ways: 1) the learning algorithm was modified to be completely unsupervised, 2) the architecture was modified to learn features of different resolutions in parallel. The modifications were made to improve the following: 1) classification accuracy, 2) generalization capability, and 3) training time. E-DSOM was tested on three datasets and compared with DSOM. E-DSOM outperformed DSOM in terms of classification accuracy with improvements of up to

15%. Generalization capability was tested by adding noise to test data. E-DSOM outperformed DSOM at all noise levels (barring one instance with comparable results), evidencing better generalization capability. Computational time was improved by gaining the same or better classification accuracies with a shallower model. E-DSOM showed training time improvements up to 19%. Therefore, empirical evidence supports our hypothesis.

Further, E-DSOM architecture was compared to other unsupervised algorithms. E-DSOM showed comparable performance to the AE and the CAE while outperforming them on two datasets. Therefore, empirical results show that E-DSOM algorithms are competitive and a viable option for unsupervised learning.

In terms of visualizations for VDM, experimental results showed that DSOM based hit maps and data histograms provided a better representation of the underlying data distribution than the SOM. Due to its high-level feature abstraction capabilities, DSOM is able to produce visualizations Figure 7: Data Histogram for SOM Figure 8: Data Histogram for DSOM, which accurately reflect the input data distributions. This enables a user to examine these visualizations and extract patterns, relationships, and behavior in data and glean a better understanding of systems. This understanding can lead to better predictive systems, monitoring systems, and control schemes. Therefore, based on experimental results, it can be concluded that DSOM is a viable method for visual data mining.

In future work, the following avenues will be explored: 1) capability of creating low dimensional embedding of high dimensional datasets using E-DSOM; 2) exploring the capability of using E-DSOM for real-world applications.

3.4 Interpretable Clustering using Self Organizing Map algorithm

Despite the tremendous benefits of machine learning (AI), many people hesitate to trust AI-based systems due to their black-box nature, which makes it difficult to get insight into the internal decision-making process of AI models [50]. Especially for human-in-the-loop systems, humans need to understand these algorithms such that they can trust these models. By addressing this question, the explainable machine learning (XAI) research area has been received a lot of attention. The goal of XAI is to provide reasoning for ML model outputs, allowing humans to understand and trust ML models' decision-making process. Currently, many entities have put their attention to XAI. DARPA is one of the first organizations that initiated XAI programs focusing on developing explainable models [5]. Their program is interested in developing a toolkit library consisting of machine learning and human-computer interface software modules that could be used to develop future explainable AI systems. Currently, many entities have put their attention to XAI, such as European Commission, NSF, NIST, and IBM [104, 105, 106].

While XAI has become a trendy research topic, the majority of the work has been focused on supervised machine learning methods. However, real-world settings such as CPSs bring the challenge of dealing with high volumes of unlabeled data at a rapid pace. The manual labeling process is expensive, time-consuming, and requires the expertise of the data [3]. It has been found that the 25% of time allocated to machine learning projects is for data labeling. Further, supervised feature learning is not only unable to take advantage of the abundance of real-world unlabelled data, but it also can result in biases by relying on labeled data. These limitation has gained the focus towards unsupervised ML algorithms and is predicted to be far more important in the long term [107]. Given the abundance of real-world unlabelled data, it is important to

focus on developing explainable unsupervised ML methods. However, in the current literature, very little work has been performed focusing on explainable unsupervised ML. Therefore, this work focuses on unsupervised explainable ML.

In this work, we explored Explainable Unsupervised Machine Learning on different aspects. Further, we propose a novel Explainable Unsupervised Machine Learning (XUnML) approach using the Self Organizing Map (SOM) algorithm.

3.4.1 Background

As we discussed in the Introduction section, existing work on XAI is mainly concentrated on supervised learning algorithms. For domain areas such as CPSs, UnML is essential for assisting human decisions in building effective ML models. These systems generate a massive amount of unlabelled data at a rapid speed. Therefore, relying on SML alone is not sufficient for data-driven decision-making for CPSs. Further, unsupervised learning has a wide range of application areas, including model pre-training, auto-regressive modeling, and generative modeling.

This section explores what XAI would look like in an unsupervised context, the need for unsupervised XAI methods, current literature on unsupervised XAI, and how unsupervised XAI can be used within the domain of CPSs. Further, this section discuss on visual data mining capabilities of SOM, which are used towards developing the novel interpretable SOM technique.

3.4.1.1 Desiderata of Explainable Unsupervised Machine Learning

As we discussed in the previous section, UnML offers a solution to analyze the large amount of real-world unlabelled data generated at a rapid speed. However, most of the existing UnML methods do not provide a way for people to understand their underlying decision-making process. Especially for non-domain experts, these models

act as black-boxes. This black-box behavior leads to many drawbacks, including limiting the user's involvement with model improvement, limiting user input integration for model debugging, and harming the user trust in these models, making humans not deploy them in real-world environments [108, 109, 110, 111, 112]. Interpretable models are essential for high-risk environments where the model outcomes can result in severe consequences. For example, one use case is anomaly detection systems in critical infrastructures. On these systems, it is not enough to get the predictions (anomaly or not) of UnML models. It is crucial to produce an explanation of why it is an anomaly. This information is essential to identify where the anomaly occurred, possible catastrophic effects and make decisions to recover the system. Therefore, it is essential to analyze and explain the result obtained through these UnML models [108, 109, 110, 111, 112].

Analyzing and interpreting the results obtained through UnML is a very challenging process. This process often requires expert-based sophisticated manual inspection, which takes a significant amount of time [108][110]. Further, complexities, high-dimensionality, and real-world data volume make it impossible to use manual expert-based data analysis. Existing unsupervised quality metrics such as Silhouette or Rank Index do not provide any explanations on why data record belongs to a specific cluster [108]. They only provide a structural insight that is not perceivable to non-domain experts. Further, many available methods are hard to explain, partially because they depend on all the data features in a complicated way, making it difficult to explain in a perceivable manner [111]. Other supervised quality metrics such as cluster purity requires labeled data, requiring expensive manual labeling. This approach is expensive and can result in partial, incorrect, or biased results [108]. Therefore, there is a crucial need to develop explainable UnML methods or develop methods to explain existing UnML methods.

3.4.1.2 Current literature on Explainable Unsupervised Machine Learning

Principle Component Analysis (PCA) has been used for interpreting the clusters by visualizing them across two or three dimensions [110]. However, it limits the number of dimensions that can be used for explaining the clusters, as visualizing is not possible when the number of dimensions increases. In [108], researchers have used existing supervised XAI methods for interpreting UnML approaches (EXPLAIN-IT). First, they cluster the input data using existing clustering methods such as K-Means or DBSCAN. A classifier is then trained on input data using the generated cluster labels as class labels for the classifier. Finally, the classifier is explained using existing model agnostic methods such as LIME. However, these can result in model biases, and the current research on this is at a primitive stage.

Interpretable tree-based clustering models have gained much attention recently as the decision tree model itself is an explainable model [113, 114]. In [113, 114], an explainable decision tree method was introduced by generating the smallest binary tree possible (threshold tree) with k leaves. Each node in the tree iteratively divides the input data into k clusters. By restricting to k leaves, they ensure that each such path accesses at most $k-1$ features. The explanations were generated using $k-1$ features. Also, in [109], researchers have proposed an explainable decision tree model (eUD3.5) where they have use compactness and separation of data clusters when evaluating feature splitting in the tree.

Deep Neural Networks (DNNs) have shown state-of-the-art performance in many areas such as computer vision and natural language processing. However, many DNNs are used as black-boxes. There are a couple of initial attempts toward explaining unsupervised DNNs such as Autoencoders. In [115]s, interpretable Variational AE

has been presented. This is performed by analyzing the gradient contributed by each feature of a data record. Another interpretable VAE is presented in [116], and [117] by changing the decoder to embody explicit expert knowledge. Therefore, these architectures result in a latent space that has semantic meaning. Fuzzy logic combined with ML has also been used for achieving interpretability There are some initial attempts towards developing interpretable systems combining fuzzy logic systems with DNNs and clustering algorithms. However, majority of these system has some degree of the supervised learning process within their pipeline.

3.4.1.3 Visual Data Mining Capabilities of SOM

The following items outline exploratory data analysis capabilities of SOM. Later of this chapter present how to use these capabilities to make explainable SOM.

- Histograms: These neuron histograms shows the data distribution of the 2D data topology of SOMs. It can be used as a visual indication for identifying whether the network can cluster the input data correctly. A properly trained network topically shows data grouped in some regions of the map (high data distribution density), making clear boundaries between clusters.

- T-distributed Stochastic Neighbor Embedding (t-SNE): This is a dimensionality reduction technique that is widely used for visualizing high-dimensional data. For SOMs, this can be used to represent the input data points and neuron weight together. It indicates to users that the network weights can represent the distribution of input data. Therefore, it is a clear indication to visually explore whether the trained network represents the trained data.

- Heat Maps: These are intensity representations of SOM network properties. Several types of heat-maps can be generated using cluster labeled or data labeled

if available.

  - Class hits: This is a different visualization of Neuron Hit Histogram. This will represent the number of classes where each neuron fired for the whole SOM network topology. If majority of data fired a neuron belong to one class label, then it can be use as an indication for a well trained SOM network.

  - Data hits: This represents the number of data points where each neuron fired for. If many neurons do not fire for any data point, then network size can be reduced. Therefore, this can be used to decide the SOM network size.

  - Class Percentages: This will represent the percentage purity of each neuron. Percentage purity can be use to ignore neurons with low purity allowing users to increase the quality of the network by retraining, redesigning, expanding the network. In case of tie, neighboring neurons are used to decide the class/cluster label.

- U-Matrix: Unified Distance Matrix (U-Matrix) is the standard visualization for SOMs representing the information regarding the distances between neighboring neurons. These maps are used to identify the naturally existing clusters and to identify well-separated clusters from overlapping clusters.

- Component Planes: This visualization shows the value of a single feature in each SOM neuron. A single component plane represents how a specific feature value changes across clusters. Further, by comparing multiple plans, it is possible to identify correlated features.

- U-Map: Similar to t-SNE, this also use to visualize high dimensional data. This

builds a high dimensional graph representation of the input data then optimizes a low-dimensional graph to be as structurally similar as possible. For SOMs, this can be used to represent the input data points and neuron weight together. It indicates to users that the network weights can represent the distribution of input data.

As described above, SOM has many visual data mining capabilities which allows domain experts and non-domain experts to interact with SOM, making SOMs good candidates for exploring application in CPSs. Further, there have many improvements have been proposed that can be done on SOMs to improve it's capabilities. However, to the best of out knowledge, there is no efforts done towards making the model interpretable. Therefore, in the next section, we propose an approach toward developing an explainable SOM algorithms.

3.4.2 Novel Explainable Technique for SOM

As we discussed above, SOM algorithm has many visual data mining capabilities. SOM is a unsupervised clustering method which is trained to produce a low dimensional representation of a large training dataset. U-Matrix of SOM neuron weights can represent any natural clusters available within training data. Component planes of SOM neuron weights can be used to visualize how the feature values change across clusters (feature summary visualization). In this work, we used SOMs training approach (winner-take-all algorithm) together with the above discussed visual data mining capabilities of SOM to make the algorithm explainable. We propose a model-specific, post-hoc interpretable method for SOMs. The result of this method consists of feature summary statistics, model internals, and feature summary visualizations. The proposed approach is able to provide both global and local explanations. Further, we will discuss how each of these generated explanations can be used for CPS opera-

Table 8. Proposed approach for Explainable SOM

Algorithm II: SOM Interpretation	
Inputs: Trained SOM, Testing dataset (X), standard deviation threshold (th)	
Outputs: Local interpretation, Global interpretation	
1:	% Calculating list of important features for each neuron in SOM
2:	**for** each neuron i do
3:	$X'_i \leftarrow$ initialize an empty array
4:	**for** each data record x in X do
5:	$bmu \leftarrow$ calculate BMU using trained SOM
6:	$if(bmu == i) : X'_i.append(x)$
7:	**end for**
8:	**for** each feature j in n dimensional feature space
	% Calculating standard deviation for each feature
9:	$\sigma f_{i,j} = \sqrt{\frac{\sum_{j=0}^{n}(X'_{i,j} - X'_{i,j})}{n-1}}$
10:	$if(f_{i,j} > th) : f_{i,j} = INF$
11:	**end for**
12:	% Calculating the order of important features for each neuron in SOM
13:	$f'_i \leftarrow$ sort indices j of σf_i in acceding order if $f_{i,j}! = INF$
14:	**end for**
15:	$KMeans \leftarrow$ Apply K-Means clustering to SOM neurons and find optimal K
16:	$clus \leftarrow$ initialize an empty array for storing cluster labels
17:	**for** each neuron i do
18:	$clus_i \leftarrow$ apply $KMeans$ to ith neuron and find its cluster label
19:	**end for**

tions. Here we will discuss the steps for identifying most important feature list using SOM algorithm, model fidelity evaluation method, and generating interpretations.

1. Training of the SOM with dim dXd:

   Trained the SOM with the winner-take-all algorithm discussed in Algorithm I (Table 1).

2. Calculating the order of important features for each neuron (Table 8, Algorithm II, line 1-13):

   After training SOM with the training dataset, the trained SOM acts as a set of data points which represent the entire training dataset. Therefore, each neuron

Table 9. Proposed approach for Explainable SOM

Algorithm III: Experiment I	
Inputs: Trained SOM, $X, f'_i, clus$	
Outputs: Swap Percentages	
1:	$X'_i \leftarrow$ initialize an empty array
2:	$count \leftarrow 0$ %initialize a variable
3:	**for** p number of features out of n where $(p * 100/n)\% <= 50\%$
4:	**for** each data record x in X do
5:	$bmu \leftarrow$ calculate BMU for x using trained SOM
6:	$x' \leftarrow$ change first p features of x from $f'_b mu$
7:	$bmu' \leftarrow$ calculate BMU for x' using trained SOM
8:	$if(clus_{bmu} != clus_{bmu'}) : count++$
9:	**end for**
10:	$tot \leftarrow$ number of data records in X
12:	Swap Percentage $= count * 100/tot$
13:	**end for**

is a generalized representation of a set of training data records. We use the training data set and extract a set of data points (X'_i) that selected i^th neuron as their BMU to calculate the ordered list of important features for i^th neuron. The importance of a feature is decided by calculating the standard deviation. We calculate the standard deviation for each feature j in x'_i.

$$\sigma f_j = \sqrt{\frac{\sum_{j=0}^{n}(X'_{i,j} - \bar{X'_{i,j}})}{n-1}} \tag{3.8}$$

3. Calculating the ordered list of important features for neuron i (Table 8 Algorithm II, line 1-13):

   Then the indices of features are ordered from lowest standard deviation to the highest standard deviation, representing the ordered list of feature from highest importance to lowest feature importance. Each neuron in SOM represents a set

Table 10. Proposed approach for Explainable SOM

Algorithm IV: Experiment II

Inputs: Number of features (t), Trained SOM, X, f'_i

Outputs: Feature Percentages

1: $list1 \leftarrow$ initialize an empty array
2: **for** each data record x in X do
3: $bmu \leftarrow$ calculate BMU for x using trained SOM
4: $l_d = |x - bmu|$ %Calculate L1 distance between x and bmu
5: $closets\_features \leftarrow$ find closest t feature indices
6: $tot1 \leftarrow$ cardinality of $closets\_features$
7: $tot2 \leftarrow$ cardinality of f'_{bmu}
8: %percentage of t feature are in f'_{bmu}
9: $list1.append(tot1 * 100/tot2)$
10: **end for**
11: $tot \leftarrow$ number of data records in X
12: Percentage = $sum(list1)/tot$
13: **end for**

of data points, and low standard deviation of a feature represents low variation of a feature values, indicating that many data points have that feature value within a small range. The domain expert/user can decide on a threshold (th) standard deviation value so that any feature with equal or less standard deviation value is considered as the most important (active) feature. These ordered important features were used to achieve interpretability.

4. Cluster SOM neurons (Table 8, Algorithm II, line 14):

   To achieve interpretability, we clustered the trained SOM neurons. In this experiment, we used K-Means clustering. The number of clusters (K) was decided based on two cluster quality metrics: Silhouette Coefficient and Davies-Bouldin Index. Further, cluster quality metrics together with U-matrix and other visualization capabilities of SOM allows the domain expert to visually

analyze the natural clusters of training data and evaluate the quality of K clusters to decide whether the results are reasonable.

5. Model fidelity evaluation:

   We designed two experimentation to evaluate whether the identified features for each neuron are actually important for the decision-making process of SOM (fidelity test). This is performed by performing two experiments on the test data-set.

   - Changing the feature values of identified important features and checking whether the cluster labels (model outcomes) changes (Table 9, Algorithm III, Experiment I). Through this experiment, we calculate the percentage of data points where the cluster label an be changes by changing their feature values of important features.
   - Calculating the percentage of important features which are included in identified important features (Table 10, Algorithm IV, Experiment II) Through his experiment, we check whether the calculated order of importance feature lists are valid for the test data-set.

   These are discussed in detail in the next section.

6. Results interpretation: Once we identified the most important features and evaluated the features, we generated local and global explanations using SOM.

   - Local interpretability: Once an ordered set of important features are calculated for each neuron; it is used to generate local interpretation for a specific input record by providing the user a subset from important features and its value range (very low, low, medium, high, very high). It has

to be noticed that the feature value granularity can be defined by users based on their preferences.

- Global Interpretability: For each feature, we can visualize how the feature value is different across clusters and what features are active within clusters. Ordered important feature summary (features and values(range) of important features) for a set of neurons belonging to a particular cluster is used as a global interpretation.

3.4.3 Experiments and Discussion

In this section we discuss the five data sets we used for this experiment, the design of the evaluation methods, results, and discussion on results. First we will discuss the data sets used for this experiment.

KDD: This is a commonly used benchmark dataset for network intrusion detection and anomaly detection. It has around 2 million records divided into train and test sets. It consists of 41 features, and all the records are labeled into two classes, attack or normal. The attack data represent four categories, namely, Denial of Service (DOS), User to Root Attack (U2R), Remote to Local (RTL), and Probing Attack. For this experiment, we used normal records and DOS records (attack). This choice was made as other types of attacks have subcategories that do not include the test set. SOM algorithms, in general, does not handle huge variation in new data. This dataset has both categorical and numerical feature values.

German Credit: This dataset has 1000 data records with 20 features. It has both categorical and numerical feature values. Each record represents a person who takes a credit from a bank. Each person is labeled as good or bad credit risks.

Bank marketing: This dataset is derived from a marketing phone call campaign of a Portuguese banking institute. This data set has 45211 records, each with 12

features. Features are only numerical values. This has two classes, yes and no. The classification goal is to predict if the client will subscribe to a term deposit or not.

Adult Income: This dataset has 48842 records, each with 14 features. Features have both categorical and numerical features. The data set is labeled into two classes, representing whether the salary exceeds 50k or not based on the features. This dataset has missing values. In this experiment, we removed records with missing values.

DoHBrw-2020: Canadian Institute for Cybersecurity provides a set of datasets for building intrusion detection systems. For this experiment, we used the CIRA-CIC-DoHBrw-2020 dataset, which consists of benign and malicious records for DoH (DNS over HTTPS protocol) traffic along with non-DoH traffic. It consists of roughly around 250k records with 28 features. It has both categorical and numerical feature values. Benign and malicious records were considered as two classes.

Since these datasets have categorical variables, we used the frequency encoding method to convert categorical variables to nominal variables. Min-max scalar was used to scale the data into the 0-1 range. When there are separate train and test sets, they were used as it is for training and testing purposes. If the original data set is not divided into train and test, 70% of the data was randomly selected for training, and the rest was used for testing.

To decide the optimal number of clusters and dimension of the SOM, we used u-matrix together with two widely used clustering performance metrics, namely Silhouette Coefficient and Davies-Bouldin Index. They do not require labels to evaluate the clusters. They use different methods to calculate the compactness (density) of a cluster and separation (distance) between clusters.

- Silhouette Coefficient: This is calculated using the mean intra-cluster distance and the mean nearest-cluster distance for each sample. The score is higher when

clusters are dense and well separated, which relates to a standard concept of a cluster. The score is bounded between -1 for incorrect clustering and +1 for highly dense clustering. Scores around zero indicate overlapping clusters.

- Davies-Bouldin Index: This index signifies the average 'similarity' between clusters, where the similarity is a measure that compares the distance between clusters with the size of the clusters themselves. A lower Davies-Bouldin index relates to a model with better separation between the clusters. Lower the better, Values closer to zero indicate a better partition.

Figure 13 shows the change in cluster quality matrices used for this experiment for the Bank Marketing data set. It shows how these matrices change with respect to SOM dimensions and the number of clusters. For a given SOM size, we calculate cluster quality metrics for SOM neuron weights (Blue) as well as for the training dataset (Orange). This analysis is used to identify the optimal SOM dimension and number of clusters. If the trained SOM neurons are a good representation of the whole data set, then cluster analysis of SOM weights and the whole data set should follow similar trends. Figure 13 shows that they follow the same trends when increasing the number of clusters. Based on the Silhouette Coefficient and Davies-Boulding index value, 3 to 5 clusters seem to be the best option for the tested SOM dimensions (8,16,2,40).

3.4.3.1 Model Fidelity

Once the ordered list of important features is calculated for each neuron in trained SOM, it is necessary to evaluate whether the identified ordered features are actually important using the a data perturbation experiments discussed in the previous section (Algorithm III). First, for each data point x in the test set, we check its BMU index

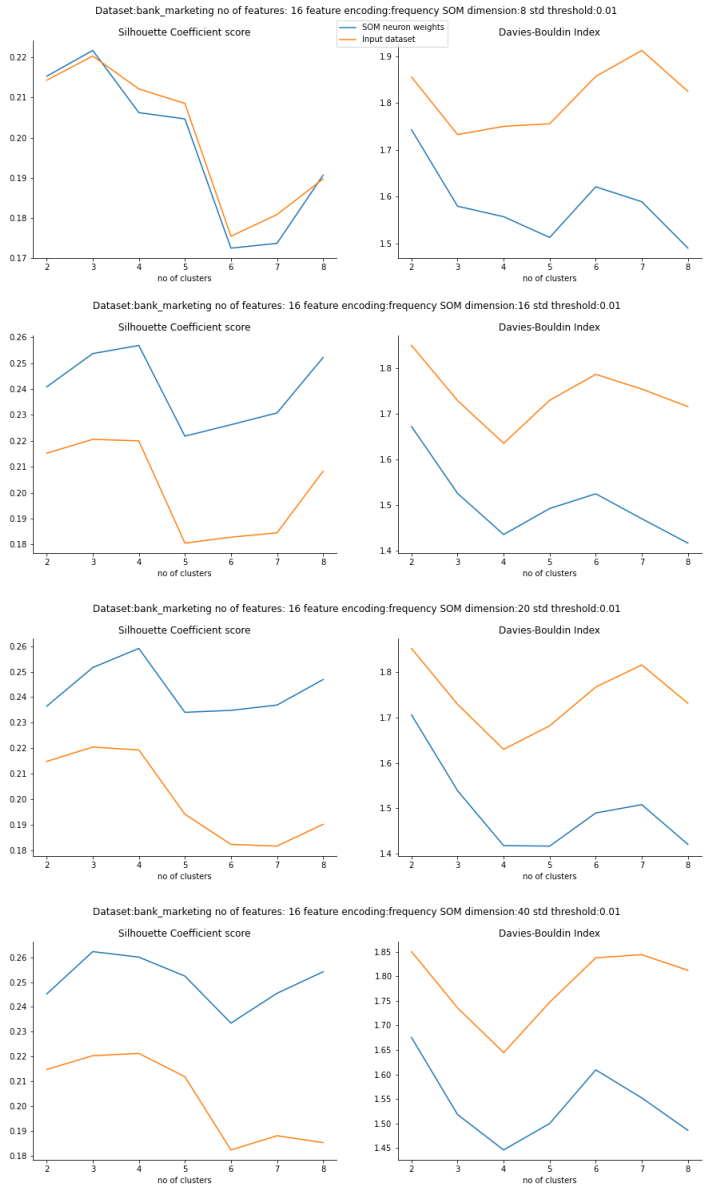


Fig. 13. Cluster quality evaluation approach for K clusters using Silhouette Coefficient and Davies-Bouldin Index for different SOM map sizes

i and the cluster label (m) of the BMU. Based on BMU index i, we have a list of most important features f_i'. It has to be noticed that different BMUs have a different number of important features based on a user-specified threshold (th) on standard deviation. To evaluate whether the identified ordered features are actually important, we change the feature values of important features of the test data record. First, we calculated the average feature values for each cluster using the neurons belong to that cluster. Then, the feature values of important features of x were replaced by the mean feature values of a cluster k where $k! = m$. Then we check whether the BMU of x is changed to another BMU, which does not belong to the original cluster label of that data point (m). Our hypothesize is that when we change the values of the most important features, the cluster label of the data point should change. We did this for the whole test data set and calculated the percentage of data records where we can change the original cluster label by changing the feature values of important features (Swap percentage). If it does, it confirms our hypothesis that the identified features decided the cluster label of that data point.

It is also essential to identify the minimum number of important features that define the cluster label of a data point. Explanations should be generated using a small number of features so that it is easy to perceive by the user rather than explaining with a higher number of features. Therefore, the cardinality of f_i' should be limited to a user-defined value. In this experiment, we tested with different cardinalities; 10%, 20%, 30%, 40%, and 50% of important features out-of the total number of features of the dataset, which is also bounded by the threshold (th) of standard deviation (total number of identified important features for neuron i). To evaluate our hypothesis, we changed the feature values of randomly selected features and unimportant features (considering highest standard deviation to lowest). I.e., given the cardinality p%, we changed the values of p% number of most important features (features with lowest

std values), p% number of randomly picked features, and p% number of most minor important features. For each data record in the test set, we checked whether it changes its cluster label when we change the feature value under the three scenarios described above. For each scenario, we checked the two cases; 1) What is the percentage of test data records where the cluster label can be swapped by at least one other cluster label, 2) What is the percentage of test data records where all other clusters can swap the cluster label. The reason for this is, for some data points, feature value perturbation using a close-by cluster features may be not strong enough to push it out of the original cluster. Therefore, we checked whether the the cluster label of a given data point can be change by using the feature values of at-least one other cluster. For example, assume we have 4 clusters and a data point j, which belong to cluster 2. We replace its feature values with average feature values of clusters 1, 2, and 4 and check whether we can change its cluster label from 2 to some other cluster label. It has to be noted that lower cardinality $n\%$ and higher swapped percentages are expected. The result of swap percentage calculation for all the data sets is present in Figure 14. It can be seen that the best results for swapped percentages (tallest bar) were shown by important features (blue), and the second-best was shown by random features (brown bar) for all the data sets except for the second scenario (What is the percentage of test data records where all other clusters can swap the cluster label) of KDD dataset (second column, last raw). The reason for this can be the highly imbalanced classes of KDD data-set and higher differences between train data and test dat, resulting poor performance. However, KDD also performce as expected for first scenario (What is the percentage of test data records where the cluster label can be swapped by at least one other cluster label). This empirical results confirms our hypothesis that the identified important features using the proposed approach for SOM decided the cluster labels of data records.

Another experiment was performed to check the percentage of selected K number of features included in the most important feature list of a BMU (Algorithm IV). For each data record in the test set, we calculated the feature-wise l1 distance between the data record and its BMU. Then the features were arranged based on the ascending order of l1 distances. Our hypothesis was that the closest features are the most important features of that data point, and they will be included in the identified important feature lists of its BMU. Once features distances are arranged in ascending order, K features are selected on three different strategies; 1) Closest, 2) Random, and 3) Furthest. We then calculated the percentage of K features are included in the important feature list of the BMU. The results are presented in Figure 15 where the X-axis represents the K number of features, and Y-axis represents the percentage of K features that were included in the important feature list. Blue color represents the closes feature, yellow represents the random features, and the green represents the furthest features. It can be seen that the blue bar shows the highest percentage for all K features, whereas yellow shows the second higher percentage. It infers that the closes features are included in the identified important feature lists of each BMU.

As described above, we identified the most important ordered set of features for each neuron in the SOM network and evaluated the model fidelity using two experimentations. Then we used the identified ordered list of important features to generate explanations.

3.4.3.2 Local Interpretability

For a given data point, a local explanation is generated based on the important features of its BMU, which were identified using Algorithm III. It has to be noticed that two different data points with the same BMU can have different orders of important features based on the l1 feature distance to the BMU. We provide a set of most

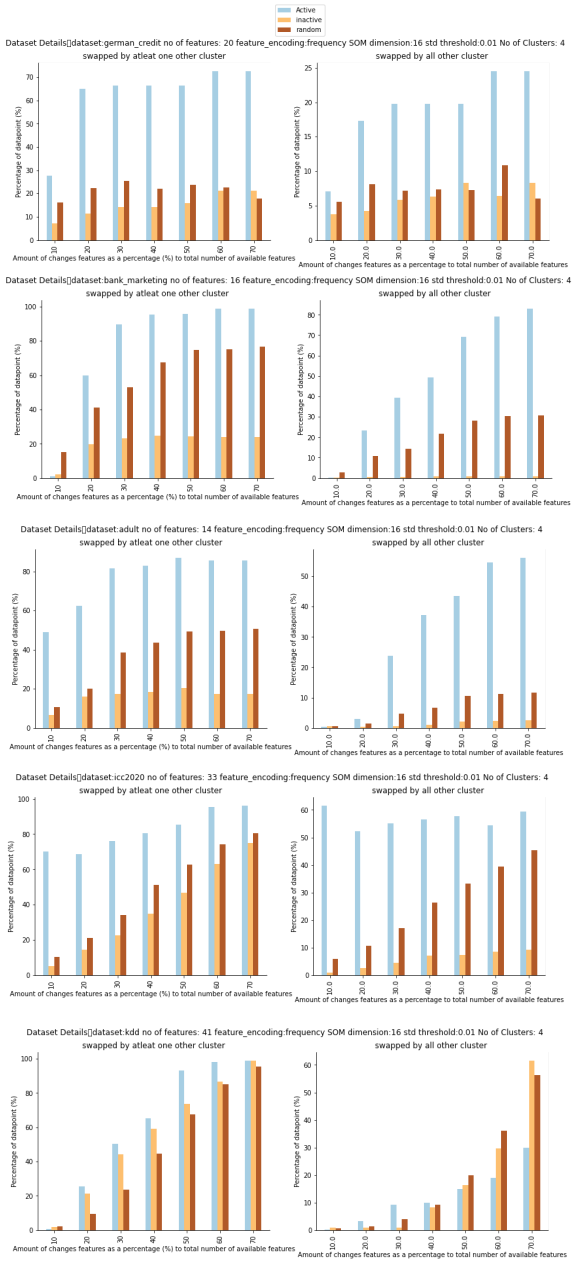


Fig. 14. Fidelity test, Experiment I: Changed the values of p% number of most important (active) features, p% number of randomly picked features, and p% number of least important (inactive) features and calculated the percentage of data points where the cluster label changes after changing %p feature out of all the feature. we checked the two cases; 1) What is the percentage of test data records where the cluster label can be swapped by at least one other cluster label (left), 2) What is the percentage of test data records where all other clusters can swap the cluster label (right).

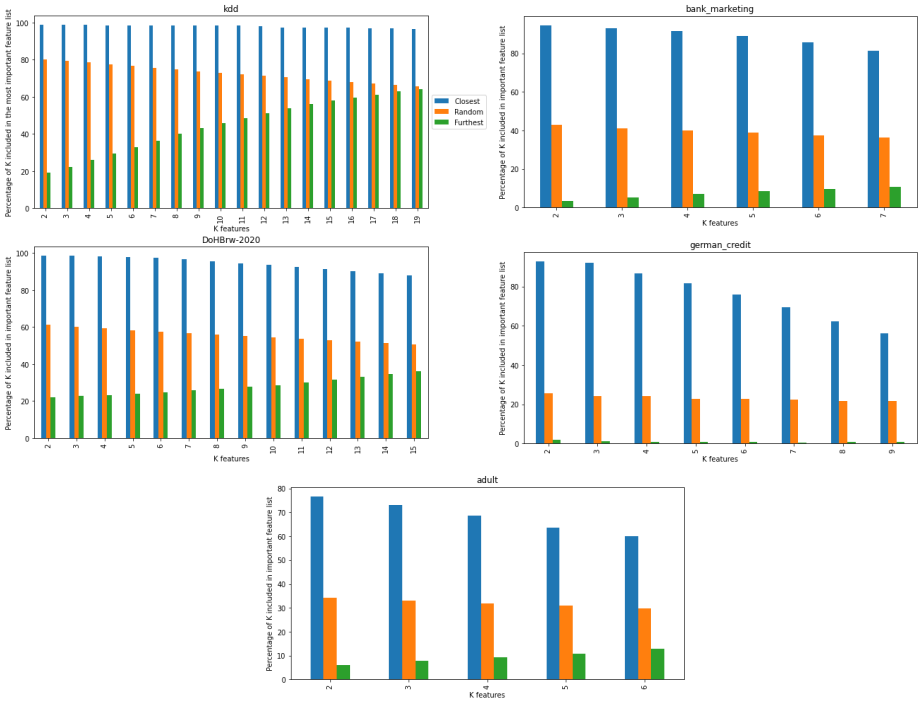


Fig. 15. The percentage of closest K number of features included in the most important feature list of the BMU

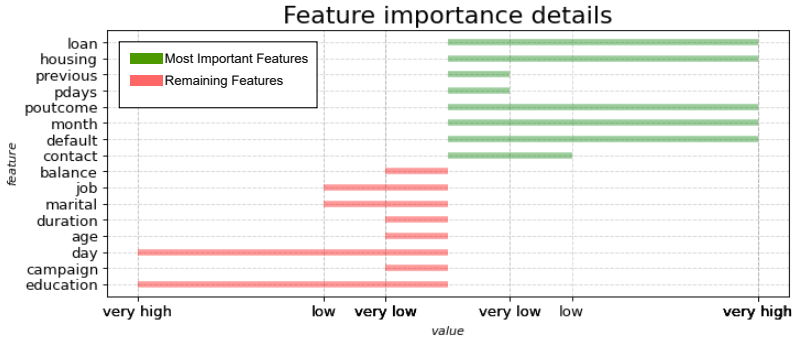


Fig. 16. Local Interpretability; Explanation for a single data record, features are ordered from ascending order based on feature wise distance to BMU

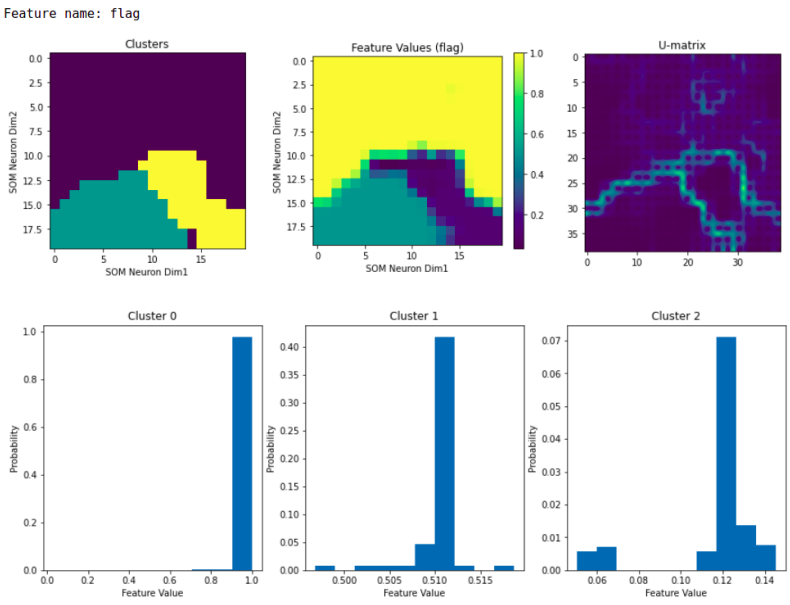


Fig. 17. Global Interpretability; Feature behavior for 'flag' feature of KDD data set across clusters (SOM neurons were clustered into three categories, U-matrix visualize the distances between clusters and how well clusters are separated, the 'flag' feature value is different across clusters).

important features and their feature values which are ordered based on the l1 feature distance calculated between the data point and its BMU. All the feature values are presented in several levels (very low, low, medium, high, very high). Low l1 distance indicates more important features specific to that data point. It has t be noticed that two data points can have the same set of important features, but the order of importance can be different.

Figure 16 shows the local explanation for a single data point of bank loan data set, generated using the proposed approach. The features are ordered based on l1 distance in ascending order (bottom to top) to its BMU. Thus, 'Education' is the furthest feature indicating the lowest importance, whereas the 'loan' is the closest feature indicating the highest feature importance. The most important features of the BMU are colored in green, whereas the rest is colored in red. It can be seen that for the given data point, the closes features are included in the set of the most important features of its BMU. In this manner, we can generate a local explanation of the important features that contributed to deciding the outcome of the SOM algorithm.

3.4.3.3 Global Interpretability

Using the experiments above, we identified the model behavior of SOM, in terms or important features for each neuron in SOM map. Once we identify important features, then we can use them to explore and discuss how each feature behaves within a cluster. In this experiment, we used the neuron-wise important features and their value ranges for global interpretability to explain the clusters.

For each feature, we checked whether it is important for one cluster or multiple clusters. We observed that some features are not important for any cluster, whereas some are important for one or more clusters. The feature value ranges of important features were visualized against the cluster assignments. Further, u-maps were used

to check the separation between clusters. Figure 17 presents an example of the 'flag' feature of the KDD dataset. First raw, the first image shows the cluster separation of SOM neurons. The second image of the first row represents the feature value of the 'flag' feature across 3 clusters (component plans). It can be seen that the 'flag' feature value is different across 3 clusters. The third image of the first raw shows that the three clusters are well separated as there is a light color area that represents the distance between neurons. Lighter the area, better the separation between clusters. The second row of Figure 17 shows fine-grain visualization of feature value scale across clusters. It has to be noticed that a given cluster contains a set of neurons, and the feature value for a given feature can be different from one neuron to another, even within the same cluster. This information is essential for a domain expert to check whether how a given feature behave within a cluster. For the 'flag' feature, it shows a higher feature value (0.7-1.0) for cluster 0; for cluster 1, it shows an intermediate feature value range (0.45-0.52); for cluster 3, it shows a very low feature value range (0.15). Further, it shows the probability of having a specific feature value within a cluster as well. For example, for cluster 0, 90% of neurons belonging to cluster 0 show a 0.9-1.0 range for that specific feature.

3.4.3.4 Discussion

It has to be noticed that the desired outcomes and evaluation methods for explainable machine learning methods are different based on many factors, including domain areas, applications, user groups, expected performance criterion, and medium of explanation. Therefore, it is not easy to establish a set of generalized requirements or outcomes of explainable machine learning systems. Further, evaluating explainable algorithms and their effectiveness is complicated as there is no clear way of measuring it [51]. Especially in the unsupervised domain, there was no clear way of measuring

and comparing the quality of the explanation methods. One classic approach for that is doing a human study with existing unsupervised explainable ML approaches for a specific problem domain, which is out of this dissertation's scope. However, we explore the model-specific features, limitations, and usability of the proposed approach with other existing explainable unsupervised ML approaches, presented in Table 11.

When looking at Table 11, it can be noted that different unsupervised XAI methods have different usability, features, and limitations. The current literature of XUnML is mainly concentrated on clustering and dimensionality reduction. When looking at Explainable SOM, the main advantage of it comes from its many Visual Data Mining capabilities, which are described in Section III. All the other methods discussed above have very limited visual data mining capabilities, limiting their usage in tasks that require VDM capabilities. A human study will be performed in future work to analyze the above methods to explore the advantages and limitations of the above methods.

It is also necessary to understand the difference between SOM neural networks and typical Feed-Forward Neural Networks (FFNNs) in terms of the learning approach, visualization capabilities, and global/local interpretability. SOMs use the winner-take-all algorithm for training while preserving the input space's topological properties. Thus the trained set of neurons in SOM represents the topological properties of input data distribution. Whereas FFNNs use error-correction learning (such as backpropagation with gradient descent) for training which does not have the capability of representing the topological properties of input data using trained neurons. FFNNs are trained to perform classification and regression, whereas SOMs are trained to perform clustering tasks. As discussed in the previous section, SOM has many in-build visual data exploration approaches for visualizing feature behaviors, whereas FFNNs have very limited inbuilt VDM capabilities.

Table 11. Comparison between XUnML methodologies

Explainable SOM	**EXPLAIN-IT**	**Interpretable Trees**	**PCA**	**Variational AE**
Interpretation Approach:				
Model-specific	Model agnostic	Model-specific	Model-specific	Model-specific
Used for:				
Clustering	Clustering	Clustering	Dimension Reduction	Dimension Reduction
Data Distribution Visualization Capability:				
Inbuild capability to visualize input data distribution: High dimensional data space to a low dimensional grid (2 dimensions)	NA	NA	Inbuild capability to visualize input data distribution: High dimensional data to low dimensional data (2 dimensions)	NA
Visual data mining capabilities:				
Many visual data mining capabilities: histograms,component plane	Limited	Limited	Limited	Limited
Model quality evaluation:				
Can apply unsupervised quality matrix such as adjusted mutual information, adjusted random score, completeness, Fowlkes-Mallows, homogeneity, silhouette, and V-measure	Can apply unsupervised quality matrix such as adjusted mutual information, adjusted random score, completeness, Fowlkes-Mallows, homogeneity, silhouette, and V-measure	Splitting criteria such as information gain and entropy	Reconstruction error measurements (Variability)	Reconstruction error measurements (MSE)
Local vs Global Explanation:				
Both local and global explanations	Both local and global explanations	Both local and global explanations	NA	NA
Integrating clustering capability:				
Can work with any clustering algorithm	Can work with any clustering algorithm	NA	NA	NA
Time Complexity with respect to the number of training samples (n):				
O(n)	Depend on the model used for clustering	O($nlog_2n$)	O(n^3)	O(n)
Limitations:				
Depending on which type of distance metric used, the result may very	Model biases can occur, Explainability is dependant on other models makes this approach complicated	Split evaluation measure are required	Principle components of the model are not interpretable, Data should be standardize	High model complexity, Need expert knowledge base for training

The difference between FFNNs and SOMs in terms of global and local interpretability are: 1) The presented interpretation technique for SOMs generates local/global interpretations for clustering tasks, whereas the most popular interpretation techniques for FFNNs generate local/global interpretations for classification and regression tasks; 2) Local interpretability: Most popular local interpretation techniques used for FFNNs produce relative feature importance scores, whereas the presented technique for SOMs does not generate relative feature importance scores. It only generates a sorted features list indicating the most important features to the least important feature; 3) Global interpretability: Most popular global interpretation techniques used for FFNNs produce a set of IF-THEN rules for explaining the model behavior for different classes, whereas the presented technique for SOMs generates a set of component planes and feature value distributions for explaining how different features behave across different data clusters; 4) SOMs carry an inherent topological understanding of data and clusters. This inherent topology directly reflects notions of local and global belonging of data to clusters and addresses local vs global interpretability, unlike FFNNs that do not have the topological understanding of the data.

3.4.4 Contribution 1 (b): Findings, Discussion, and Future Work

We proposed a novel model-specific explainable method for the Self-Organizing Map (SOM) algorithm, generating local and global explanations. Through feature value perturbation, we evaluated the model fidelity and showed that the proposed approach identifies the most important feature used by the decision-making process of SOMs. We showed that the changing of features values of important features affects the model outcomes of SOMs. We presented the proposed approach as a strong candidate as a XUnML method by comparing it with current XUnML methods

in terms of model-specific features, limitations, and usability. In future work, the proposed approach will be further evaluated through a human study.

3.5 Contribution 1: Chapter Summary

This chapter presented the first contribution of the dissertation "Improving and Interpreting Self Organizing Neural Network". This contribution consisted of two sub-contributions: 1) A novel deep Self Organizing Neural Network algorithm with an improved feature learning capability and 2) A novel technique for interpreting Self Organizing Neural Network algorithm for unsupervised clustering.

Under the *first sub-contribution*, a **novel Deep Self Organizing Neural Network** architecture was presented. The presented Enhanced DSOM (E-DSOM) architecture showed improved performance compared to the initial DSOM in terms of 1) classification accuracy, 2) generalization capability, and 3) training time. E-DSOM was tested on three datasets and compared with DSOM. E-DSOM outperformed DSOM in terms of classification accuracy with improvements of up to 15%. Generalization capability was tested by adding noise to test data. E-DSOM outperformed DSOM at all noise levels (barring one instance with comparable results), evidencing better generalization capability. Finally, computational time was improved by gaining the same or better classification accuracies with a shallower model. E-DSOM showed training time improvements up to 19%. Therefore, empirical evidence supports our hypothesis.

Under the *second sub-contribution*, a **novel model-specific explainable method for the Self-Organizing Map** (SOM) algorithm was presented. The presented approach identifies the most important features for the decision-making process of SOM for clustering tasks. Then it generates both local and global explanations for clustering tasks using the identified important features. We showed that the changing

of features values of important features affects the model outcomes of SOMs. We presented the proposed approach as a strong candidate as an eXplainable Unsupervised Machine Learning (XUnML) method by comparing it with current XUnML methods in terms of model-specific features, limitations, and usability. The presented interpretable SOM based clustering approach will be extended to the proposed novel DSOM architecture in future work. Further, the proposed approaches will be evaluated through a human study and human readiness level while exploring the capability of using proposed approaches for real-world applications.

CHAPTER 4

IMPROVING AND INTERPRETING AUTOENCODER NEURAL NETWORK

4.1 Contributions and Published Papers

This chapter presents the ***Contribution 2, (a) and (b)*** ;

a. A deep Autoencoder neural network based framework for unsupervised feature learning and deep embedded clustering with improved robustness to network depth.

b. Interpreting the Autoencoders for Anomaly Detection

Papers supports this work:

1. **C. S. Wickramasinghe**, D. L. Marino, and M. Manic, "RX-ADS: Interpretable Anomaly Detection method using Adversarial ML for Electric Vehicle CAN data", 2022. (Under review in IEEE Transactions on Intelligent Transportation Systems).

2. ©[2022] IEEE. Reprinted, with permission from **C. S. Wickramasinghe**, D. L. Marino, and M. Manic, "ResNet Autoencoders for Unsupervised Feature Learning From High-Dimensional Data: Deep Models Resistant to Performance Degradation", in IEEE Access, vol. 9, pp. 40511-40520, 2021, DOI: 10.1109/ACCESS.2021.3064819.

3. ©[2022] IEEE. Reprinted, with permission from **C. Wickramasinghe**, D. Marino, and M. Manic, "Deep Embedded Clustering with ResNets", in Proc.

14th International Conference on Human System Interaction, IEEE HSI 2021, Poland, July 8-10. 2021.

4. ©[2022] IEEE. Reprinted, with permission from **C. S. Wickramasinghe**, D. Marino, K. Amarasinghe, M. Manic, "Generalization of Deep Learning For Cyber-Physical System Security: A Survey", in Proc. 44th Annual Conference of the IEEE Industrial Electronics Society, IECON 2018, Washington DC, USA, Oct. 21-23, 2018. DOI: 10.1109/IECON.2018.8591773.

5. ©[2022] IEEE. Reprinted, with permission from D. L. Marino, **C. S. Wickramasinghe**, and M. Manic, "An Adversarial Approach for Explainable AI in Intrusion Detection Systems", in IECON 2018 - 44th Annual Conference of the IEEE Industrial Electronics Society, 2018, pp. 3237-3243, doi: 10.1109/IECON.2018.8591457.

4.2 Introduction

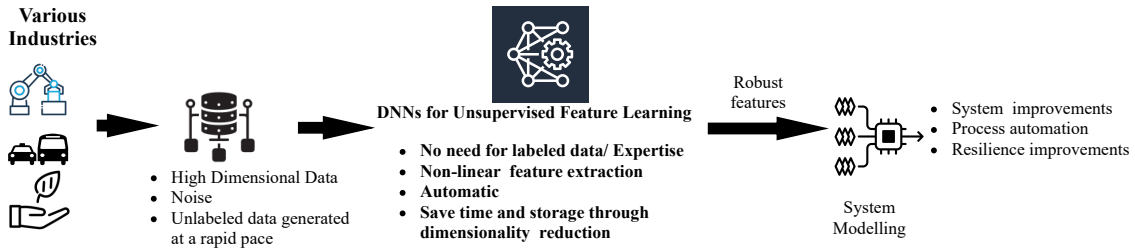


Fig. 18. The need for Deep Neural Networks (DNN) based unsupervised feature learning and its advantages

In this era of industrial big data, a massive amount of data is available to the public through various industries such as intelligent transportation [52] [53], power grids [2], cloud computing [54], and finance [55]. Knowledge extraction on these data is crucial for continuous improvements, process automation, and resilience improvements of these industrial systems [118]. Generally, the knowledge extraction of these

vast quantities of records is performed using machine learning approaches such as classification and clustering [119]. Even though data availability increases exponentially with time, these multi-variety data has many intricacies such as incompleteness, high-dimensionality, noise, and rarely labeled [120]. This work motivated by two main intricacies; high-dimensional and unlabeled data.

The first area of focus is the high-dimensionality of data. The reliability of knowledge extraction methods generally deteriorates due to the curse of dimensionality [121]. In other words, extracting relevant features leads to a reduced number of features that results in efficient knowledge extraction methods with high accuracy [122]. Therefore, when using high-dimensional data for data-driven machine learning tasks, it is necessary to capture only the relevant information [53] [123] [121]. Extraction of relevant features and reduction of input data dimensions are performed using various feature learning and dimensionality reduction techniques. This is achieved by performing non-linear mapping of input data into an embedded representation [124] [125] [126]. Since the embedded representation only contains relevant information, we can use these learned embedded representations to perform various down stream machine learning tasks such as classification and clustering.

The second area of focus is the abundance of unlabeled data. Real-world settings bring the challenge of dealing with high volumes of unlabeled data. The manual labeling process is time-consuming, expensive, and requires the expertise of the data [3]. Further, supervised learning not only is unable to take advantage of unlabelled data, but it also can result in biases by relying on labeled data. Therefore, unsupervised learning approaches such as unsupervised feature learning(feature extraction) and clustering has gained tremendous attention.

Many dimensionality reduction based unsupervised feature learning methods has been proposed in the recent literature. Widely used unsupervised feature learning

techniques include Principal Component Analysis (PCA) [127], Independent component analysis (ICA), Locally Linear Embedding (LLE) [127], Factor Analysis embedding, and SVD embedding. Recently, Deep Learning has shown remarkable performance in many areas. It has been successfully used to convert high-dimensional feature spaces into new embedded representations with relevant and robust features [121][3][128]. This effective transformation of the input data space to embedded space has been achieved through unsupervised deep learning methods such as deep convolutional autoencoders (C-AEs) [124][126]. Figure 18 shows current applications of Deep Neural Network (DNN) based approaches for various industrial applications such as process automation and resilience improvement. Further, deep learning-based clustering algorithms have gained huge attention due to the state-of-the-art performance of neural networks in many machine learning applications. This process is called deep clustering [129]. Deep clustering techniques boost the clustering algorithm performance by using the powerful feature extraction ability of Deep Neural Networks (DNNs) such as variants of Autoencoders (AEs).

Even though Deep learning had become the primary technique with state-of-the-art performance in many areas, they have the problem of vanishing gradient, i.e., when the network goes deeper, its performance gets saturated or even starts degrading rapidly. [130]. Because of this, the shallow counterparts can perform better than deep networks [130]. He et al. proposed residual blocks between layers to alleviate the problem of performance degradation [130]. These networks are called ResNets [131, 131, 132, 133, 134, 135]. While the ideas of adding residual connections do exist, there has been very limited work that has applied it to both unsupervised feature learning and deep clustering. I.e., the existing work does not address the effect of performance degradation of deep neural networks for unsupervised feature learning and deep clustering. Therefore in this work, we present a framework that

consists of residual blocks in AE architectures. We analyse the effectiveness of the presented framework for both unsupervised feature learning based classification and deep clustering. Further, we integrate a interpretation technique into the presented framework for performing interpretable anomaly detection. Thus this chapter consists of following three sections where first two covers the Contribution 2 (a) whereas third one covers Contribution 2 (b) of this dissertation.

- ResNet Autoencoder based unsupervised feature learning (Section 4.3)
- ResNet Autoencoder based Deep Embedded Clustering (Section 4.4)
- Interpretable anomaly detection using ResNet Autoencodes (Section 4.5)

The rest of the section is organized as follows. Section 4.3 presents the deep Autoencoder framework for unsupervised feature learning; Section 4.4 extends the framework for deep embedded clustering; Section 4.5 presents the interpretation technique for Autoencoder based anomaly detection; and finally, Section 4.6 provides a summary of the second contribution of this dissertation.

4.3 ResNet Autoencoder based Unsupervised Feature Learning

In this work, we use AEs to perform unsupervised feature learning. The unsupervised here refers to the unsupervised process of feature learning, i.e., learning of embedded representation from input data without using any labels. We used data labels only for the evaluation of learned embedded representations. We hypothesize that AEs with residual connections (RAE) will have improved resistance to performance degradation of learned features and improved feature learning capability compared to standard AEs. I.e., residual connections will alleviate possible information loss when increasing the number of hidden layers, and embedded representation will provide better separability for classification/clustering tasks.

Mainly for unlabeled data, it is challenging to decide the optimal number of hidden layers ahead when designing dimensionality reduction experiments. The proposed approach will always perform similar or better, even with a higher number of layers. Therefore, users have the advantage of designing few experiments with large networks, knowing that there is no adverse effect on the network's dimension reduction performance. To test our hypothesis, it is necessary to show that RAEs have lower performance degradation of unsupervised feature learning than AEs when increasing the networks' depth. We showed the effectiveness of the approach quantitatively by calculating the classification accuracy drop. I.e., we increased the number of hidden layers on both AEs and RAEs and checked how the classification accuracies on embedded representations change with the increase of the number of hidden layers. We used K Nearest Neighbor (KNN) for classification, as it allows us to check whether the same class samples are close to each other in the learned embedded representation (if the learned feature space learns a representation that encodes high-level concepts such as the classes of the input datasets).

As described before, this subsection presents the following:

- Address the effect of performance degradation of deep neural networks for unsupervised feature learning
- Performance comparison between proposed architecture (RAE) and standard AE based feature learning, using a different number of hidden layers on three different datasets.
- Performance comparison between widely used unsupervised dimensionality reduction methods

We compare the presented method against two relevant groups of methods (a total of 7 different methods). The first group is represented by Autoencoders, which

the literature indicates to be the most commonly used state-of-the-art deep learning based unsupervised dimensionality reduction architectures. We focus on standard Autoencoder and standard Convolutional Autoencoders because these are: 1) most frequently used; 2) other variants of AEs in the literature follow the principles of these two. Our objective was to evaluate how residual connections improve "feature learning", as such we compared against the same models with and without residual connections to evaluate improvement. The second group represents other types of feature extraction methods (five of those): Principal Component Analysis, Independent Component Analysis, Locally Linear Embedding, Factor Analysis, and Singular Value Decomposition.

4.3.1 Background and related work

This section consists of three subsections. The first subsection discusses widely used traditional unsupervised dimensionality reduction techniques. The second section discusses Autoencoder based deep learning approaches for dimensionality reduction. The third section discusses the theory behind residual connections.

Traditional unsupervised machine learning for dimensionality reduction: As discussed in the introduction, feature learning is essential for efficient and accurate machine learning tasks. Two types of dimensionality reduction based feature learning techniques exist, namely feature selection and feature transformation [136]. A subset of features from the original space is selected in feature selection, whereas in feature transformation (Dimension reduction), it generates an entirely new set of features. Both try to keep as much information in the data as possible while reducing the dimension. However, feature selection can be misleading as it assigns weights to individual features ignoring the correlation between features [136]. Therefore, feature transformation approaches are preferable. Widely used such dimension reduction

techniques are discussed below.

- Principal Component Analysis (PCA): A linear algorithm which preserves most of the data's variability in the latent space [127]. It minimizes the redundancy (measured through covariance) of data while maximizing information (Measured through variance) in the resulted space. Limitations include; 1) it only considers linear correlation, 2) input variables are assumed to be scaled at the numeric level [137].

- Independent Component Analysis (ICA): A linear transformation method that minimizes the dependence of the components of the transformed feature space [137]. Linearity is a major disadvantage of this method.

- Locally Linear Embedding (LLE): This is a non-linear algorithm that uses neighborhood preservation learning to generate subspace [127, 137]. However, this method has a high sensitivity for noise/outliers.

- Factor Analysis: This is the same as PCA in cases where the added noise is zero [138]. This method assumes that input data represent independent, random samples from a multivariate distribution. If variables are correlated, generated factors can be highly correlated [139].

- Singular Value Decomposition (SVD): This is mainly used for sparse data, i.e. when data contains many zero values. It converts the input data space to a latent representation with a reduced number of features while keeping the maximum information from the original space [140]. This approach is computationally expensive.

Unsupervised Deep Autoencoders For Dimensionality Reduction: The traditional concept of unsupervised learning was mainly limited to the idea of data

clustering and association rule mining. However, the expansion of deep learning methods and data mining combined with this era of big data has given a much broader perspective to traditional unsupervised learning. Therefore, unsupervised learning is used not only for clustering, but also for dimentionality reduction (also referred as unsupervised feature learning / deep embedded representation learning) [18][23], generative modelling [19] [20], and auto-regressive modelling [21] [22]. This work focuses on deep unsupervised feature learning, which is the process of transforming the input space to an embedded space, preferably a lower dimension compared to the input data space, using deep neural networks.

Many recent classification tasks use different variants of AEs, to learn feature representation from high-dimensional input data, where the learned (extracted) features will provide good separability for classification tasks. In these cases, feature extraction will be performed in an unsupervised manner, whereas classification will be performed on the extracted features in the reduced dimension in a supervised manner. Feature learning using variants of AEs has shown the following advantages: improve the robustness of feature learning [126], non-linear feature extraction [125], replacing handcrafted features with efficient algorithms for unsupervised feature learning [141], and reduces the time and storage space through dimensionality reduction [142].

The variant of deep AEs has been successfully used for deep embedded clustering tasks that perform feature learning and clustering simultaneously. In the past, clustering and feature learning were performed sequentially, i.e., it embeds the input space to a latent space and then performs clustering on the embedded space [23] [143]. With deep embedded clustering, it performs a joined optimization of feature learning (dimensionality reduction), and clustering [23]. For example, in [144], the authors have presented a deep clustering approach using fully connected convolutional AEs. They argue that the embedded representations extracted from an encoder may not be

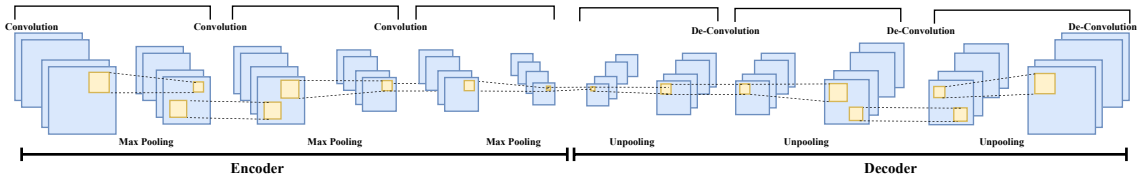


Fig. 19. Standard architecture of Stacked Convolutional Auto-Encoder.

discriminative enough for efficient clustering. To overcome that, they have proposed a soft k -means model on top of the encoder to make a unified clustering model.

Residual Connection Within Deep Neural Networks: He et al. raised the awareness towards the problem of performance degradation [131]. I.e., when the network's depth increases, the network's performance will start to saturate, and eventually, it can even deteriorate [132]. This is not caused due to the over-fitting, but by the vanishing gradient of deep neural networks [132].

This problem has been addressed by various network designs networks such as ResNets [131, 133], Highway Networks [134], and DenseNets [135]. All these networks use the same design principle, i.e., skip connections or residual connections [132]. These networks with skip connections have consistently shown state-of-the-art performances in different neural network typologies [131, 134]. Other advantages of skip connection includes better easier training [132], numerical stability and easier optimization [145] [132]. Empirical evidence has shown that these deep architectures with skip connections should not produce a large error than their shallow counterparts [131, 133].

4.3.2 Proposed Approach: ResNet Autoencoder Based Feature Learning for Deep Embedded Classification

This section discusses the stacked ResNet Autoencoder (RAE) based feature learning approach for classification. Figure 19 presents the standard C-AE architec-

ture with multiple convolution and max-pooling layers with multiple filters.

In this work, we implemented standard and convolutional AEs (AEs and C-AEs) with residual connections. Our intent was to convey the advantages of adding residual connection into AE networks to improve feature learning capability. Therefore, we designed a simple and reproducible experiment, which can run in a reasonable amount of time. We introduced residual connection into the AE architecture and presented the novel Residual Autoencoder (RAE) framework for deep embedded classification. We call its convolutional counterpart C-RAE. The proposed framework is presented in Figure 20 where (a) presents the training of presented RAE and (b) represent the classification task on learned features.

As similar to AEs, RAEs are trained to regenerate their inputs from its output (Figure 20 (a)). The input sample x is typically a n dimensional vector. Therefore the input layer consists of n neurons. Since the RAE network is trained to reconstruct the input, the output layer has the same number of neurons as the input layer. The hidden layers consist of m neurons.

Similar to AE, RAE also consists of two phases, i.e., encoding phase and decoding phase [6], [146]. For a high-dimentional input x, the encoder E computes a hidden representation $z = E(x)$. The decoder D reconstructs the hidden representation back to the high-dimensional input space $y = D(z)$. Both encoder and decoder have several hidden layers, making a deep (stacked) RAE.

For the decoder, each hidden layer is a non-linear mapping of the form $\sigma(Vz+c)$, where σ is an activation function such as sigmoid, tanh, softsign, or Relu [146]. V is the weight matrix. We use superscripts $V^{(l)}$ to denote the weight matrix that corresponds to layer l. In convolutional neural networks (C-RAE), the matrix multiplication is replaced by a convolution operation and max-pooling (see Fig. 20).

For the encoder, each hidden layer l is composed by a non-linear mapping $f(\cdot)$

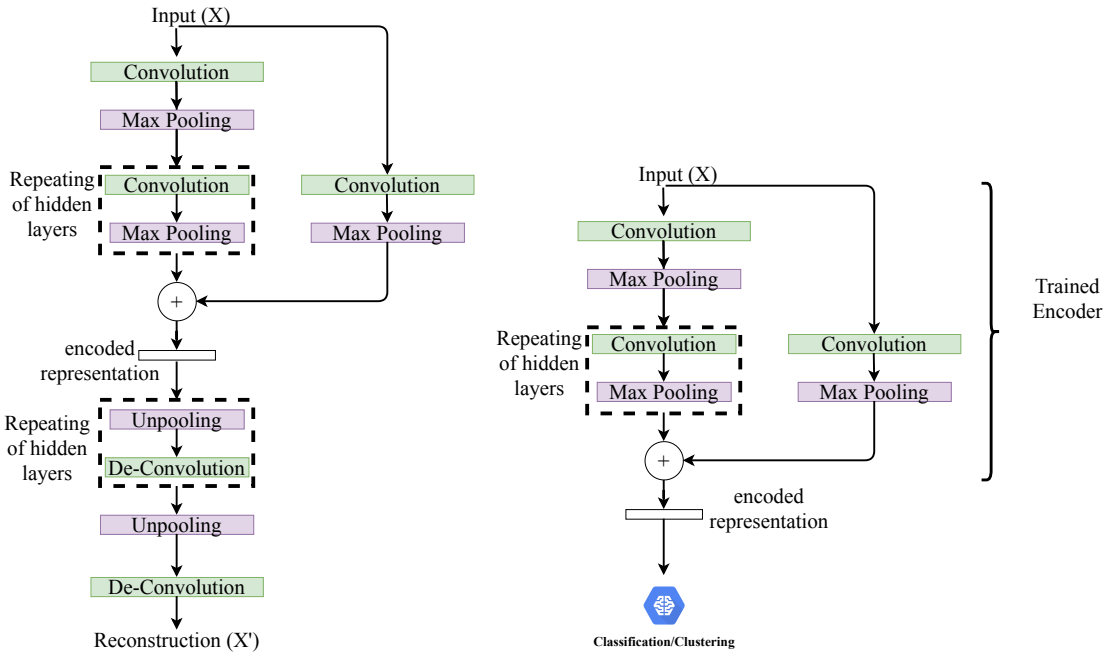


Fig. 20. RAE based feature learning (a) Training of C-RAE, (b) C-RAE based classification/Clustering

and a residual connection $r(\cdot)$. Each hidden representation $h^{(l)}$ in a hidden layer l is computed follows:

$$h^{(l+1)} = r\left(h^{(l)}\right) + f\left(h^{(l)}\right) \tag{4.1}$$

The residual connection $r(h) = W_r h$ is a linear mapping that ensures the dimensions match the output of the function f. The function f can be thought of as a smaller network with F number of layers. Each layer in f is a non-linear mapping $\sigma(Wh+b)$, similar to the decoder layers. W is the weight matrix, and we use superscripts $W^{(l,j)}$ to denote the weight matrix that corresponds to layer l and sub-layer j. For C-RAEs, both matrix multiplications (W_r and W) are replaced by convolution operations and max pooling (see Fig. 20).

Similar to AE, the loss function J_θ of the RAE network is also computed using

Table 12. Algorithm for training the proposed RAE

Algorithm I: RAE Training

Inputs: Training set of images (X)
Outputs: Trained $RAE, Encoder$

Random Weight initialization
for each epoch e do
  for $i = 1...T$ **do** //number of training samples
    $x_i \leftarrow$ pick random input record from X
    $h \leftarrow x_i$
    for $l = 1...L_e$ **do** //each hidden layer l in encoder
      $h_f \leftarrow h$
      for j=1...F **do** //each layer j in f
        $h_f \leftarrow \sigma(W^{(l,j)} h_f + b^{(l,j)})$
      end for
      add residual connection to the hidden activation h_f
        $h \leftarrow W_r^{(l)} h + h_f$
    end for
    $y_i \leftarrow h$
    for $l = 1...L_d$ **do** //each hidden layer l in decoder do
      $y_i \leftarrow \sigma\left(V^{(l)} y_i + c^{(l)}\right)$
    end for
  end for
  Compute the reconstruction loss:
    $J_\theta = \frac{1}{T} \sum_{i=1}^{T} (x_i - y_i)^2$
  Perform one-step of the optimizer:
    $\theta = \text{argmin}_\theta(J_\theta)$
end for

Algorithm II: Deep Embedded Classification using KNN

Inputs: Training set(X), Training labels (Y), Testing set(X'), Testing labels (Y'), Trained $Encoder$
Outputs: Accuracy

$z_{train} \leftarrow Encoder(X)$ %convert training data to embedded representation
$z_{test} =\leftarrow Encoder(X')$ %convert testing data to embedded representation
Initialize a list (z_y) to store predicted class label
for each sample (i) in z_{test} do
  Initialize a list ($list$) to store $< distance, class >$ pairs
  $dist \leftarrow 0$, $label \leftarrow 0$
  for each sample (j) in z_{train} do
    $dist \leftarrow \|z_{test,i} - z_{train,j}\|$
    $label \leftarrow classlabelofhx$
    $list \leftarrow append < dist, label >$
  $class\_list \leftarrow$ find list of labels of K nearest neighbors
  $predicted\_class \leftarrow mode(class\_list)$
  $z_y =\leftarrow append(predicted\_class)$
  end for
end for
calculate accuracy using z_y and Y'

the difference between input(x) and the output(y), I.e. the error.

$$J_\theta = \frac{1}{T} \sum_{i=1}^{T} \|x_i - y_i\|^2 \tag{4.2}$$

where x_i is the ith input sample, y_i is the output for ith input sample, θ denotes the set of parameters of the autoencoder (weights and biases).

The RAE is trained to minimize the above loss function with T training samples using error-back-propagation. The pseudo-code for RAE training is presented in Algorithm I.

Similar to AE, the dimension of the hidden representations (z) of RAE can be smaller or larger than the dimension of x. When the hidden representation is small, the RAE performs dimensionality reduction (data compression) [146].

The encoded value z is viewed as the extracted feature or the hidden representation for the input data. Once the encoder converts the input samples (x) to an embedded representation z, then classification or clustering can be performed on this latent space (shown in Figure 20 (b)).

For classification purposes, any supervised classification algorithm can be integrated at the end of the encoder (Figure 20 (b)). For this experiment, the K-Nearest Neighbor algorithm (KNN) is used. Algorithm II presents the KNN based deep embedded classification.

As presented in Algorithm II, the trained RAE's encoder is used to generate an embedded representation of train and test data (line 1-2). Then class labels for test data can be predicted by comparing each test record with all the train records and find the mode class label of K nearest train records (Algorithm II line 4-12). The distance between a test record and a train record should be calculated using a distance calculation method to find the nearest neighbors. For this experiment, Euclidean distance is calculated:

$$dist(z_{test}, z_{train}) = \sqrt{\sum_{i=0}^{dim} (z_{test,i} - z_{train,i})} \tag{4.3}$$

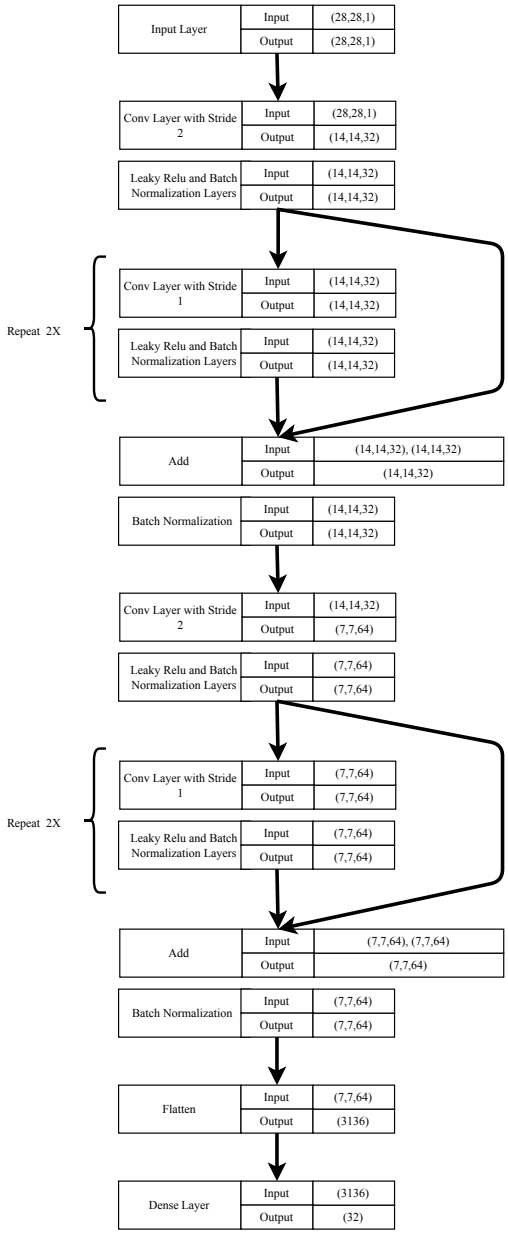


Fig. 21. Architecture

where z_{test} is the test record, z_{train} is the train record, and dim is the dimension of the embedded feature space (z). However, it is possible to use other distance calculation methods such as Minkowski, Manhattan, Mahalanobis, and cosine. Finally, predicted labels and actual labels are compared to calculate the accuracy of the KNN

algorithm.

4.3.3 Experiment and Results

This section discusses the experiments and results. First, we discuss the datasets used for experimental evaluation. Then, we present the experimental set-up and architecture details of the networks. Finally, we discuss the results of the experiment with a comparison between existing dimensionality reduction methods.

Datasets: Three datasets were used for experimental evaluation: 1) MNIST [147], 2) CIFAR10 [148], and 3) Fashion MNIST [149]. All the datasets were scaled to the 0-1 range. These benchmark datasets were selected due to their relatively high dimension and reasonable training time with deep networks. Datasets were directly obtained from the Keras library [150].

The **MNIST dataset** consists of hand-written digits (0-9), where each digit is an image of 28 X 28 pixels in size. The complete MNIST dataset was used, which consist of 55000 train images and 10000 test images.

The **Fashion MNIST dataset** benchmark dataset consist of images used for clothing classification. It consists of images with 28 X 28 pixels in size. Class labels include (T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, Ankle boot) The complete Fashion MNIST dataset was used, which consists of 60000 train images and 10000 test images. Images belong to 10 classes.

The **CIFAR10 dataset** consists of color images of 32 X 32 pixels in size. These images correspond to 10 classes (airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck). The complete CIFAR10 dataset was used, which consist of 50000 train images and 10000 test images.

Hyper-parameters and architectural details: To maintain consistency in the experiments, all the architectures were kept constant across datasets when in-

Table 13. Classification Accuracies of models for different datasets

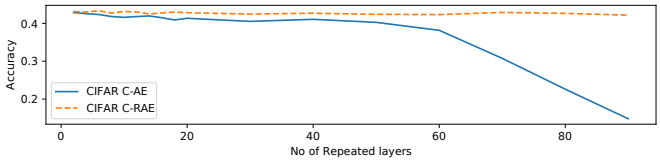
Dataset	Model	No of Repeated Layers											Performance
		2	**6**	**10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**Degradation**
MNIST	***C-AE***	0.9849	0.9836	0.9830	0.9824	0.9805	0.9767	0.9806	0.9482	0.9379	0.5559	0.3752	61.90
	C-RAE	0.9846	**0.9853**	0.9845	0.9835	0.9776	0.9768	0.9762	0.9761	0.9753	0.9764	0.9770	0.86
CIFAR	***C-AE***	0.4285	0.4233	0.4158	0.4134	0.4052	0.4107	0.4026	0.3817	0.3076	0.2262	0.1480	65.46
	C-RAE	0.4313	**0.4333**	0.4312	0.4281	0.4246	0.4268	0.4239	0.4231	0.4289	0.4263	0.4217	2.68
Fashion	***C-AE***	0.8844	0.8839	0.8838	0.8795	0.8785	0.8600	0.8558	0.8078	0.7875	0.6805	0.5892	33.38
MNIST	***C-RAE***	**0.8858**	0.8850	0.8826	0.8805	0.8751	0.8750	0.8733	0.8692	0.8698	0.8712	0.8683	1.97

Table 14. Comparative Analysis

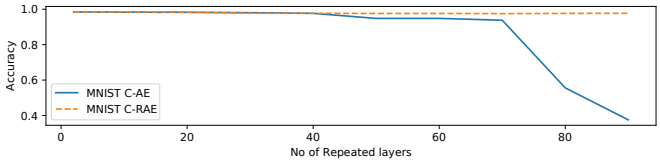
Dataset	KNN applied after unsupervised feature extraction (Embedded classification)									KNN on original
	AE	**RAE**	**C-AE**	**C-RAE**	**PCA**	**Standard LLE**	**ICA Embedding**	**Factor Analysis Embedding**	**Truncated SVD Embedding**	**high-dimensional feature space**
MNIST	0.9745	0.9758	0.9849	**0.9853**	0.9758	0.9684	0.9713	0.9621	0.9755	0.9688
Fashion MNIST	0.8599	0.8617	0.8844	**0.8858**	0.8524	0.8126	0.8538	0.8499	0.8517	0.8552
CIFAR10	0.4182	0.4201	0.4285	**0.4333**	0.4039	0.2831	0.4134	0.4070	0.4019	0.3398

creasing the number of layers. Only two filters were used with size 32 and 64. The size of the embedded representation is kept at 32. The number of layers were increased by repeating the convolution layer and pooling layer for a given filter size. For this experiment number of repeating layers were increased from 2 to 90 for each filter. Optimizer (adadelta) and K(5) were kept constant for all the experiments across datasets. Batch normalization and leakyRelu was used to improve model performance. For illustration purposes, the MNIST dataset architecture with two filters (32,64) and 2 repeats is presented in Figure 21. For a given number of repeats (f), the total number of hidden layers is 2+(f*no. of filters).

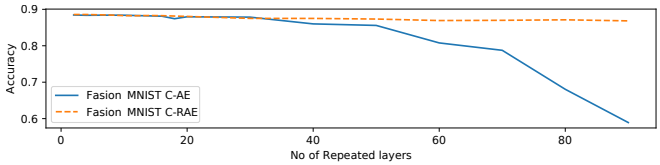
Classification Accuracy: The trained autoencoder models were used to generate the embedded representation for the datasets. These embedded representations were used for the classification using the KNN algorithm, i.e., encoder followed by KNN used as the classification network. Each experiment was repeated five times,



(a) CIFAR



(b) MNIST



(c) Fashion MNIST

Fig. 22. Classification accuracy vs number of hidden layers

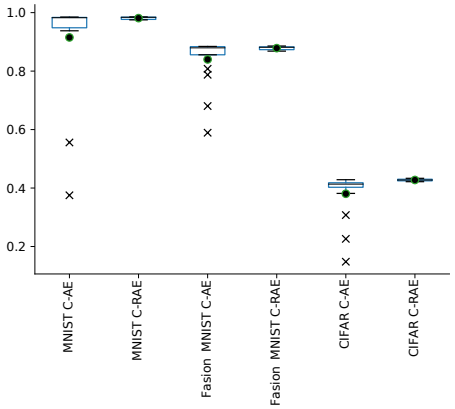


Fig. 23. Accuracy distribution

and the average performances were recorded.

Table 13 shows the deep embedded classification accuracy obtained using the two models, C-AE and C-RAE, for different datasets when increasing the number

of hidden layers. When comparing all the models, C-RAE showed improved accuracy compared to C-AE for all three datasets (highlighted values in Table 13). Table 13, column 6 shows the classification performance of KNN on the original high dimensional data. It can be seen that both C-AE and C-RAE based deep embedded classification showed better accuracies than just applying KNN on original data. This infers that these deep neural network models convert original data into embedded representations that are more suitable than using the original input data for down-stream tasks such as classification.

Figure 22 shows a plot of the accuracies against no of repeated layers. When increasing the number of layers, a small fluctuation of accuracy was observed for small models (up to 20 repeated layers) for all the datasets. For large models, when increasing the no of hidden layers, the accuracies started to decrease. However, C-RAE showed significantly lower degradation compared to C-AE. Therefore, it can be inferred that C-RAE based embedded representations are less likely to under-perform when increasing the number of layers.

Figure 23 shows classification accuracy distribution in box and whisker graphs for all three datasets when increasing the number of layers. The height of the box plot indicates the variability of classification accuracy for each model. Anything outside the normal distribution is marked as outliers shown as "X" marks. A shorter box and whiskers plot indicates low variability of classification accuracies. For all the C-RAE, the whiskers are shorter than C-AEs, and there are no outliers. It shows that C-RAE has consistent performance with low variability when increasing the number of layers. Mean values are marked with "O". All C-RAEs mean values are higher than the C-AEs. These observations show that with the change of the number of hidden layers, C-RAEs have consistent performance, whereas, for standard C-AEs, a thorough cross-validation process is needed.

The last column of Table 13 shows the overall performance degradation for deep embedded classification when increasing the number of hidden layers. The performance degradation (PD) was calculated as the percentage accuracy drop when increasing the number of layers:

$$PD = \frac{(MaximumAcc - MinimumAcc) * 100}{MaximumAcc} \tag{4.4}$$

Both C-RAE and C-AE showed some performance degradation for all three datasets. C-AE without residual connection showed 33.38% - 65.46% performance degradation whereas C-RAE showed 0.86% - 1.97% performance degradation. Based on the experimental result, it can be seen that residual connections reduce possible performance degradation significantly.

Comparison Between Widely used Dimensionality Reduction Methods:

Table 14 presents the performance comparison between proposed approaches and widely used unsupervised dimensionality reduction methods. We compared the proposed approach with two state-of-the-art deep neural network based dimensionality reduction methods (AE and C-AE) and five most widely used conventional dimensionality reduction methods in the recent literature (PCA, LLE, ICA, Factor Analysis embedding, Truncated SVD embedding). As described in the previous section, all these methods were used to convert the high dimensional input space to an embedded representation of 32 features. Then, KNN was used to perform the classification on the embedded representations. Further, KNN was ran to calculate the classification accuracy on the original high dimensional space (last column of Table 14). For MNIST, all the embedded classification approaches except LLE and FAE showed better accuracies compared to applying KNN on the original high dimensional feature

space. C-RAE showed the highest accuracy (0.9853) for MNIST. For Fashion MNIST, only deep neural network based embedded classification showed higher accuracy compared to KNN. C-RAE showed the highest accuracy (0.8858) for Fashion MNIST. For CIFAR10, all the embedded classification methods except LLE showed higher accuracy compared to KNN. C-RAE showed the highest accuracy (0.4333) for CIFAR10. When comparing AEs and RAEs on all three datasets, RAEs showed slightly better performance. When comparing RAE and C-RAE, C-RAE showed better accuracy on all three datasets. The results of Table 13 and Table 14 infers that deep neural network models convert original data into embedded representations that are more suitable than using the original input data for down-stream tasks such as classification, and C-RAE based embedded representations are less likely to under-perform when increasing the number of layers.

Overall discussion and future work: Our hypothesis was that when adding new layers to standard AEs, their ability for effective feature learning degrades. Through accuracy comparison in Table 13, we confirmed that addition of residual connections to AEs (RAEs), improved their overall classification accuracy without incurring significant performance degradation (relative to standard AEs).

Through a comprehensive comparison of widely used unsupervised dimensionality reduction methods in Table 14, we demonstrated that the C-RAE outperforms widely used feature learning methods such as standard AE, KNN, PCA, LLE, ICA, Factor Analysis, and SVD by 1%-3% improvements of classification accuracy. In addition to the accuracy improvement over standard CAE, C-RAE showed significantly lower performance degradation of classification accuracy (less than 3%) compared to CAE (33%-65%), when increasing the network depth. These results evidenced the advantages and the overall superiority of C-RAEs for unsupervised feature learning compared to standard AEs and widely used traditional methods.

Finally, by implementing the novel RAE framework presenting here, one does not need to go through a trial and error process of finding the best architecture. Instead, one can safely go with more layers in case a more complex model is required for improved overall performance while not sacrificing the dimensionality reduction performance.

The experiment was tested using three datasets that can be trained with deep neural networks within a reasonable amount of time. However, it has to be noticed that the advantage of using a deep neural network is more prominent when dealing with more complex datasets. Therefore, in future work, the framework will be tested with more complex datasets, which are high in dimension and number of data records.

4.3.4 Contribution 2 a): Findings, Discussion, and Future Work

In this subsection, we tackle the performance degradation problem of automated deep unsupervised feature learning. We introduced an unsupervised deep learning framework, consisting of ResNet Autoencoder (RAE) and its convolutional version C-RAE, that allows making deeper neural networks while not sacrificing its dimensionality reduction based feature learning performance. In this way, we improve resistance to performance degradation compared to standard Autoencoders (AEs) for feature learning. The performance of RAE on learning deep embedded representations was evaluated on a classification task using KNN. RAE was compared against AE while increasing the number of hidden layers. We did this comparison on three benchmark datasets. We demonstrated that C-RAE showed the highest accuracy on all three datasets. At the same time, C-RAE based classification only showed 0.86% to 2.68% performance degradation, which is significantly lower than the performance degradation showed by standard C-AE (33.38% - 65.46%).

The empirical results confirmed that RAE reduces performance degradation of

deep embedded representation based classification. This framework allows users to design fever number of experiments knowing that larger networks will not affect the network performance, especially when dealing with unlabelled data where the optimal network size is challenging to decide. Further, the classification accuracy distribution showed that RAE models perform better in terms of mean accuracy and accuracy variance (low variance), making them more suitable for deep embedded classification tasks than AE. Finally, we compared RAEs with widely used dimensionality reduction methods and showed that C-RAE outperforms on all experimented datasets. As future work, this framework will be integrated to real-world CPS setting and explore how to adapt this framework for specific needs of CPSs.

4.4 ResNet Autoencoder based Deep Embedded Clustering

Clustering is the method of grouping a collection of data records based on some similarity criteria such that records in the same category are similar to each other compared to records in another category [151, 119, 152]. It is a major task in exploratory data analysis and a commonly used technique in machine learning for many fields such as image analysis, bio-informatics, finance, and natural language processing. All the clustering methods primarily follow two steps: 1) picking initial clusters randomly and (2) optimize the clusters gradually, until an optimal solution is reached [153, 152]. Widely used non-neural network based clustering techniques include K-Mean, Mean shift, DBSCAN, Gaussian Mixture Models (GMM), and hierarchical clustering [154, 155]. Recently, deep clustering-based clustering algorithms have gained huge attention due to the state-of-the-art performance of neural networks in many machine learning applications. This process is called deep clustering [129].

Deep clustering techniques boost the clustering algorithm performance by using the powerful feature extraction ability of Deep Neural Networks (DNNs) such as vari-

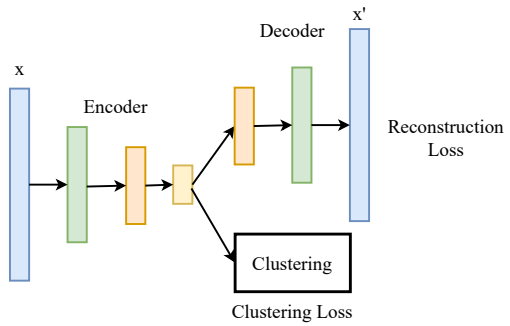


Fig. 24. Deep Embedded Clustering (DEC)

ants of Autoencoders (AEs). DNNs have the capability to convert input data space into a cluster-friendly feature space through non-linear transformations [156]. Recent approaches have shown that dimensionality reduction and representation learning techniques can be used to transform the input data space into an embedded latent representation. Then, the clustering and classification tasks can be performed on the embedded space more efficiently compared to direct use of input data space. This AE-based deep embedded clustering approach was initially proposed by Song et al. in 2013 [157, 129]. They have proposed a new object function and embedded it in the Autoencoder model. This allows a joint optimization of the Autoencoder's non-linear mapping (minimizing reconstruction error) and clustering (updating the cluster centers). In [158], the authors proposed a novel clustering method, Deep Embedded Clustering (DEC), which concurrently learns embedded representation and cluster assignments using DNNs. The idea of DEC is presented in Figure 24. This approach gradually improves the clustering performance and the learned feature representation, resulting in significant improvements over cutting edge clustering methods [158].

Even though deep learning has shown state-of-the-art performance in many machine learning tasks, when increasing the network's depth, their performance gets saturated or even degrade. This happens due to the vanishing gradient problem of

deep networks, resulting in shallow counterparts performs better than their deep neural networks [130]. In [130], authors proposed residual blocks (ResNets) within the layers of deep neural networks to avoid possible performance degradation (Figure 25) [130]. In our previous chapter, we explored the performance drop of DNNs for unsupervised feature learning [27]. We analysed the change of classification accuracies on latent representations when increasing the network depth (no of layers) for both AEs and RAEs (AE with residual connections: RAEs). Our experiments evidenced that compared to AE, the RAE has improved feature learning capability and reduces classification performance on learned features.

In this subsection, we perform DEC with ResNets (RDEC). While the idea of ResNets does exist, ResNets mainly has been used for classification tasks. There has been very limited work that had been done on DEC with ResNets. Moreover, the current work does not illustrate how performance degradation of neural networks affects DEC. Therefore, in this work, we introduced improved DEC approach in which we perform DEC by introducing residual connections. Then we use the joint optimization approach proposed by Xie at el. in [158] for performing DEC. To illustrate the advantage of having residual connections, we performed DEC using AE and RAEs. We performed DEC while increasing the number of network hidden layers and calculated the clustering accuracy drop of AE and RAE. The results showed that the RDEC has less clustering accuracy drop compared to DEC. The major advantage of this strategy is that the use of ResNets for DEC allows practitioners to reduce possible clustering performance degradation when designing large neural networks.

4.4.1 Methodology: ResNet Based DEC

This section discuss the DEC with Resnet Autoencoder (RDEC). The presented algorithm consist of basic architecture of AE (Figure 26 (a)) with residual connections

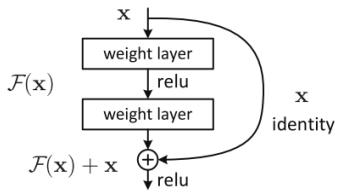


Fig. 25. Resnet Architecture [158]

(Figure 25). The proposed RAE architectures are presented in Figure 26 (b). Figure 26 (c) presents the proposed DEC with RAE (RDEC).

As similar to AE, RAE is trained to reproduce the input image from its output (Figure 26 (b)). The input (x) is a vector with n dimensions. I.e., the input layer consists of n neurons. RAE reconstructs the input data sample from the output of the network; therefore, the output layer also has n neurons. The hidden layers consists of m neurons. Between the input layer and the hidden layer, the weight matrix W' has the size $\mathbb{R}^{m \times n}$. Between the hidden layer and the output layer, the weight matrix W' has the size $\mathbb{R}^{n \times m}$.

Both AE and RAE architectures have an encoder and decoder, consisting of many hidden layers creating deep AEs and deep RAEs. Therefore, training of them consists of two stages, i.e., encoding stage and decoding stage. The input is transformed into an embedded representation by the encoder. In the encoding phase, the input is transformed by the first hidden layer as follows:

$$h = f\left(Wx + b\right) \tag{4.5}$$

where f is the activation function and b is the bias term. For all the other hidden layers, the h generated by previous hidden layer act as the x to the next hidden layer (line 5-7 of Algorithm I).

In the decoding phase, the embedded representation h is reproduced back to the

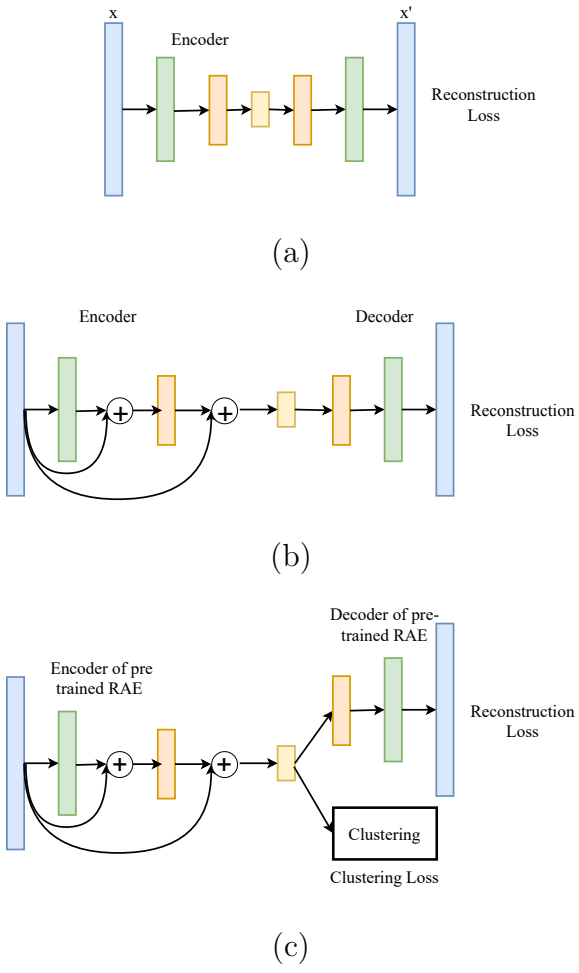


Fig. 26. RAE based deep embedded representation learning (a) AE, (b) RAE, and (c) RDEC

original input data record as follows:

$$y = g\left(W^{'}h + b^{'}\right) \tag{4.6}$$

In here, g denotes the activation function of the decoder. This will be performed for all the hidden layers of decoding phase (line 10-12 of Algorithm I). Encoding functions and decoding functions are non-linear mapping functions such as sigmoid, tanh, softsign and Relu [146]. For this experiment, we were using Relu as the no-linear mapping.

When adding a residual connection, the intermediate encoded representation (h) and input (x) should add together into one representation. To do that, both representations must be in same dimension. Therefore, a non-liner mapping of input x to a dimension which matches the intermediate encoded representations is performed. The new encoded representation is calculated as follows:

$$y = W^{''}x + y \tag{4.7}$$

The J_θ denotes the loss function of the RAE network. It is computed using the difference between input data record and the reconstructed generated by the decoder(y).

$$J_\theta = \frac{1}{tot}\sum_{i=1}^{tot}(x_i - y_i)^2 \tag{4.8}$$

In above, x_i denotes the ith input data record, tot is the total number of data records in the input data, y_i is the reconstruction of ith input sample generated by the decoder, parameter set of the encoder is denotes as θ (ex: $W, W^{'}, W^{''}, b, b^{'}$).

The error-back-propagation was used to minimise the above loss function during training. The algorithm is resented in Algorithm I in Table 15.

The training of the network does not require any class labels or prior knowledge of input training data. The h is the extracted feature or the hidden representation generated from the the input data x. The dimension of h can be different from the dimension of input. When the size of h is small, the process is known as dimensionality reduction, which is a widely used data compression technique.

Then the trained encoder is used for DEC technique. The ResNet based DEC (RDEC) is presented in Figure 26 (b) and Figure 26 (c). The encoder is used to convert the input data samples (x) to a hidden representation $X_{embedded}$, then clustering is performed on the embedded representation. For DEC, any clustering algorithm

Table 15. Pseudo-code for training of RAE

Inputs to the algorithm: Set of training samples (X)

Outcomes: Encoder, Trained ResNet AE

Network parameter initialization (Weights)
FOR each epoch DO
  FOR each samples in training data DO
    $x \leftarrow$ randomly pick an input data sample from X
    $h \leftarrow x$
    FOR each layer l in hidden layers of encoder DO
      $h = f\left(W^l h + b^l\right)$
      compute the residual connection:
        $h = W'^l x + h$
    END FOR
    $y \leftarrow h$
    FOR each layer l in hidden layers of decoder DO
      $y = g\left(W'^l y + b'^l\right)$
    END FOR
    Calculate the error of reconstruction using the loss function:
      $J_\theta = \frac{1}{tot}\sum_{i=1}^{tot}(x_i - y_i)^2$
    Perform one-step of error-back-propagation using a optimizer:
      $\theta = arg_\theta min(J_\theta)$
  END FOR
END FOR

can be used. This is done by integrating a clustering algorithm into the encoder so that encoder output will be fed into the clustering algorithm. In this experiment, the DEC architecture proposed by Xie et al. is used.

It has to be noticed that it is not mandatory to add residual connections for each and every hidden layer. They can be added only to some selected hidden layers. Further, these residual connection does not have to come from input x. It can come from some intermediate layer outputs h as well. Various types of shortcut connections have been proposed in the past [159].

Table 16. Hyper-parameters of models

Dataset	Number of neurons in each layers of the network	Number of Neurons in the output layer
MNIST	2 layers: 2000	10
Fashion MNIST	4 layers: 500, 500, 2000	10
	6 layers: 500, 500, 500, 500, 2000	
	8 layers: 500, 500, 500, 500, 500, 500, 2000	
	10 layers: 500, 500, 500, 500, 500, 500, 500, 500, 2000	
CIFAR10	12 layers: 500, 500, 500, 500, 500, 500, 500, 500, 500, 500, 2000	10

Table 17. Clustering accuracies of DEC and RDEC

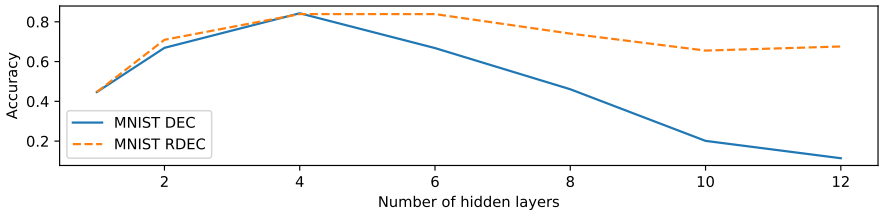
Dataset	Model	Number of Hidden Layers						
		1	**2**	**4**	**6**	**8**	**10**	**12**
MNIST	DEC	0.4467	0.6690	**0.8429**	0.6678	0.4607	0.2011	0.1139
	RDEC	0.4467	0.7095	**0.8388**	0.8387	0.7406	0.6552	0.6758
Fashion MNIST	DEC	0.5243	0.5306	0.5335	0.5010	0.5070	**0.5572**	0.2635
	RDEC	0.5243	0.5497	0.5572	**0.5999**	0.5931	0.5775	0.5704
CIFAR10	DEC	0.1662	0.1692	**0.1862**	0.1511	0.1314	0.1493	0.1272
	RDEC	0.1662	0.1653	0.2117	**0.2224**	0.2215	0.2150	0.2137

4.4.2 Experiment set up Results and Discussion

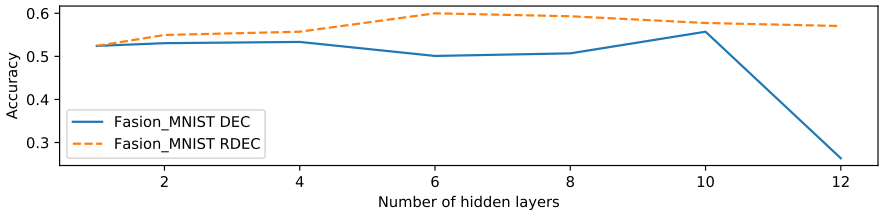
This section discusses the setup of the experiments, the data sets used, and the results. The performance comparison was performed between DEC and RDEC while increasing the network's hidden layers. The architectures (number of layers and neurons in each layer) were kept the same for all the datasets for the simplicity of the experiment. For this experiment, we used the same benchmark datasets used in for ResNet based feature learning in the previous section: 1) MNIST, 2) Fashion MNIST, and 3) CIFAR.

4.4.2.1 Hyper-parameters and architectural details

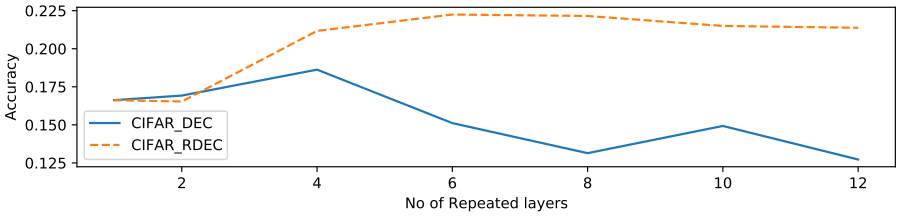
Hyper-parameters of the models are presented in Table 16. For keeping the simplicity of this experiment, we used the same architecture for all the datasets, except for the number of neurons in the final hidden layer of the encoder. It has to be noticed that the number of neurons in the final hidden layer of the encoder is



(a)



(b)



(c)

Fig. 27. Clustering accuracy vs number of hidden layers . (a) MNIST, (b) Fasion MNIST, and (f) CIFAR

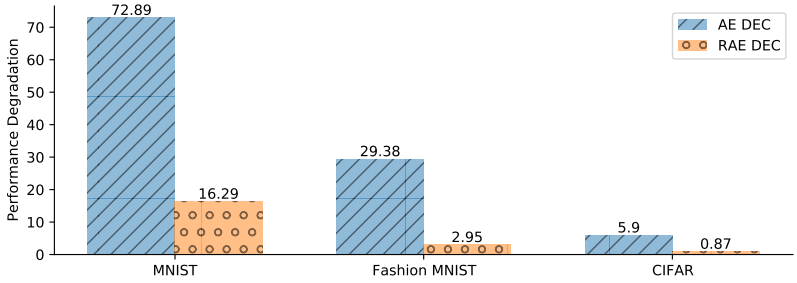


Fig. 28. Clustering Accuracy distribution of DEC

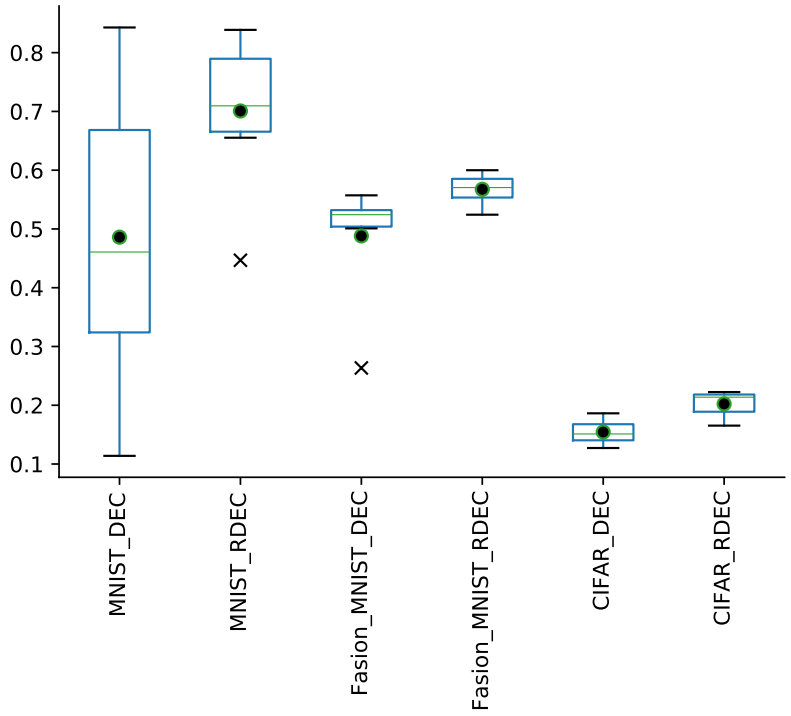


Fig. 29. Performance Degradation for DEC .

equivalent to the number of classes for a given data set. The detailed explanation on this is presented in original DEC paper [158]. For example, RDEC for MNIST has 10 neurons at the last hidden layer of the encoder. During clustering, both mean squared error (MSE) and KL divergence were minimized simultaneously. KL divergence is minimized between the auxiliary distribution and target distribution as described in [158]. The number of layers was increased from 1 to 12. For simplicity, one residual connection was added from input to the hidden layer just before the last hidden layers. After training of AE and RAE, the encoders of both models was integrated with the clustering algorithm (DEC), which is proposed in [160] by Xie at el. The clustering performance was compared between DEC and RDEC.

4.4.2.2 Clustering Accuracy

Figure 27 and Table 17 presents the clustering accuracy with respect to the number of hidden layers. For all the datasets, the clustering accuracies of both models were increased at the beginning and then started to decrease after some number of hidden layers (Figure 27). For all the datasets, the clustering accuracy of at least one or more deep models (models with two or more layers) showed better accuracy compared to the single-layer models. Therefore, it can be inferred that DEC and RDEC models have better performance compared to direct input data clustering on these datasets. Further, it can be inferred that deep models (models with two or more layers) perform better compared to single-layered neural network models.

For all the datasets, when increasing the network depth, RDEC based model shows less clustering accuracy degradation compared to DEC (see Figure 27). Therefore it can be inferred that, RDEC are less likely to show performance degradation for clustering tasks when increasing the neural network depth. This observation supports our hypothesis of RAE. Figure 28 presents the box and whisker graphs of clustering accuracy distributions of two models when increasing the network depth of AE and RAE. It shows that RDEC has a low variance of clustering accuracy compared to DEC. Further, the mean clustering accuracy value of RDEC was higher compared to DEC.

4.4.2.3 Performance Degradation

Figure 29 shows the performance degradation for deep embedded clustering when increasing the network depth (number of hidden layers from 2 to 12). The performance degradation of clustering was calculated as the difference between the maximum accuracy and minimum accuracy for all tested architectures (models).

According to the result, we observed a clear clustering performance degradation on three datasets (MNIST, HAR and Fashion MNIST), when increasing the network depth of AEs. For embedded representation-based clustering, the RDEC showed significantly lower reduction in performance degradation compared to DEC. Based on the empirical result, it can be seen that the ResNet connection can decrease the clustering performance degradation. Further, RDEC based models showed very steady performance (low variance) when increasing the number of layers (Figure 27). Therefore, the results inferred that adding residual connections result in better deep embedded clustering tasks than the models that don't use residual connections.

4.4.3 Contribution 2 a): Findings, Discussion, and Future Work

This subsection explored Deep Embedded Clustering (DEC) performance degradation when increasing the depth of the deep neural networks. We introduced ResNet architectures into DEC by using Autoencoders with residual connections (RAEs), referred to as RDEC. This modification was made to improve DEC's resistance to performance degradation when using deep Autoencoders (AEs). RDEC was compared with DEC while increasing the network depth of both AE and RAE, on four benchmark datasets. The empirical result showed that RDEC showed up to 56% of less performance degradation compared to DEC. Further, when comparing the variance of the clustering accuracy distribution, RDEC outperformed DEC by showing a lower accuracy variance. Therefore, the empirical results supported our hypothesis and confirmed that RDEC had improved resistance to clustering performance degradation compared to DEC. As future work, we will perform a comparative analysis of RDEC using variants of AEs with residual connections and other widely used clustering methods. Further, the proposed DEC will be applied to the real-world industrial setting and explore the advantages of the proposed method for the specific needs of

CPSs.

4.5 Interpretable Anomaly Detection using ResNet Autoencoders

In the previous section, we extensively studied the advantages of ResNet based Autoencoders for unsupervised feature learning and deep embedded clustering. In this section, we used the presented ResNet architecture for developing an Interpretable Anomaly Detection System (RX-ADS). In this effort, we are focusing on Electric Vehicle (EV) infrastructure as EVs are becoming a primary component in Intelligent Transportation Systems (ITSs) as it decreases fossil fuel consumption and greenhouse gas emissions, reducing negative environmental impact [161]. In recent years, there has been a rapid growth in EV infrastructure, expanding to various areas, including EV manufacturing, charging stations, battery advancements, electric vehicle supply equipment, and other roadside infrastructures [162, 163, 164]. Within EV infrastructure, different communication technologies such as vehicletoVehicle (V2V), Vehicletosensorboard (V2S), vehicletoinfrastructure (V2R), vehicletohuman (V2H), and vehicletointernet (V2I) plays a major role in building resilient operations [165]. Security of these technologies is critical to avoid vulnerabilities such as DoS attacks, false data injections, spoofing and modification [165, 166].

Intrusion Detection Systems (IDSs) are widely used in critical infrastructures such as EV infrastructure [167, 168]. The purpose of IDS is to detect attacks and intruders in communication systems of critical infrastructure, thus avoiding possible catastrophic failures and economic losses. For example, in an EV, attacks can cause break malfunction, engine overheating, control steering issues, and door lock issues, resulting in life-threatening and catastrophic damages [167]. Not only EVs, other infrastructure components such as charging stations are prone to severe advanced persistent threats (APT) such as ransomware and malware [168]. Thus building IDSs

has become a vital component within EV infrastructure.

During the last decade, data-driven machine learning approaches such as Neural Networks (NNs) have been widely used for building IDSs for various critical infrastructure settings [169]. There are two main type of IDSs: Signature based IDS and Anomaly Based IDS [170]. Typically, Anomaly Detection systems (ADSs) have the advantage of detecting both known attacks and unknowns/new attacks/abnormalities in the systems [170, 171]. The idea of ADSs is to learn the normal behavior of a system such that anything outside learned normal behavior is detected as an anomaly. The majority of ADSs are trained using only data coming from normal class/behavior. Therefore, it does not require expensive data labeling process (time-consuming, costly, and requires expertise in data) [27]. Further, ADSs can be developed with an abundance of unlabelled data generated in real-world systems. Out of widely used NN architectures for ADS development, Autoencoders (AE) has gained much attention. The reasons for this include many advantages of AEs such as in-build anomaly detection capability, can be trained with unlabelled data, scalability, feature extraction and dimentionality reduction capability. Therefore, in this work, we are developing a RAE based ADS.

Trustworthy AI is a widely discussed topic when applying NNs for mission-critical infrastructures. Despite the performance benefits of NNs, people hesitate to trust these systems. The main reason for this is the difficulty of understanding the decision-making process of the AI models, making these systems black-box models [4]. It is crucial to address these trust-related issues to build trust between humans and these AI systems. By addressing these issues, the Trustworthy AI research area has emerged. One main component of Trustworthy AI is the Explainability or Interpretability of AI systems (XAI). XAI aims to provide an understanding of black-box models, enabling users to question and challenge the outcomes of AI systems. It pro-

vides many advantages, including justifying outcomes of AI systems, improving trust in AI models, model debugging, and diagnosing [172]. Therefore, this work presents an interpretable ADS developed using RAEs.

This work presents the followings:

1. **Feature Extraction:** Window based feature engineering approach which uses a overlapping sliding window of data frames to extract cyber features.

2. **Anomaly Detection:** ResNet AE based Anomaly Detection System Framework: Framework only used baseline behavior data for learning the normal behavior of the system, thus any deviation from the baseline are tagged as anomaly.

3. **Explainable Interface:** Explanations for anomalous behaviors are generated by using adversarial machine learning. These explanation helps with understanding anomalous behavior, understanding the decision making process of AE, and distinguishing different types of anomalies.

The presented approach was tested on two benchmark datasets which were provided by the Hacking and Countermeasures Research Laboratory. This approach was developed for an ongoing effort with Idaho National Laboratory (INL) for building ADS for an EV charging system (EVCS). Specifically for EVCSs, RX-ADS can provide multiple advantages, including understanding root courses of a given anomaly, allowing domain experts to distinguish different types of anomalies and common anomaly behaviors, and AI model debugging and diagnostics. Further, to the best of the authors' knowledge, no prior research has been attempted to develop Explainable ADSs for CAN data.

Fig. 30. CAN data frame

4.5.1 Background

This section first discusses the Data used for building the presented ADS: CAN data, a widely used communication protocol for in-vehicle communication. Then, we discuss the current work on IDSs developed for CAN data. Finally, we discuss the background of adversarial sample generation and its applications.

4.5.2 CAN Protocol

Controller Area Network (CAN) is the most widely used standard bus protocol for in-vehicle communication. It enables efficient communication between Electronic Control Units (ECUs). It is a broadcast-based protocol that allows multi-master communication, and every node can initiate communication to any other node in the network. Thus, CAN frames do not contain a destination address unlike other protocols. In CAN protocol, each ECU is able to sends messages to the vehicle communication network using data frames [166]. ECUs send frames with their ID number, and the ECU on destination identifies messages by the sender ID included in the frame. The collision of messages and data is avoided by comparing the message ID of the node; the highest priority frame has the lowest ID. CAN is proved to have many advantages, including reducing wiring cost, low weight, low complexity, and operating smoothly in an environment where electromagnetic disturbance factors exist [173].

CAN protocol operates with four main types of frames: the data frame, the remote frame, the error frame, and the overload frame [173]. Most of the communication happens using CAN data frame. The structure of the CAN data frame is

presented in Figure 30, which consists of several common fields that are explained below [173].

- *SOF (Start of Frame)*: indicates the beginning of the frame.
- *Arbitration Field*: is composed of message Id and RTR (Remote Transmission Request) bit. Depending on RTR state the frame will be identified as data or remote frame. During communication frames are prioritized using the ID of the frame.
- *Control Field*: sends the data size
- *Data Field*: the actual data that node wants to send using a data frame, this field can have 0-64 bits.
- *CRC Field*: contains 15-bit checksum that is used for error detection
- *Ack Field*: is used to acknowledge that a valid CAN farme was received by sending a dominant state.
- *EOF (End of Frame)*: indicates the ending of the frame

4.5.3 Anomaly detection using CAN data

Modern vehicles highly rely on ECU communication. Thus CAN has become the standard protocol for facilitating the data exchange between ECUs. While CAN protocol has many advantages, it also has security flaws such as lack of authentication, vulnerability for various attack vectors, and lack of encryption technologies [174]. In the last couple of years, there has been a surge in research addressing security and vulnerabilities of the CAN protocol, most recent work proposes different IDS methods. This subsection discusses existing IDS work on CAN data, specifically focusing on neural network methods and feature extraction techniques.

To develop CAN IDSs, there are four types of feature extraction approaches that have been tested in the literature [174]. First, *frequency or time-based* features where timing between CAN frames and sequencing of CAN frame IDs were used for developing CAN IDSs. In [175], the broadcast time interval for each ID within a window (a discrete, non-overlapping, contiguous set of CAN frames) of CAN frames were calculated. Similar approach was used in [176], where they calculate the signal co-occurrence time of IDs to calculate the absolute-error from expectation for identifying intrusions. Second feature extraction approach is the *Payload-based* approach, where message content bits are directly used for building CAN IDSs [174]. Third approach is the *Signal-based* approach where message content is decoded into a signal before feeding into the IDS. For example in [177], payload bits are encoded before feeding into Neural network architecture for detection intrusions in CAN. Finally, the forth approach is the *Physical side channels* approach, where physical attributes such as voltage and temperature are used to detect intrusions [174, 178]. Other than these four approaches, some IDSs have used rule based methods where characteristic of CAN communication was encoded into rules for detecting intrusions [174].

Neural Network (NN) based CAN IDSs are mainly developed by encoding the characteristics of CAN communication into a set of features and training NN algorithms on these extracted sets of features. The features are extracted to ensure capturing the normal behavior patterns of CAN bus communication apart from abnormalities. For example, in [177], Long Short Term Memory (LSTM) and AE-based unsupervised IDS was developed for detecting intrusions. This architecture consists of a neural network architecture where CAN data from each ID type is presented to its assigned LSTM. The results of LSTM networks are aggregated into AE NN. They have tested their approach on Synthetic CAN data only. A similar approach was used in [179] where they used LSTM for detecting anomalies. However, they have

not aggregated results of multiple LSTMs using AE. In [180], deep NN-based IDS was presented, which used Deep Belief Network (DBN) for initial parameter optimization. This approach is a supervised approach where labeled data was required to build the IDS. Convolutional Neural Network (CNN) based supervised CAN IDS was proposed in [181] where they have tested their system on a real CAN data set. They have directly fed information in CAN frames as features for the training of CNN.

4.5.4 Adversarial Machine Learning

Adversarial samples are generally referred to as malicious input samples designed to fool machine learning algorithms [182]. These samples are typically created by adding a slight modification into real data samples, such that the outcome of a machine learning model for crafted samples will be different than the real sample [169]. Typically, machine learning models are vulnerable to these generated adversarial samples, resulting in unintended or incorrect outcomes. Generally, adversarial samples are generated to maximize the impact on the model while minimizing the ability to identify the adversarial sample apart from a real sample. This ensures by keeping the adversarial sample inside the domain of valid inputs.

Adversarial machine learning has been widely used for exposing vulnerabilities in machine learning [169]. The positive or negative impact of adversarial ML depends on the purpose of the use of these samples. For example, an attacker can use these samples to gain information on a trained ML model, obtain information on the data set the model was trained on, and attack a model. These results in possible privacy invasion, safety failures, data corruption, and model theft [169, 183]. On the other hand, adversarial machine learning also can be used for improving the performance of machine learning models. For example, it can be used to eliminate undefined behaviors of ML models, exploit vulnerabilities, assess model robustness and improve

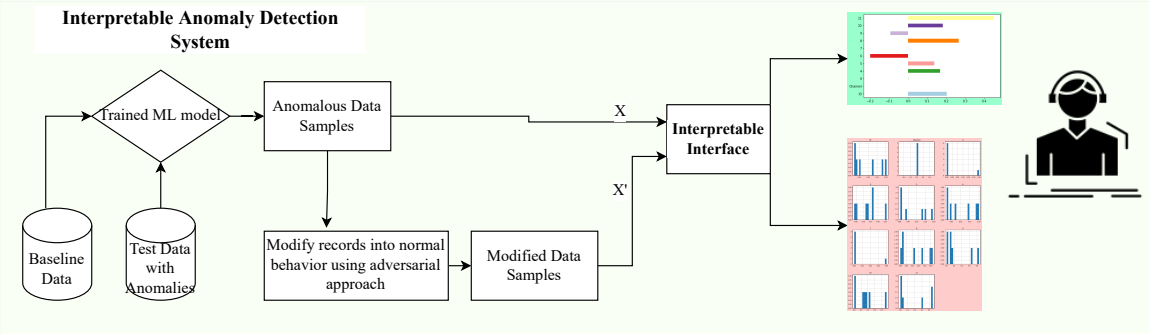


Fig. 31. Interpretable Anomaly Detection System Framework

generalization [169, 183, 184, 185]. This work uses adversarial ML to interpret CAN ADS, which helps with understanding the decision making process of black-box NN models and helps with NN model debugging and diagnostics.

4.5.5 Interpretable Anomaly Detection System

This work develops an interpretable ADS that generates explanations for identified anomalies. Figure 31 illustrates the framework of the proposed approach. First, it trains the ML model using baseline data, i.e., data that represent the normal behavior of the system. Then the trained ML model was tested using various abnormal scenarios. Once it identifies an abnormal sample, these samples are modified using an adversarial approach, i.e., it performs the minimum modification required to change the anomalous records (x') into normal/baseline records (x). The difference between x and x' is used to explain the ADS outputs, illustrating the most relevant features that lead to anomalous behaviors. The individual system components are explained below.

Data-Driven Machine Learning Model: As discussed in the introduction, the Autoencoder (AE) NN model was used to learn the normal/baseline behavior of the system. Specifically, we used deep ResNet AE architecture to avoid possible

performance degradation, and easy parameter optimization [27]. AE has an encoder and decoder, each consisting of multiple hidden layers. Training of the model consists of two stages, the encoding stage, and the decoding stage. The encoding stage transforms the input data into an embedded representation, whereas in the decoding stage, the embedded representation is reproduced back to the original input record (reconstruction). Encoding and decoding functions are non-linear transformation functions. The loss function (J_θ) of the AE model is computed using the difference between the input (x) and the reconstruction (x'). Thus reconstruction error of the AE is calculated as follows,

$$J_\theta = \frac{1}{T}\sum_{i=1}^{T} \|x_i - x_i'\|^2 \tag{4.9}$$

where x_i is the i th input sample, x_i' is the reconstruction for ith input sample, θ denotes the set of parameters of the AE (weights and biases).

During training, AE is trained with data coming from the normal behavior of the system. Therefore, it only learns the possible normal behaviors of the system. When unseen records are presented to the trained AE, the amount of reconstruction error indicates how much the presented data differs from the learned normal behavior. A threshold value is defined to identify possible anomalies. The data records were detected as anomalies if the reconstruction error is higher than the defined thresholds value. Thus, given data record x_i is detected as anomaly ($y = 1$) or normal ($y = 0$) as follows,

$$J_{\theta,i} = \|x_i - x_i'\|^2 \tag{4.10}$$

$$y = \begin{cases} 1 & : \ J_{\theta,i} \geq th \\ 0 & : \ J_{\theta,i} < th \end{cases} \tag{4.11}$$

where $J_{\theta,i}$ is the reconstruction error of ith data record, th denotes the threshold

value of reconstruction error, and y represents predicted label: anomaly or not. The threshold value is optimized based on the training baseline data, i.e., the threshold value should capture the baseline data boundary, capturing the normal behavior fluctuations.

Modifying Anomalous Samples: This work uses adversarial Machine Learning (ML) to understand why a given sample is detected as an anomaly. Explanations of individual data samples are aggregated to understand different scenarios and how they are different from each other by identifying what feature changes are prominent in each scenario. The presented adversarial ML approach aims to understand the decision boundary of normal data and to understand how abnormal scenarios affects the system.

The concept of adversarial sample generation was used to find the minimum modification needed to change the anomalous sample x' into a normal behavior sample. This is achieved by finding an adversarial sample x'' that is detected as normal sample with the given th while minimizing the distance between abnormal sample x' and adversarial/modified sample x''.

$$\min_{x''} \|x' - x''\|^2 \tag{4.12}$$

$$\begin{aligned} s.t : J_{\theta,x''} \leq th \\ x_{min} <= x'' <= x_{max} \end{aligned} \tag{4.13}$$

We constrain the adversarial sample x'' o be inside the bounds (x_{min}, x_{max}). These bounds are defined using the training data, ensuring that the adversarial samples are inside the domain of data distribution.

Interpretable Interface: The presented interpretable interface generates explanations for detected anomalies. For calculating explanations, RX-ADS uses the

identified anomalous samples as references, then uses Eq 4 and 5 to find the adversarial samples with minimum modifications. Explanations are generated under two categories:

- **Explanations for individual Anomaly Samples:** These explanations are generated by calculating the difference between the anomaly sample and the closest adversarial sample ($x' - x''$) and visualizing it using a bar chart. This bar chart shows the deviation of the anomaly sample from what the model learned as normal behavior. Domain experts can analyze these graphs quantitatively to understand the root courses of a given anomaly.

- **Explanations for global anomalous behavior:** Explanations generated for anomalous data records can be aggregated to understand common anomaly behaviors in the system. It allows domain experts to distinguish different types of anomalies and common anomaly behaviors.

4.5.6 Feature Engineering Approach

In this work, we present a window-based feature extraction approach. This is motivated by widely used window-based network flow feature extraction methods in industrial control ADSs [186]. The main goal of this approach is to extract a set of features using the messages contained within a defined sized time window. These features are selected based on the available literature on CAN bus data [187, 166]. These features are extracted to represent the fluctuation in normal behavior compared to attack/abnormal behaviors in the system. The set of extracted features with their description is presented in Table 18. All or some of the features are extracted for each tested dataset.

The sliding window based feature extraction algorithm is presented in Algorithm

Table 18. Feature List

Feature	**Description**
no_of_records	Number of CAN messages
no_of_ids	Number of unique CAN message IDs
no_of_dlc	Number of unique CAN message payload lengths
time_interval	Time interval between messages (minimum, maximum and mean)
no_of_req_msgs	Number of request frames
no_of_res	Number of responses
no_of_lost	Number of lost responses
ratio (min, max, mean)	Number of messages between request frame and response frame
instant_reply_count	Number of instant reply messages
reply_time_interval (min, max, mean)	Time difference between request frame and corresponding response frame
high_priority_count/ 0000	Number of high priority messages
no_XXXX	Number of messages with ID XXXX
payload_P1_XXXX	Mean payload of signal P1 with ID XXXX

I in Table 19. For each dataset, we used a time window and extracted features using the CAN data frames within that window. Overlaps between two windows are kept as half of the window size. In this experiment, window features are extracted using different time window sizes ($winSize$). The extracted features were fed into the AE model for building data-driven ADS.

4.5.7 Experimental setup, Results, and Discussion

The proposed system was tested against two well-studied benchmark datasets and an electric vehicle charging system dataset provided by INL. Both datasets contain CAN bus data representing normal behavior and several abnormal/attack behaviors. This section first describes the training of RX-ADS. Then it discusses RX-ADS results and discussion for each dataset.

Training of RX-ADS: We experimented with different time window sizes and different RAE architectures. Since some of the features, such as payload values, can

Table 19. Proposed feature extraction method

Algorithm I: Extract features
Inputs: Dataset (X), Time window size ($winSize$), Possible set of signal IDs ($IDList$)
Outputs: Window features
$startTime = 0$
$listRecords = [] \leftarrow =$ Initialize a list to store window features
$endTime = Timestampoflastrecordof X \leftarrow =$ Store the last timestamp of the dataset
% Calculating features for each overlapping time window
while $startTime < endTime$ do
$windowMessages \leftarrow$ Extract massages from X where timestamp is within range $startTime - (startTime + winSize)$
$no\_of\_records \leftarrow$ Number of messages in $windowMessages$
$no\_of\_ids \leftarrow$ Number of unique IDs in $windowMessages$
$no\_of\_dlc \leftarrow$ Number of unique data length of messages in $windowMessages$
$time\_interval \leftarrow$ Minimum/Maximum/Mean timestamp differences of messages in $windowMessages$
$no\_of\_req\_msgs \leftarrow$ Number of remote frames in $windowMessages$
$no\_of\_res \leftarrow$ Number of response frames in $windowMessages$
$no\_of\_lost \leftarrow$ Number of lost response frames in $windowMessages$
$ratio \leftarrow$ Number of messages between requests and responses in $windowMessages$ (Minimum, Maximum, Mean)
$reply\_time\_interval \leftarrow$ Minimum/Maximum/Mean timestamp differences of requests and responses in $windowMessages$
$0000 \leftarrow$ Number of high priority messages (ID=0000) in $windowMessages$
$no\_XXXX \leftarrow$ Number of messages with $ID = XXXX$ in $windowMessages$
$payload\_pX\_XXXX \leftarrow$ Mean signal values of payload signal x from messages with $ID = XXXX$ in $windowMessages$
$startTime+ = (winSize/2) \leftarrow$ Calculate start time of next window
end while

result in data sparsity. Therefore, L1 regularized RAE architecture was used. Mean squared error was used as the loss function. A different number of hidden layer sizes were tested. The observed best anomaly detection performance was reported here. We divided the window features of baseline/normal data into two sets (train/test) with a 0.7/0.3 ratio. The data was scaled to the 0-1 range. Once the RAE model is trained with baseline data, the reconstruction errors on train data were used to define an error threshold by keeping 99.99% train data within the defined threshold. The trained RAE's performance and the threshold were tested on test baseline data and abnormal scenario data.

The presented adversarial approach was used to generate explanations for identified abnormal data records. First, the minimum modification needed to correctly detect them as normal records were calculated for the identified abnormal samples (x_0). Then, the explanations are generated by calculating the difference ($x_0 - x'$) between identified abnormal samples (x_0) and the modified/adversarial samples (x'). This difference shows the deviation of the abnormal records from what the model considers as normal behavior of the system. Explanations are generated for different abnormal scenarios separately to compare and distinguish properties of different abnormal behaviors (Ex Dos vs. Fuzzy). The explanation can be generated in two ways. First, they can be generated for a set of abnormal records by calculating the average deviation. These aggregated results help with distinguishing different types of abnormal behaviors. Second, explanations can be generated for an individual abnormal record, presenting how much it deviates from the learned normal behavior.

4.5.7.1 OTIDS dataset

This benchmark CAN dataset was released by Hacking and Countermeasures Research Lab (HCRL) [173]. This dataset contains real CAN data collected from a Kia Soul vehicle in normal behavior as well as during a set of attacks: DoS, Fuzzy, and impersonate. With the release of this dataset, the authors also proposed an Offset Ratio and Time interval-based Intrusion Detection System (OTIDS). Their approach uses offset ratio and time interval of remote frame responses to identify CAN data's Dos, Fuzzy, and Impersonate attacks. In this experiment, we used their baseline, DoS and Fuzzy CAN data to simplify the experiment and compare the explanations. All the features except $payload\_PX\_XXXX$ were extracted for this dataset. This was performed due to the available domain knowledge on this dataset shows that it is possible to identify abnormalities by only using remote request and response-based

Table 20. RX-ADS anomaly detection comparison with recent literature: OTIDS dataset

Approach	**Normal Behavior**	**DoS**	**Fuzzy**
HIDS [187]	100%	100%	100%
OCSVM	**99.77%**	**100%**	**100%**
LOF	**99.32%**	**100%**	**100%**
RX-ADS	**100%**	**100%**	**100%**

features. It has to be noticed that this is the only open dataset with remote frames and responses, such that it is important to experiment on how this information is essential for anomaly identification.

Anomaly Detection System Performance: We experimented with different millisecond time windows: 0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1, 2, 5. We found that when the time window was too small (< 0.02), the performance of the fuzzy detection rate was low. Further, the baseline accuracy was reduced if the time window is too large (> 0.1). The best-observed results were observed for 0.05 milliseconds window size, which is reported in this section. Table 20 shows the detection performance of presented RX-ADS compared to recent state-of-the-art IDSs on OTIDS dataset: Histogram-based approach (HIDS) presented in [187]. We also implement two widely used anomaly detection algorithms: Local Outlier Factor (LOF) and One-Class SVM (OCSVM).

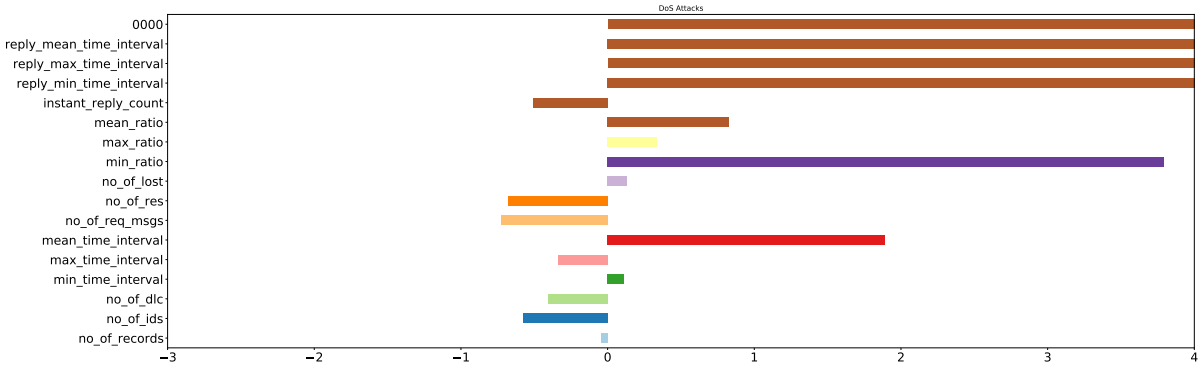
It can be seen that RX-ADS shows comparable performance with the state-of-the-art approach on this dataset.Both HIDS and RX-ADS used window-based feature exaction methods; however, the Histogram-based approach uses a fixed number of CAN frames as a window, whereas RX-ADS uses CAN message within a fixed time window. It has to be noted that the Histogram-based approach uses the K-Nearest

Neighbor (KNN) algorithm for performing multi-class classification. Thus it requires labeled data from all the classes (normal, DoS, and Fuzzy) for training. However, RX-ADS only requires data from normal class for training which provide an advantage when dealing with data unlabeling. To the best of our knowledge, none of the IDSs proposed on the literature for this dataset use an explainable approach. It has to be noted that the goal of our approach is not only to detect anomalies but also to interpret the reasons behind anomalies and interpret the decision-making process of black-box AI models.

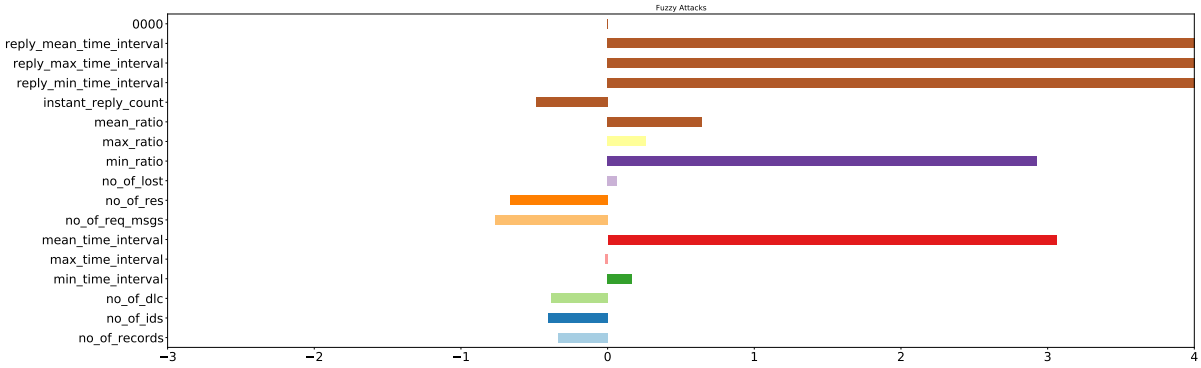
Explanation generation: Figure 32 shows the generated explanations for two types of abnormal/attack behaviors (DoS and Fuzzy). As we discussed before, these deviations of feature values not only explain the behavior of attacks compared to normal behavior but also help with distinguishing different types of attacks. To make the comparison easy, deviations for two types of attacks were presented with the same scale. Explanations for attack behaviors can be naturally interpreted in the following manner:

These samples are detected as anomalies (bot DoS and Fuzzy) due to following reasons (see Fig. 32):

- Higher number of high priority messages with ID 0000
- Higher min/max/mean time interval between remote and response messages
- Higher number of CAN messages between the remote frame and its corresponding response frame (min/max/mean ratio)
- Higher number of lost response messages
- Lower number of instant reply, request, and response messages
- Lower number of unique IDs, number of records, and unique DLC values



(a) DoS



(b) Fuzzy

Fig. 32. Explanations generated for DoS records and Fuzzy records

Related literature on this dataset confirms that the normal state has a very low lost reply rate, a higher number of instant reply rates, and a very low/zero amount of high priority messages. Thus, the identified features on attacks match the domain experts knowledge on this data.

The explanation generated for two types of attacks can be compared against each other to identify distinguishing features between them. The explanations for distinguishing two behaviors can be naturally interpreted in the following manner:

DoS and Fuzzy attacks affects the system differently based on the following observations:

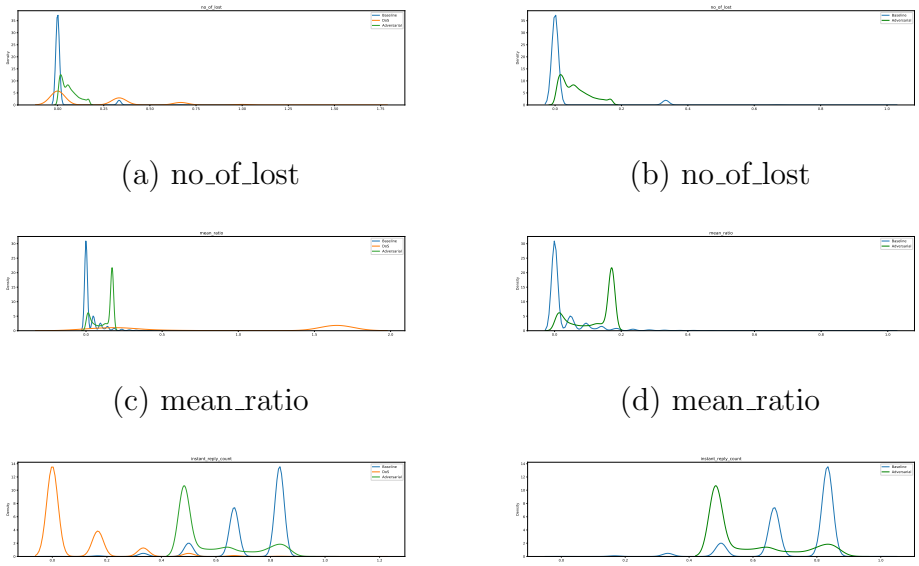
- DoS result in a higher number of high priority messages (0000), whereas Fuzzy does not result in high priority messages with ID 0000
- Min/Max/Mean ratio is higher for DOS due to high priority message communication.
- No of lost response message rate is higher for DoS.

The explanation generated for two types of attacks also can be used to understand the decision-making process of the model. These identified important features allow domain experts to question the model, debug the model, and diagnose the model. These features should be further discussed with domain experts, and possible improvements should be implemented based on the feedback.

Once adversarial samples are generated, feature value distribution of baseline, attack, and adversarial records also give insights into how different features behave under abnormalities. Figure 33 illustrates the feature behavior for selected features under DoS attacks. It can be seen that many of the identified features deviate from the baseline behavior with different magnitudes (Orange line). However, generated adversarial samples (green line) have a much closer feature value distribution than the baseline (blue line). Feature value distribution during Fuzzy attacks is also presented in Figure 34, which also shows similar behavior.

4.5.7.2 Car Hacking dataset

This is the most recent dataset released by the Hacking and Countermeasures Research Lab (HCRL). This dataset contains real CAN data collected from Hyundai YF Sonata. This dataset contains a baseline data file, a data file with DoS attacks,



(a) no_of_lost (b) no_of_lost

(c) mean_ratio (d) mean_ratio

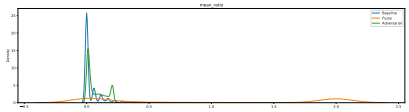
(e) instant_reply_count (f) instant_reply_count

Fig. 33. Feature behavior of DoS data and Adversarial data compared to normal behavior

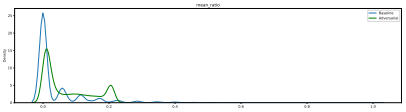
a data file with Fuzzy attacks, and a data file with Spoofing attacks. CAN frame structure is very similar compared to their first released OTIDS dataset. Remote frames indicators are only included in baseline data; thus, this experiment ignores remote information bit from CAN frames. Similar to the previous experiment, we used only baseline, DoS and Fuzzy CAN data to simplify the experiment and compare the explanations. All the features except payload PX_XXXX were extracted for this dataset. This was performed due to the available domain knowledge confirming that it is possible to identify intrusions only using timing information and ID frequencies. This dataset seems to be the most widely used dataset in the CAN IDS literature [174]. Further, initial data analysis indicated large gaps between CAN frames during attacks. These analysis also confirms the previous research work on this data [188, 174]. Hence we trimmed attack datasets before using them for the experimentation. Further, compared to baseline, time intervals between CAN frames are higher during

(a) no_of_lost

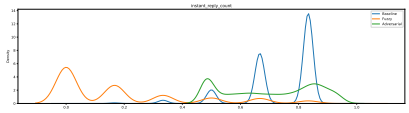
(b) no_of_lost



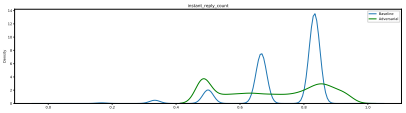
(c) mean_ratio



(d) mean_ratio



(e) instant_reply_count



(f) instant_reply_count

Fig. 34. Feature behavior of DoS data and Adversarial data compared to normal behavior

attack communication.

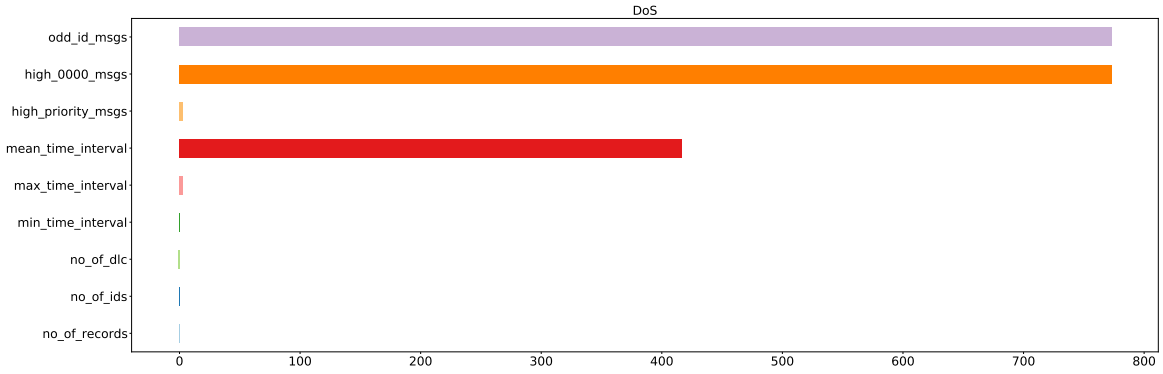
Anomaly Detection System Performance: We experimented with different millisecond time windows: 0.01, 0.02, 0.03, 0.04, 0.05. When looking at frames within the window, we observed some differences between normal and attack communication. When the window size is too small, there are many windows without injected intrusion frames (During attack communication). However, even without any injected frames, the window features of CAN frames are different due to the fact that these windows exist during attacks. Thus, considering these windows as normal windows is inaccurate. If the time window is too large, the number of records generated from window based feature extraction decreases. This results in less number of data records for training. Out of the tested time windows, 0.03 and 0.04 milliseconds gave the best results. The best-observed results were recorded in this section. Table 21 shows the detection performance of presented RX-ADS compared to recent state-of-the-art IDSs

Table 21. RX-ADS anomaly detection comparison with recent literature: Car Hacking Dataset

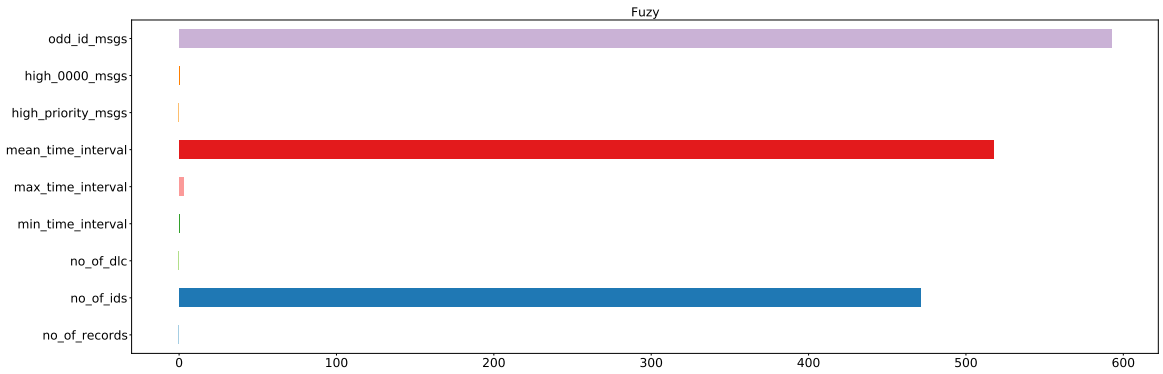
Method	**Data**	**Accuracy**	**Precision**	**Recall**	**F1**
HIDS[187]	*DoS*	97.28	100	96.2	98.06
	Fuzzy	95.17	99.55	94.3	97.18
GIDS [166]	*DoS*	97.9	96.8	99.6	-
	Fuzzy	98.0	97.3	99.5	-
RX-ADS	*Baseline Test*	100	-	-	-
	DoS	99.47	99.6	99.74	99.67
	Fuzzy	99.19	99.39	99.63	99.51

on Car hacking dataset: Histogram-based approach (HIDA) presented in [187] and GIDS presented in [166]. We calculate Accuracy, precision, recall, and F1 scores for comparison purposes with available literature.

It can be seen that the anomaly detection rate of RX-ADS is higher for both DoS and Fuzzy intrusions compared to other approaches. As we discussed before, RX-ADS has an advantage over the HIDS approach as RX-ADS does not require labeled data for training. Further, they have implemented different variants of OCSVM-attack models for each intrusion, whereas RX-ADS only implements one model. RX-ADS can use aggregated explanations for distinguishing DoS from Fuzzy intrusions. GIDS is similar to RX-ADS as they only train on normal data. GIDS requires converting CAN data into image format for training [166]. Thus it is a complex and expensive pre-processing step compared to the simple window-based feature extraction used in RX-ADS. The major advantage of RX-ADS is the model interpretability, making domain experts verify model outcomes and debug and diagnose when necessary.



(a) DoS



(b) Fuzzy

Fig. 35. Explanations generated for DoS records and Fuzzy records for Car Hacking Dataset

Explanation generation: Figure 35 shows the generated global explanations for two types of abnormalities (DoS and Fuzzy). It can be seen that some features highly deviated during abnormal behaviors compared to baseline. Further, there is a clear difference between the two abnormal behaviors. Explanations for attack behaviors can be naturally interpreted in the following manner:

- DoS: Compared to baseline, there are more frames with ID 0000 and odd IDs

during DoS attacks. Further, the mean time interval between frames is higher during DoS attacks.

- Fuzzy: Compared to baseline, a higher number of unique ID frames can be seen during Fuzzy attacks. The mean time interval between frames is also higher during the Fuzzy attacks. The number of odd ID frames is also higher.

We can also compare the behaviors between two attacks for distinguishing unique behaviors of them. DoS and Fuzzy attacks affect the system differently based on the following observations

- During attack behaviors, both Fuzzy and DoS shows a higher number of odd ID frames (frame with IDs that have not been encountered during baseline behavior). However, during DoS, these are mainly are coming from frames with ID 0000, whereas in Fuzzy, these are not high priority frames. These are coming from random IDs which haven't encounter during baseline behavior.

- Mean time interval between frames is higher for both compared to baseline. However, the Fuzzy attack shows the highest mean time interval than DoS.

- Number of unique IDs is very high during fuzzy attacks compared to DoS.

Related literature on this dataset confirms the above-discussed behavior. For example, normal communication has a very low number of high-priority messages. In addition, the Fuzzy attacks result in CAN frames with random IDs which have not been encountered during baseline behaviors. During DoS, the number of high-priority messages with ID 0000 is higher compared to baseline and Fuzzy attacks. Both attacks result in fewer frames within a time window. This happens because, during attacks, it generates high-priority messages or spoofs random messages. These high-priority

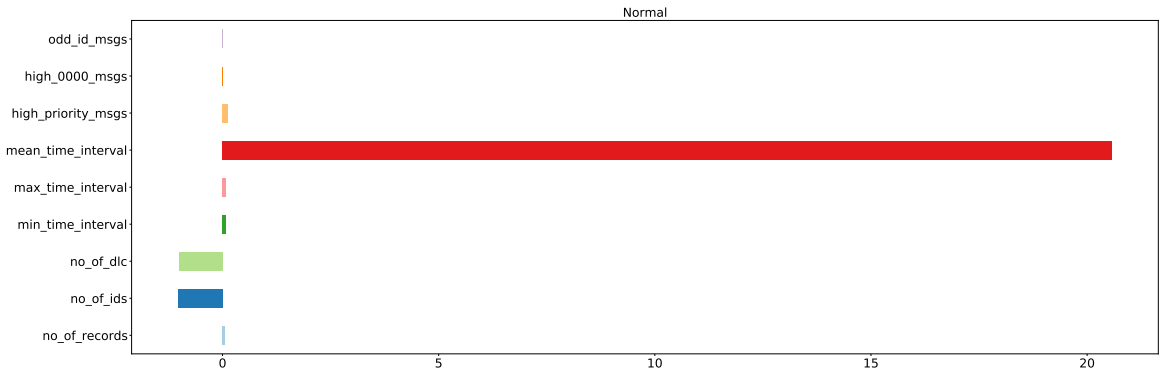


Fig. 36. Normal Communication during Intrusions (DoS)

frames and other attack frames can have multiple effects, such as packet collisions and paralyzing the functions of a vehicle resulting in delays or even suspension of other messages [173]. Therefore, the mean time interval between frames is higher compared to the baseline. These may be due to the CAN frame collisions and paralyzing the functions of a vehicle resulting in delays, or even suspension of other CAN messages [173].

Figure 36 shows the explanations generated for normal communication frames, which are in-between attack behaviors. Even though these samples do not have injected attack frames, the overall communication pattern of these windows was significantly different from the baseline. Thus, the error threshold value was increased to detect these windows as normal windows. This can also be used as a similar filtering method to detect windows without injected frames within attacks (a similar filtering approach was proposed in HIDS). These normal windows during attack communication are expected to be different from baseline communication as attack behaviors result in pre and post-effects in the systems, resulting in deviations from the baseline behavior. These deviations seem to mainly result from higher mean time

intervals between frames. Further, the number of unique IDs and DLC values seems low compared to the baseline. This matches domain experts' knowledge: attack communication results in a latency of CAN frames. These features need to discuss with domain experts.

4.5.8 Contribution 2 b): Findings, Discussion, and Future Work

This subsection presented an approach (RX-ADS) for generating explanations for abnormal behaviors in CAN protocol communication. The ResNet Autoencoder model was used to learn the baseline/normal behavior from the CAN communication data. The reconstruction error threshold of ResNet Autoencoder was used to distinguish abnormal behaviors. The explanation generation method uses an adversarial sample generation approach for identifying the deviation of abnormal behavior from learned baseline behavior. This is achieved by finding the minimum modification required to covert abnormal samples to normal samples. These modifications are used to identify, visualize and explain the relevant feature behaviors for abnormalities. The approach was tested on two widely used benchmarks CAN datasets released by the Hacking and Countermeasures Research Lab.

RX-ADS detected abnormalities in the two benchmark CAN protocol datasets and showed comparable performance compared to the current work on these two datasets. Further, the proposed approach is able to explain the abnormal behaviors of the intrusions matching the expert knowledge. The relevant features found by the presented approach helped with distinguishing between different abnormal behaviors. Experimental results showed that the presented RX-ADS methodology provided insightful and satisfactory explanations for the selected datasets. In future work, the proposed approach will be extended to add physical features for providing more holistic abnormal behavior detection in EV infrastructure.

4.6 Contribution 2: Chapter Summary

This chapter presents the second contribution of the dissertation "Improving and Interpreting Autoencoder Neural Networks". This contribution consisted of two sub-contributions: a) A deep Autoencoder based framework for unsupervised feature learning and deep embedded clustering with improved robustness to network depth, and b) Interpreting Autoencoders for anomaly detection.

Under the *first sub-contribution*, a **ResNet Autoencoder based deep neural network framework** was presented for both unsupervised feature learning based classification and deep embedded clustering. The framework allows making deeper neural networks while not sacrificing its dimensionality reduction based feature learning performance. This framework allows users to design a fewer number of experiments knowing that larger networks will not affect the network performance, especially when dealing with unlabelled data where the optimal network size is challenging to decide. Thus, this framework has the advantage of easy optimization of Deep Autoencoders for unsupervised feature learning and deep embedded clustering tasks. The presented framework was tested with three widely used Deep NN benchmarking datasets. The empirical result showed that the proposed framework showed significantly less performance degradation of the above tasks compared to just using Deep Autoencoders.

Under the *second sub-contribution*, the presented framework was used for developing an **interpretable Anomaly Detection System (RX-ADS)**. This sub contribution first introduced a time window-based cyber feature extraction method for CAN protocol data. The proposed method was tested on two benchmark CAN datasets, and the results showed that the presented RX-ADS methodology provided insightful and satisfactory explanations for the selected datasets. These explanation

helps with understanding anomalous behavior, understanding the decision-making process of AE, and distinguishing different types of anomalies. In future work, the proposed approach will be extended to add physical features for providing more holistic abnormal behavior detection in Electric Vehicle infrastructure.

CHAPTER 5

DISCUSSION AND FUTURE RESEARCH DIRECTIONS

This section discusses the possibility of using existing terms and concepts of XAI in unsupervised machine learning approaches, application areas of eXplainable Unsupervised Machine Learning (XUnML), and possible future research directions.

5.1 Towards XAI in Unsupervised Machine Learning

As we discussed before, existing explainable AI mainly concentrates on supervised algorithms and is composed of many overlapping terms and concepts. Therefore, it is essential to explore how these existing concept of XAI fits the unsupervised learning domain. Here we discuss our view on mapping from existing XAI concepts to the unsupervised domain.

Intrinsic or Extrinsic: The *Intrinsic or Extrinsic* model concepts can be used as it is in the domain of unsupervised learning. For example, unsupervised models like Principle Component Analysis visualized with two or three dimensions can be considered as an Intrinsic interpretable model. Association rule mining techniques can be considered as intrinsic models as they generate rules based on the conditions specified by the user. These conditions can utilize for generating interpretations. Unsupervised models like Mean Clustering go under Extrinsic interpretable models as they need external interpretation models after training to achieve interpretability.

Model Specific and Model Agnostic: The terms *Model Specific and Model Agnostic* can also be used as it is in the unsupervised domain. Small decision trees are one such example of Models Specific interpretable model as the splitting crite-

ria used to explain decision trees are restricted to decision tree algorithms. Some existing agnostic models can be used to explain existing unsupervised clustering approaches. Typically, model agnostic models require labels on data records to achieve interpretability. We can use the cluster labels generated through unsupervised clustering algorithms as dummy labels to existing model agnostic methods. However, this area of research is still at a primitive stage.

Local Interpretability and Global Interpretability: In the unsupervised domain, *Local Interpretability* can be used to explain how a specific data point belongs to a given cluster or how to change the cluster label of a data point by changing its feature values. In auto-regressive modeling, we can present what features of the previous data records lead to predicting future data records. The *Global interpretation* can be defined as generating explanations on why a set of data points belongs to a specific cluster, the important features that decide the similarities between points within a cluster, and the feature value differences between different clusters.

Feature Summary Statistics: Methods used in the supervised domain to present the result of interpretation models can also be mapped to the unsupervised domain. For example, important *feature summary statistics* can be presented using different visualization mediums such as bar charts, tabular format, and linguistic explanations for clustering tasks. Model internal values such as cluster centers of K-Means clustering can be used as a general representation for the data distribution.

5.2 Application areas of XUnML

In this section, we discuss how to use XUnML, specifically interpretable SOMs and interpretable AEs for specific requirements of CPSs.

Trustworthy AI Artificial Intelligence (AI) nowadays influences all the areas of day to day human activities with the state-of-the-art performance in many areas in-

cluding health [55], industry [2], natural language processing [55], space exploration [55] and science [80]. Despite their tremendous benefits, many people hesitate to trust AI-based systems due to the black box behaviors, which makes it difficult to get insight into their internal decision-making process [50]. In order to build trust between AI systems and humans, it is essential that AI systems answer the following questions, *Why did you do that?, Why not something else?, When do you succeed?, When do you fail?, When can I trust you?, How do I correct an error?*. To address these trust related issues, the research area of *Trustworthy AI* was introduced recently. Trustworthy AI aims to strengthen human trust in AI systems, allowing humans and societies to develop, deploy, and use AI systems without fear and doubt. Many respectful academic and non-academic organizations define trustworthyness as combination of diverse research areas which includes *fairness, robustness, explainability, accountability, verifiability, transperency, and sustainability of AI systems*. Therefore, the proposed approaches contribute to one main area of trustworthy AI: the transparency and explainability of AI systems.

Safety and Security: One of the main challenges of CPSs is maintaining safety and security of CPSs. Many modern critical infrastructures have CPSs at their core. Therefore, these systems are highly vulnerable to various attack vectors. Consequently, maintaining the safety and security of CPSs is a primary focus. One approach is to develop data-driven ML-based Anomaly Detection Systems (ADSs) to ensure the security of CPSs. Typically, for developing ADSs, data-driven ML algorithms require collecting data that represents the normal behavior of CPSs. SOMs can be trained with normal data records for this task and identify possible natural clusters (different normal behaviors) and interpretations for each cluster. The domain expert can analyze SOM based explanation to decide whether the collected data represent all status of the normal behavior, what are the dominant natural status in the

system, what features are dominant in each cluster, and the amount of data record distribution among identified normal status are good enough to train ML algorithms. Interpretable AEs also can use to train ADS by using only with normal behaviors. Proposed RX-ADS can use to identify abnormal behaviors, normal behaviors, and explanations for different behaviors. This information allows domain experts to take necessary actions such as attack identifications, root course identification, attack localization, avoiding possible data biases, improving resilience, and reducing the data dimension.

Strategic planning: Today, many large companies need sales strategies targeting different customer groups, which is essential for developing customized sales strategies targeting customer satisfaction and profit increase. Clustering is a widely used approach for discovering customer groups in companies. SOM and DEC can be used to identify cluster groups; then, cluster explanations can determine why a set of customers belongs to a specific cluster. Domain experts can use generated global explanations and evaluate whether the cluster explanations are meaningful or proceed with possible clustering method improvements. These identified meaningful explanations can be used towards building marketing strategies targeting meaningful customer clusters. Further, local explanation allows for analyzing individual customers and provides customer-specific customization.

Generalizability: Lack of generalizability is one main problem in CPS as data-driven ML models mode for one system may not be useful to other CPSs even when both have many similarities. One approach is to retrain and re-purpose models used within one CPSs to another by using pre-trained ML models. SOMs can be used as pre-trained models as SOM can arrange their neuron weights to represent the input data distribution. Therefore, a trained SOM for one task can be used and retrained efficiently for another similar task. Deep AEs are also widely used neural

network models for transfer learning as they can be trained with rich data sources and can be used in target tasks with less amount of data. However, to use pre-trained models effectively, it is essential to evaluate whether the trained models represent meaningful clusters and feature representations. Therefore, presented explainable methods provide a way to evaluate these models and get insights into how they will behave on unseen scenarios.

Real-time Operations: In CPSs, a large amount of high-dimensional data is generated at a rapid speed. For example, in power grids, large volumes of readings come from physical components of the system (voltages, currents) and cyber components (network flow features such as packet rate, payload size, flag). When it comes to high-dimensional data, training ML algorithms can be very costly: generating outcomes from real-time high-dimensional data can be computationally expensive, and storing data can be difficult due to large volumes. Further, it can be impossible to perform real-time processing of these vast volumes of data generated at a rapid speed. In such situations, feature learning is beneficial as it reduces the dimension of input feature space, reducing the number of computations in downstream ML tasks. Further, it reduces the storage requirements for storing data. SOM-based global explanations can be used to identify feature correlations in these situations as it shows how different features values change across clusters. Thus, domain experts can identify and remove highly correlated features, resulting in low dimensional feature spaces. AEs are also used widely as a dimensionality reduction technique, resulting in low computational costs in downstream machine learning tasks. Further, the scalability of AEs is also an advantage as they scale well with the increase of the data due to their data compression capability. Consequently, reducing the storage requirements and computational cost of downstream ML tasks.

Model debugging and diagnostics: The developed AI system should be able

to provide a general understanding of the system, which enables those adversely affected by the system to question and challenge its outcomes. This includes implementing methods that enable users to understand the outcomes of the AI system plainly and easily. The presented interpretable techniques for SOM based clustering and AE based anomaly detection allows users to understand what features these models depend on, some insights into feature importance, and the decision-making process of these algorithms. Therefore, which allows domain experts and machine learning experts to evaluate these models on whether ML models predict the right outcomes for the right reasons, when they can fail, why they fail, and take necessary actions to debug and diagnose ML models.

5.3 Research Directions in Explainable Unsupervised Machine Learning

As we discussed in the background, the traditional concept of UnML was mainly limited to the idea of exploratory data analysis and dimensionality reduction. However, this era of big data and advancements of Deep Neural Networks has given much broader perspective to traditional UnML. Currently, UnML is used in many areas including generative modelling [19], dimentionality reduction [18], feature learning [23], and auto-regressive modelling [21, 20]. This subsection discuss some of these concepts and how explainable AI could help with them.

Transfer Learning: Unsupervised learning can be very successfully used for Transfer Learning [189]. The concept here is to perform representation learning (feature learning or self-taught learning) on a data-rich source to transfer that learned knowledge to an under-resourced target task [190]. In computer vision, this *pre-trained* model concept is widely used to learn generic features from high-resource datasets like ImageNet and then fine tune the models on other image classification tasks. The availability of unlabeled data is abundant. Thus, applying UnML can

greatly help other target tasks such as classification and regression. Incorporating XUnML help human users to understand the UnML models and their outcomes effectively, allowing better utilization of learned knowledge from these rich data sources.

Unsupervised Generative learning: Unsupervised Generative learning is typically used for generating new data samples from a learned representation from unlabeled data [30]. These learned distributions are used to find good representations for large data sets and deal with missing data. Recently, they are also using these models for performing *Exploratory Data Analysis* and *Representation Learning* [191]. Specialty, exploratory data analysis plays a significant role in this era of big data as learning hidden structures from large volumes and revealing inconsistencies in data such as corrupted data, missing data, and redundancies of data [191]. Thus, performing these techniques and communicating the learned knowledge through XUnML to humans is essential.

Qualitative and Quantitative Analysis: It is crucial to notice that *qualitative and quantitative* analysis in unsupervised explainable models can be problematic. The main reason for this is that many existing model evaluation methods require some prior knowledge/data labels. In the unsupervised domain, prior knowledge of data is not available. Further, as described in the previous section, available unsupervised quality metrics are not explainable. Therefore, new evaluation mechanisms should be developed for XUnML methods.

Human study is a classic evaluating mechanism for XAI approaches, where machine learning experts apply the UnML method to a real-world application and provide global/local explanations to domain experts/users using appropriate visualization methods (Application-level evaluation). Domain experts can qualitatively evaluate explanations on whether the learned clusters represent some important similarities (human-level evaluation) or whether the model depends on correct features

for predicted outcomes. Therefore, it is important to focus on researching the effectiveness of human studies in XAI applications.

Model fidelity: Another approach is to use *model fidelity* which evaluates how truthfully the explanation represents the underlying model [192]. Model fidelity of UnML can evaluate by using the information on important subsets of features [193]s. These features can be perturbed, removed, or weighting can be used to get some notion of the truthfulness of features for the decision-making process on a model. For example, model faithfulness of clustering can be evaluated by checking how the cluster label changes when changing the feature values of data samples (quantitative). In unsupervised machine learning, these approaches are not adequately discussed/experimented within the literature. Thus, there is a research gap in using model fidelity not only in UnML but also using them in XUnML.

Human Readiness Levels: Another effective approach for evaluating XUnML is using *Human Readiness Levels* which is a technique that enables evaluating, tracking, and communicating the readiness of a system to human use. The majority of AI systems are only focused on Technology readiness levels, which does not focus enough on the human-ware or users of the system [194]. Therefore, it is crucial to evaluate these generated explanations based on experts' opinions encouraging human involvement for the development stage of XUnML system.

Human-in-the-loop XAI: The concept of *Human-in-the-loop XAI* system is already exist in the literature [195]. However, the existing work is mainly domain/application-specific. Thus, it will be interesting to focus on the Human-in-the-loop XUnML system, exploring: how to effectively communicate the knowledge extracted from unlabeled data to domain experts using XUnML, and how to integrate domain experts knowledge back into XUnML systems.

Uncertainty quantification is also can be used to improve the explanation gen-

eration approaches. They can be used to provide additional security and minimize the risk of wrongful explanations generated from an AI system for many possible reasons, including unseen data, data drifts, data biases, model biases, misleading/noisy data, and possible attacks. This gives additional assurance to users on an XAI system.

Bias is a frequently addressed topic in the machine learning community. Bias in machine learning can exist in many shapes and forms, such as data biases (ex: measurement biases, representation biases, data processing biases), algorithmic biases (ex: algorithmic design choices related biases), and user biases (ex: user interaction biases and evaluation biases). Interpretable unsupervised machine learning can be used to address some forms of Bias in ML models. Unsupervised models that perform clustering and dimensionality reduction can be used to eliminate data biases, revealing what such data actually represents (data clusters), how clusters are different, and how clusters are correlated/overlapped. Thus, using the global and local explanations, users can understand which features the model depends on, feature behaviors on different clusters, and what features drive the model decisions. This information allows machine learning experts and domain experts to understand what the training data represents, helping them preprocess data appropriately to improve the data quality, hence reducing data biases.

Benchmark datasets with domain knowledge: Current research community has access to millions of benchmark datasets representing different domains. The majority of these datasets contain data labels. However, these datasets do not have comprehensive domain knowledge descriptions included with them. These domain knowledge descriptions can include information such as system details, common properties of each class, distinguishing properties of classes, and information from domain experts on possible system behaviors and how they can be represented within data. Therefore, it limits the research advancements in XAI, as once the explanations

are generated, one of the very accurate ways of evaluating these XAI outcomes is by comparing with the domain knowledge of the dataset.

CHAPTER 6

CONCLUSIONS

This chapter summarizes the objectives, contributions, conclusions, and possible future research directions of this dissertation.

Real-world systems generated a massive amount of unlabeled data at a rapid phase, limiting the use of supervised Machine Learning (ML) algorithms. Further, even with the tremendous success of ML models, their black-box nature makes humans not trust ML models. Therefore, the objective of this dissertation is to improve and interpret unsupervised neural networks.

In this dissertation, improving unsupervised neural networks refers to improving the feature learning capability of unsupervised neural networks, whereas interpreting unsupervised neural networks refers to developing techniques to explain the underline decision-making process of these algorithms effectively. This dissertation focus on two unsupervised learning algorithms, Self Organizing Neural Network and Autoencoder Neural Network. Thus, this dissertation provided two main contributions, each with two sub-contributions;

1. *Contribution 1:* Improving and Interpreting Self Organizing Neural Network (SOM)

    (a) A novel unsupervised Self Organizing Neural Network architecture for learning features of different resolutions in parallel layers: improve classification accuracy and generalizability

    (b) A novel technique for interpreting Self Organizing Neural Network algo-

rithm for unsupervised clustering

2. *Contribution 2:* Improving and Interpreting Autoencoder (AE) Neural Networks

   (a) A deep Autoencoder Neural Network based framework for unsupervised feature learning and deep embedded clustering: improve robustness to network depth

   (b) A novel technique for interpreting deep Autoencoder based framework for anomaly detection

Contribution 1: Improving and Interpreting Self Organizing Neural Network (SOM)

Under the **contribution 1 a)**, we developed a *novel Enhanced Deep Self-organizing Map (E-DSOM)* architecture that can perform unsupervised learning of features of different resolutions in parallel layers. The proposed DSOM architecture enhances the performance of existing Deep SOM (DSOM) architecture in two ways: 1) the learning algorithm is completely unsupervised, and 2) the architecture learns features of different resolutions in parallel in a single hidden layer. E-DSOM was tested on three datasets and compared with DSOM. E-DSOM outperformed DSOM in terms of classification accuracy with improvements of up to 15%. Generalization capability was tested by adding noise to test data. E-DSOM outperformed DSOM at all noise levels (barring one instance with comparable results), evidencing better generalization capability. E-DSOM also showed improved computational time by gaining the same or better classification accuracy with a shallower model. E-DSOM showed training time improvements up to 19%. Therefore, the empirical results evidenced that the presented architecture showed improved performance compared to the DSOM architectures in terms of 1) classification accuracy, 2) generalization capability, and 3)

training time. Additionally, E-DSOM architecture was compared to other unsupervised algorithms. Empirical results show that E-DSOM algorithms are competitive and a viable option for unsupervised learning.

Under the **contribution 1 b)**, we presented a *novel model-specific explainable method for the SOM based clustering*. Through feature value perturbation, we evaluated the model fidelity and showed that the proposed approach identifies the most important feature used by the decision-making process of SOMs. We showed that the changing of features values of important features affects the cluster label outcomes of SOMs. We presented the proposed approach as a strong candidate as an eXplainable Unsupervised Machine Learning (XUnML) method by comparing it with current XUnML methods in terms of model-specific features, limitations, and usability.

As future work for the contribution 1: The presented interpretable SOM-based clustering approach will be extended to the proposed E-DSOM architecture. Further, the proposed approaches will be evaluated through a human study and human readiness level while exploring the proposed approach's capability for cyber-physical system applications.

Contribution 2: Improving and Interpreting Autoencoder Neural Network (SOM)

The second contribution consists of sub-contributions: a) A deep Autoencoder based framework for unsupervised feature learning and Deep Embedded Clustering (DEC), and b) Interpreting deep Autoencoders for anomaly detection.

Under the **contribution 2 a)**, we introduce a deep Autoencoder neural network framework for unsupervised feature learning and deep embedded clustering. This framework consisting of ResNet Autoencoder (RAE) that allows for making deeper neural networks while not sacrificing its dimensionality reduction-based feature learning performance and deep clustering performance. In this way, we improve resistance

to performance degradation compared to standard Autoencoders (AEs) for feature learning as well as deep clustering. The performance of RAE on learning deep embedded representations was evaluated on a classification task (using the K Nearest Neighbors algorithm). RAE was compared against AE while increasing the number of hidden layers on three benchmark datasets. We demonstrated that RAE showed the highest accuracy on all three datasets. Both RAE and AE showed performance degradation when increasing the network depth. However, RAE based classification only showed 0.86% to 2.68% performance degradation, which is significantly lower than the performance degradation shown by standard AE (33.38% - 65.46%). Further, the classification accuracy distribution showed that RAE models perform better in terms of mean accuracy and accuracy variance (low variance), making them more suitable for deep embedded classification tasks than AE. Further, we compared RAEs with widely used dimensionality reduction methods and showed that RAE outperforms on all experimented datasets.

Under DEC, we introduced ResNet architectures into DEC by using Autoencoders with residual connections, referred to as RDEC. This modification was made to improve DEC's resistance to performance degradation when using deep Autoencoders (AEs). RDEC was compared with DEC while increasing the network depth of both AE and RAE, on the same three benchmark datasets used for RAE based classification. The empirical result showed that RDEC showed up to 56% of less performance degradation compared to DEC. Further, when comparing the variance of the clustering accuracy distribution, RDEC outperformed DEC by showing a lower accuracy variance. The above empirical results confirmed that RAE reduces performance degradation of deep embedded representation based classification and DEC. Further, this framework allows users to design a fever number of experiments knowing that larger networks will not affect the network performance. This is a major advantage

with unlabelled data where the optimal network size is challenging to decide.

Under the **contribution 2 b)**, the presented framework was also used for developing an *explainable Anomaly Detection System - ResNet Autoencoder based eXplainable Anomaly Detection System (RX-ADS)*. RX-ADS was developed to detect anomalies in the CAN bus protocol, which is the standard communication protocol for in-vehicle communication. The ResNet Autoencoder framework was used to learn the baseline/normal behavior from the data. The reconstruction error threshold was used to distinguish abnormal behaviors. The explanation generation method uses an adversarial sample generation approach for identifying the deviation of abnormal behavior from learned baseline behavior. This is achieved by finding the minimum modification required to covert abnormal samples to normal samples. These modifications are used to identify, visualize and explain the relevant feature behaviors for abnormalities. The approach was tested on two widely used benchmarks CAN datasets released by the Hacking and Countermeasures Research Lab: OTIDS and Car Hacking.

RX-ADS detected abnormalities in the two tested datasets and showed comparable performance compared to the current work on these two datasets. Further, the proposed approach is able to explain the abnormal behaviors of the intrusions matching the expert knowledge. The relevant features found by the presented approach helped with distinguishing between different abnormal behaviors. Experimental results showed that the presented RX-ADS methodology provided insightful and satisfactory explanations for the selected datasets. This work was funded by Idaho National Laboratory (INL). The presented RX-ADS is currently being transitioned to testbed at Idaho National Laboratory.

As future work for contribution 2, we will perform a comparative analysis of feature learning and RDEC using variants of AEs with residual connections and

other widely used clustering methods. Further, the proposed RX-ADS approach will be extended to add physical features for providing more holistic abnormal behavior detection in EVCS communication.

Appendix A

ABBREVIATIONS

AI	Artificial Intelligence
ML	Machine Learning
SML	Supervised Machine Learning
UnML	Unsupervised Machine Learning
XAI	Explainable/interpretable Machine Learning
ANN	Artificial Neural Network
DNN	Deep Neural Network
CPS	Cyber Physical System
AE	Autoencoder
CNN	Convolutional Neural Network
RAE	ResNet Autoencoder
C-RAE	Convolutional ResNet Autoencoder
DEC	Deep Embedded Clustering
SOM	Self Organizing Neural Network
DSOM	Deep Self Organizing Map
EDSOM	Enhanced Deep Self Organizing Neural Network
RX-ADS	ResNet eXplainable Anomaly Detection System
IDS	Intrusion Detection System
BMU	Best Matching Unit
VDM	Visual Data Mining
XUnML	Interpretable Unsupervised Machine Learning

Appendix B

LIST OF PUBLICATIONS BY THE AUTHOR

This appendix presents a list of the author's published journal and peer-reviewed conference publications.

B.1 Journal Publications

- **C. S. Wickramasinghe**, D. L. Marino, and M. Manic, "RX-ADS: Interpretable Anomaly Detection method using Adversarial ML for Electric Vehicle CAN data," 2022. (Under review in IEEE Transactions on Intelligent Transportation Systems)

- **C. S. Wickramasinghe**, K. Amarasinghe, D. L. Marino, C. Rieger and M. Manic, "Explainable Unsupervised Machine Learning for Cyber-Physical Systems", in IEEE Access, vol. 9, pp. 131824-131843, 2021, doi: 10.1109/ACCESS.2021.3112397.

- **C. S. Wickramasinghe**, D. L. Marino, and M. Manic, "ResNet Autoencoders for Unsupervised Feature Learning From High-Dimensional Data: Deep Models Resistant to Performance Degradation", in IEEE Access, vol. 9, pp. 40511-40520, 2021, DOI: 10.1109/ACCESS.2021.3064819.

- **Chathurika S. Wickramasinghe**, Kasun Amarasinghe, Milos Manic, "Deep Self-Organizing Maps for Unsupervised Image Classification", in IEEE Transactions on Industrial Informatics , vol. 15, no. 11, pp. 5837-5845, Nov. 2019, DOI: doi: 10.1109/TII.2019.2906083, 2018-2019.

- D. L. Marino, **C. S. Wickramasinghe**, B. Tsouvalas, C. Rieger and M. Manic, "Data-Driven Correlation of Cyber and Physical Anomalies for Holistic System Health Monitoring", in IEEE Access, vol. 9, pp. 163138-163150, 2021, doi: 10.1109/ACCESS.2021.3131274.

- D. L. Marino, **C. S. Wickramasinghe**, V. K. Singh, J. Gentle, C. Rieger and M. Manic, "The Virtualized Cyber-Physical Testbed for Machine Learning Anomaly Detection: A Wind Powered Grid Case Study", in IEEE Access, vol. 9, pp. 159475-159494, 2021, doi: 10.1109/ACCESS.2021.3127169.

- Vaagensmith B, Kumar Singh V, Ivans R, Marino DL, **Wickramasinghe CS**, Lehmer J, Phillips T, Rieger C, Manic M., "Review of Design Elements within Power Infrastructure Cyber-Physical Test Beds as Threat Analysis Environments", in Energies 2021, DOI: https://doi.org/10.3390/en14051409

- Deepak Kumbhare, Viktoras Palys, Jamie Toms, **Chathurika Wickramasinghe**, Kasun Amarasinghe, Milos Manic, Evan Hughes, Kathryn Lois Holloway, "Nucleus Basalis of Meynert stimulation for dementia: Theoretical and Technical considerations", in Frontiers in Neuroscience 2018, vol. 12, pp.614. DOI: doi: 10.3389/fnins.2018.00614

B.2 Conference Publications

- C. Wickramasinghe, D. Marino, and M. Manic, "Deep Embedded Clustering with ResNets", in Proc. 14th International Conference on Human System Interaction, IEEE HSI 2021, Poland, July 8-10. 2021.

- C. Wickramasinghe, D. Marino, J. Grandio, and M. Manic, "Trustworthy AI Development Guidelines for Human System Interaction", in Proc. 13th Inter-

national Conference on Human System Interaction, IEEE HSI 2020, Tokyo, Japan, June 6-8. 2020.

- C. Wickramasinghe, K. Amarasinghe, D. Marino, and M. Manic, "Deep Self-Organizing Maps for Visual Data Mining", in Proc. 11th International Conference on Human System Interaction, IEEE HSI 2018, Gdansk, Poland, July 04-06, 2018. , DOI: 10.1109/HSI.2018.8430845.

- C. Wickramasinghe, K. Amarasinghe, M. Manic, "Parallalizable Deep Self-Organizing Maps for Image Classification" , in Proc. 2017 IEEE Symposium Series on Computational Intelligence, IEEE SSCI 2017, Honolulu, Hawaii, USA, Nov, 27- Dec 1, 2017. sDOI: 10.1109/SSCI.2017.8285443.

- D. Marino, C. Wickramasinghe, M. Manic, "An Adversarial Approach for Explainable AI in Intrusion Detection Systems", in Proc. 44th Annual Conference of the IEEE Industrial Electronics Society, IECON 2018, Washington DC, USA, Oct. 21-23, 2018. DOI: 10.1109/IECON.2018.8591457

- C. Wickramasinghe, K. Amarasinghe, D. Marino, Z. Spielman, I. Pray, D. Gertman, and M. Manic, "Intelligent Driver System for Improving Fuel Efficiency in Vehicle Fleets", in Proc. 12th International Conference on Human System Interaction, IEEE HSI 2019, Richmond VA, USA, June 25-27, 2019.

- C. Wickramasinghe, D. Marino, F. Yucel, E. Bulut, and M. Manic, "Data-Driven Hourly Taxi Drop-offs Prediction using TLC Trip Record Data", in Proc. 12th International Conference on Human System Interaction, IEEE HSI 2019, Richmond VA, USA, June 25-27, 2019.

- C. Wickramasinghe, D. Marino, K. Amarasinghe, M. Manic, "Generalization of Deep Learning For Cyber-Physical System Security: A Survey", in Proc. 44th

Annual Conference of the IEEE Industrial Electronics Society, IECON 2018, Washington DC, USA, Oct. 21-23, 2018.

- K. Amarasinghe, C. Wickramasinghe, D. Marino, C.Rieger, M. Manic, "Framework for Data-Driven Health Monitoring of Cyber-Physical Systems", in IEEE Resilience Week (RW) 2018, Denver, CO, USA, Aug 20-23, 2018.

- M. Stuart, C. Wickramasinghe, D. Marino, D. Kumbhare, K. Holloway, M. Manic, "Machine Learning for Deep Brain Stimulation Efficacy using Dense Array EEG", in Proc. 12th International Conference on Human System Interaction, IEEE HSI 2019, Richmond VA, USA, June 25-27, 2019.

- D. Marino, C. Wickramasinghe, C. Rieger, M. Manic, "Data-driven Stochastic Anomaly Detection on smart-Grid communications using Mixture PoissonD istributions", in Proc. 45th Annual Conference of the IEEE Industrial Electronics Society, IECON 2019, Lisbon, Portugal, Oct. 14-17, 2019.

- Daniel L. Marino, Chathurika S. Wickramasinghe, Kasun Amarasinghe, Hari Challa, Philip Richardson, Ananth A. Jillepalli, Brian K. Johnson, Craig Rieger, Milos Manic, "Cyber and Physical Anomaly Detection in Smart-Grids", in Proc. of the IEEE Resilience Week (RW) 2019, San Antonio, TX, USA, Nov 4-7, 2019.

- D. Marino, J. Grandio, C. Wickramasinghe, K. Schroeder, K. Bourne, A.V. Filippas, and M. Manic, "AI Augmentation for Trustworthy AI: Augmented Robot Teleportation", in Proc. 13th International Conference on Human System Interaction, IEEE HSI 2020, Tokyo, Japan, June 6-8. 2020.

REFERENCES

[1] By: IBM Cloud Education. *What is Machine Learning?* URL: https://www.ibm.com/cloud/learn/machine-learning.

[2] ©[2022] IEEE. Reprinted with permission from Chathurika S. Wickramasinghe et al. "Generalization of Deep Learning for Cyber-Physical System Security: A Survey". In: *IECON 2018 - 44th Annual Conference of the IEEE Industrial Electronics Society*. 2018, pp. 745–751. DOI: 10.1109/IECON.2018.8591773.

[3] Martin Längkvist, Lars Karlsson, and Amy Loutfi. "A review of unsupervised feature learning and deep learning for time-series modeling". In: *Pattern Recognition Letters* 42 (2014), pp. 11–24.

[4] ©[2022] IEEE. Reprinted with permission from Chathurika S. Wickramasinghe et al. "Trustworthy AI Development Guidelines for Human System Interaction". In: *2020 13th International Conference on Human System Interaction (HSI)*. 2020, pp. 130–136. DOI: 10.1109/HSI49210.2020.9142644.

[5] *Explainable Artificial Intelligence (XAI)*. URL: https://www.darpa.mil/program/explainable-artificial-intelligence.

[6] Sushant Jain et al. "Exploiting Mobility for Energy Efficient Data Collection in Wireless Sensor Networks". In: *MONET* 11 (June 2006), pp. 327–339. DOI: 10.1007/s11036-006-5186-9.

[7] J. Shi et al. "A survey of Cyber-Physical Systems". In: *2011 International Conference on Wireless Communications and Signal Processing (WCSP)*. 2011, pp. 1–6. DOI: 10.1109/WCSP.2011.6096958.

[8] Y. Zhang et al. "Optimal Adaptive System Health Monitoring and Diagnosis for Resource Constrained Cyber-Physical Systems". In: *2009 20th International Symposium on Software Reliability Engineering*. 2009, pp. 51–60. DOI: 10.1109/ISSRE.2009.21.

[9] Guangyu Wu, Jian Sun, and Jie Chen. "A survey on the security of cyber-physical systems". In: *Control Theory and Technology* 14 (Feb. 2016), pp. 2–10. DOI: 10.1007/s11768-016-5123-9.

[10] R. Rajkumar et al. "Cyber-physical systems: The next computing revolution". In: *Design Automation Conference*. 2010, pp. 731–736. DOI: 10.1145/1837274.1837461.

[11] S. Vashi et al. "Internet of Things (IoT): A vision, architectural elements, and security issues". In: *2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*. 2017, pp. 492–496. DOI: 10.1109/I-SMAC.2017.8058399.

[12] Anonymous. *Cyber-Physical Systems*. Oct. 2020. URL: https://ec.europa.eu/digital-single-market/en/cyber-physical-systems.

[13] *Cyber-Physical Systems (CPS)*. URL: https://www.nsf.gov/pubs/2021/nsf21551/nsf21551.htm.

[14] Kristy.thompson@nist.gov. *Cyber-Physical Systems*. Nov. 2019. URL: https://www.nist.gov/el/cyber-physical-systems.

[15] Aishwarya Mujumdar and V. Vaidehi. "Diabetes Prediction using Machine Learning Algorithms". In: *Procedia Computer Science* 165 (Jan. 2019), pp. 292–299. DOI: 10.1016/j.procs.2020.01.047.

[16] Amanpreet Singh, Narina Thakur, and Aakanksha Sharma. "A review of supervised machine learning algorithms". In: *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*. 2016, pp. 1310–1315.

[17] Tammy Jiang, Jaimie L. Gradus, and Anthony J. Rosellini. "Supervised Machine Learning: A Brief Primer". In: *Behavior Therapy* 51.5 (2020), pp. 675–687. ISSN: 0005-7894. DOI: https://doi.org/10.1016/j.beth.2020.05.002. URL: https://www.sciencedirect.com/science/article/pii/S0005789420300678.

[18] A.A. Mohamed. "An effective dimension reduction algorithm for clustering Arabic text". In: *Egyptian Informatics Journal* (2019).

[19] Ian Goodfellow et al. "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems 27*. Ed. by Z. Ghahramani et al. Curran Associates, Inc., 2014, pp. 2672–2680.

[20] Ian J. Goodfellow. "NIPS 2016 Tutorial: Generative Adversarial Networks". In: *CoRR* abs/1701.00160 (2017).

[21] Pauline Luc et al. "Predicting Deeper into the Future of Semantic Segmentation". In: Oct. 2017, pp. 648–657.

[22] Guang-yong Chen, Min Gan, and Guo-long Chen. "Generalized exponential autoregressive models for nonlinear time series: Stationarity, estimation and applications". In: *Information Sciences* 438 (2018), pp. 46–57.

[23] Elie Aljalbout et al. "Clustering with Deep Learning: Taxonomy and New Methods". In: *arXiv e-prints*, arXiv:1801.07648 (Jan. 2018), arXiv:1801.07648.

[24] Memoona Khanam et al. "A Survey on Unsupervised Machine Learning Algorithms for Automation, Classification and Maintenance". In: *International Journal of Computer Applications* 119 (June 2015), pp. 34–39. DOI: 10.5120/21131-4058.

[25] Steffen Schneider et al. *wav2vec: Unsupervised Pre-training for Speech Recognition*. 2019. arXiv: 1904.05862 [cs.CL].

[26] Feng Wang et al. *Unsupervised Representation Learning by InvariancePropagation*. 2020. arXiv: 2010.11694 [cs.CV].

[27] ©[2022] IEEE. Reprinted with permission from C. S. Wickramasinghe, D. L. Marino, and M. Manic. "ResNet Autoencoders for Unsupervised Feature Learning From High-Dimensional Data: Deep Models Resistant to Performance Degradation". In: *IEEE Access* 9 (2021), pp. 40511–40520. DOI: 10.1109/ACCESS.2021.3064819.

[28] Piotr Bojanowski and Armand Joulin. "Unsupervised Learning by Predicting Noise". In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, June 2017, pp. 517–526. URL: http://proceedings.mlr.press/v70/bojanowski17a.html.

[29] Krzysztof J. Cios et al. "Unsupervised Learning: Association Rules". In: *Data Mining: A Knowledge Discovery Approach*. Boston, MA: Springer US, 2007, pp. 289–306. ISBN: 978-0-387-36795-8. DOI: 10.1007/978-0-387-36795-8_10. URL: https://doi.org/10.1007/978-0-387-36795-8_10.

[30] Zhao-Yu Han et al. "Unsupervised Generative Modeling Using Matrix Product States". In: *Phys. Rev. X* 8 (3 July 2018), p. 031012. DOI: 10.1103/

PhysRevX . 8 . 031012. URL: https : / / link . aps . org / doi / 10 . 1103 / PhysRevX.8.031012.

[31] Anthony Zador. "A critique of pure learning and what artificial neural networks can learn from animal brains". In: *Nature Communications* 10 (Dec. 2019). DOI: 10.1038/s41467-019-11786-6.

[32] Maryam M. Najafabadi et al. "Deep learning applications and challenges in big data analytics". In: *Journal of Big Data* 2.1 (Feb. 2015), p. 1. ISSN: 2196-1115. DOI: 10.1186/s40537-014-0007-7. URL: https://doi.org/10.1186/s40537-014-0007-7.

[33] Yoshua Bengio et al. "Greedy Layer-wise Training of Deep Networks". In: *Proceedings of the 19th International Conference on Neural Information Processing Systems*. NIPS'06. Canada: MIT Press, 2006, pp. 153–160. URL: http://dl.acm.org/citation.cfm?id=2976456.2976476.

[34] Ian Goodfellow et al. "Measuring Invariances in Deep Networks". In: *Advances in Neural Information Processing Systems 22*. Ed. by Y. Bengio et al. Curran Associates, Inc., 2009, pp. 646–654. URL: http://papers.nips.cc/paper/3790-measuring-invariances-in-deep-networks.pdf.

[35] Jürgen Schmidhuber. "Deep Learning in neural networks: An overview". In: *Neural Networks* 61 (2015), pp. 85–117. ISSN: 18792782. DOI: 10.1016/j.neunet.2014.09.003. arXiv: 1404.7828. URL: http://dx.doi.org/10.1016/j.neunet.2014.09.003.

[36] Mohd Zaki Mas'ud et al. "Analysis of Features Selection and Machine Learning Classifier in Android Malware Detection". In: *2014 International Conference on Information Science & Applications (ICISA)* (2014), pp. 1–5. ISSN:

2162-9048. DOI: 10.1109/ICISA.2014.6847364. URL: http://ieeexplore.ieee.org/document/6847364/.

[37] Basant Subba, Santosh Biswas, and Sushanta Karmakar. "A Neural Network based system for Intrusion Detection and attack classification". In: *2016 Twenty Second National Conference on Communication (NCC)* (2016), pp. 1–6. DOI: 10.1109/NCC.2016.7561088. URL: http://ieeexplore.ieee.org/document/7561088/.

[38] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9.8 (1997), pp. 1735–1780. ISSN: 08997667. DOI: 10.1162/neco.1997.9.8.1735. arXiv: 1206.2944.

[39] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[40] Y LeCun and Y Bengio. "Convolutional networks for images, speech, and time series". In: *The handbook of brain theory and neural networks* 3361.April 2016 (1995), pp. 255–258. ISSN: 1098-7576. DOI: 10.1109/IJCNN.2004.1381049. arXiv: arXiv:1011.1669v3. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.32.9297%7B%5C&%7Drep=rep1%7B%5C&%7Dtype=pdf.

[41] Wei Wang, Mengxue Zhao, and Jigang Wang. "Effective android malware detection with a hybrid model based on deep autoencoder and convolutional neural network". In: *Journal of Ambient Intelligence and Humanized Computing* 0.0 (123), p. 0. ISSN: 1868-5145. DOI: 10.1007/s12652-018-0803-6. URL: https://doi.org/10.1007/s12652-018-0803-6.

[42] Bojan Kolosnjaji et al. “Deep Learning for Classification of Malware System Call Sequences”. In: 9992 (2016), pp. 137–149. DOI: 10.1007/978-3-319-50127-7. URL: http://link.springer.com/10.1007/978-3-319-50127-7.

[43] Weibo Liu et al. “A survey of deep neural network architectures and their applications”. In: *Neurocomputing* 234.October 2016 (2017), pp. 11–26. ISSN: 18728286. DOI: 10.1016/j.neucom.2016.12.038. URL: http://dx.doi.org/10.1016/j.neucom.2016.12.038.

[44] Weibo Liu et al. “A survey of deep neural network architectures and their applications”. In: *Neurocomputing* 234 (2017), pp. 11–26. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2016.12.038. URL: http://www.sciencedirect.com/science/article/pii/S0925231216315533.

[45] Ni Gao et al. “An Intrusion Detection Model Based on Deep Belief Networks”. In: *2014 Second International Conference on Advanced Cloud and Big Data* (2014), pp. 247–252. ISSN: 2329-6267. DOI: 10.1109/CBD.2014.41. arXiv: arXiv:1011.1669v3. URL: http://ieeexplore.ieee.org/document/7176101/.

[46] Yuancheng Li, Rong Ma, and Runhai Jiao. “A hybrid malicious code detection method based on deep learning”. In: *International Journal of Security and its Applications* 9.5 (2015), pp. 205–216. ISSN: 17389976. DOI: 10.14257/ijsia.2015.9.5.21.

[47] David Gunning and David Aha. “DARPA’s Explainable Artificial Intelligence (XAI) Program”. In: *AI Magazine* 40.2 (June 2019), pp. 44–58. DOI: 10.1609/aimag.v40i2.2850. URL: https://ojs.aaai.org/index.php/aimagazine/article/view/2850.

[48] R. Roscher et al. “Explainable Machine Learning for Scientific Insights and Discoveries”. In: *IEEE Access* 8 (2020), pp. 42200–42216. DOI: 10.1109/ACCESS.2020.2976199.

[49] Alejandro Barredo Arrieta et al. “Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI”. In: *Information Fusion* 58 (2020), pp. 82–115. ISSN: 1566-2535. DOI: https://doi.org/10.1016/j.inffus.2019.12.012. URL: https://www.sciencedirect.com/science/article/pii/S1566253519308103.

[50] A. Adadi and M. Berrada. “Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI)”. In: *IEEE Access* 6 (2018), pp. 52138–52160. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2018.2870052.

[51] Christoph Molnar. *Interpretable Machine Learning. A Guide for Making Black Box Models Explainable*. https://christophm.github.io/interpretable-ml-book/. 2019.

[52] Q. Zhang et al. “An Efficient Deep Learning Model to Predict Cloud Workload for Industry Informatics”. In: *IEEE Transactions on Industrial Informatics* 14.7 (2018), pp. 3170–3178.

[53] Karin Kailing, Hans-Peter Kriegel, and Peer Kröger. “Density-Connected Subspace Clustering for High-Dimensional Data”. In: *Proceedings of the 2004 SIAM International Conference on Data Mining*. 2014, pp. 246–256.

[54] Y. Xu et al. “Industrial Big Data for Fault Diagnosis: Taxonomy, Review, and Applications”. In: *IEEE Access* 5 (2017), pp. 17368–17380.

[55] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. “Deep learning”. en. In: *Nature* 521.7553 (May 2015), pp. 436–444.

[56] Long Wen et al. “A New Convolutional Neural Network-Based Data-Driven Fault Diagnosis Method”. In: *IEEE Transactions on Industrial Electronics* 65.7 (2018), pp. 5990–5998. DOI: 10.1109/TIE.2017.2774777.

[57] Di Wu et al. “A Highly Accurate Framework for Self-Labeled Semisupervised Classification in Industrial Applications”. In: *IEEE Transactions on Industrial Informatics* 14.3 (2018), pp. 909–920. DOI: 10.1109/TII.2017.2737827.

[58] FengJi Luo et al. “Advanced Pattern Discovery-based Fuzzy Classification Method for Power System Dynamic Security Assessment”. In: *IEEE Transactions on Industrial Informatics* 11.2 (2015), pp. 416–426. DOI: 10.1109/TII.2015.2399698.

[59] Marco Cococcioni, Beatrice Lazzerini, and Sara Lioba Volpi. “Robust Diagnosis of Rolling Element Bearings Based on Classification Techniques”. In: *IEEE Transactions on Industrial Informatics* 9.4 (2013), pp. 2256–2263. DOI: 10.1109/TII.2012.2231084.

[60] H. Akagi. “New trends in active filters for power conditioning”. In: *IEEE Transactions on Industry Applications* 32.6 (1996), pp. 1312–1322. DOI: 10.1109/28.556633.

[61] J.R. Stack, T.G. Habetler, and R.G. Harley. “Fault classification and fault signature production for rolling element bearings in electric machines”. In: *4th IEEE International Symposium on Diagnostics for Electric Machines, Power Electronics and Drives, 2003. SDEMPED 2003.* 2003, pp. 172–176. DOI: 10.1109/DEMPED.2003.1234568.

[62] Christian Szegedy et al. “Going deeper with convolutions”. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 1–9. DOI: 10.1109/CVPR.2015.7298594.

[63] Andrew Howard et al. "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications". In: (Apr. 2017).

[64] Friedhelm Schwenker and Edmondo Trentin. "Pattern classification and clustering: A review of partially supervised learning approaches". In: *Pattern Recognition Letters* 37 (2014). Partially Supervised Learning for Pattern Recognition, pp. 4–14. ISSN: 0167-8655. DOI: https://doi.org/10.1016/j.patrec.2013.10.017. URL: https://www.sciencedirect.com/science/article/pii/S0167865513004091.

[65] J. F. Martins, V. Ferno Pires, and A. J. Pires. "Unsupervised Neural-Network-Based Algorithm for an On-Line Diagnosis of Three-Phase Induction Motor Stator Fault". In: *IEEE Transactions on Industrial Electronics* 54.1 (2007), pp. 259–264. DOI: 10.1109/TIE.2006.888790.

[66] M.R.G. Meireles, P.E.M. Almeida, and M.G. Simoes. "A comprehensive review for industrial applicability of artificial neural networks". In: *IEEE Transactions on Industrial Electronics* 50.3 (2003), pp. 585–601. DOI: 10.1109/TIE.2003.812470.

[67] Sanghoon Lee and M. M. Crawford. "Unsupervised Multistage Image Classification Using Hierarchical Clustering with a Bayesian Similarity Measure". In: *Trans. Img. Proc.* 14.3 (Mar. 2005), pp. 312–320. ISSN: 1057-7149. DOI: 10.1109/TIP.2004.841195. URL: https://doi.org/10.1109/TIP.2004.841195.

[68] Sanghoon Lee and M.M. Crawford. "Hierarchical clustering approach for unsupervised image classification of hyperspectral data". In: *IGARSS 2004. 2004 IEEE International Geoscience and Remote Sensing Symposium.* Vol. 2. 2004, 941–944 vol.2. DOI: 10.1109/IGARSS.2004.1368563.

[69] Zoltan Kato, Josiane Zerubia, and Marc Berthod. "Unsupervised Parallel Image Classification Using a Hierarchical Markovian Model". In: Apr. 1995, pp. 169–174. DOI: 10.1109/ICCV.1995.466790.

[70] SAMANWOY GHOSH-DASTIDAR and HOJJAT ADELI. "SPIKING NEURAL NETWORKS". In: *International Journal of Neural Systems* 19.04 (2009). PMID: 19731402, pp. 295–308. DOI: 10.1142/S0129065709002002. eprint: https://doi.org/10.1142/S0129065709002002. URL: https://doi.org/10.1142/S0129065709002002.

[71] Ian J. Goodfellow et al. *Generative Adversarial Networks*. 2014.

[72] Hélène Paugam-Moisy and Sander Bohte. "Computing with Spiking Neuron Networks". In: *Handbook of Natural Computing*. Ed. by Grzegorz Rozenberg, Thomas Bäck, and Joost N. Kok. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 335–376. ISBN: 978-3-540-92910-9. DOI: 10.1007/978-3-540-92910-9_10. URL: https://doi.org/10.1007/978-3-540-92910-9_10.

[73] E. Goodman and D. Ventura. "Effectively using recurrently-connected spiking neural networks". In: *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.* Vol. 3. 2005, 1542–1547 vol. 3. DOI: 10.1109/IJCNN.2005.1556107.

[74] Kunfeng Wang et al. "Generative adversarial networks: introduction and outlook". In: *IEEE/CAA Journal of Automatica Sinica* 4.4 (2017), pp. 588–598. DOI: 10.1109/JAS.2017.7510583.

[75] Christos Ferles, Yannis Papanikolaou, and Kevin Naidoo. "Denoising Autoencoder Self-Organizing Map (DASOM)". In: *Neural Networks* 105 (May 2018), pp. 112–131. DOI: 10.1016/j.neunet.2018.04.016.

[76] T. Kohonen et al. “Engineering applications of the self-organizing map”. In: *Proceedings of the IEEE* 84.10 (1996), pp. 1358–1384. DOI: 10.1109/5.537105.

[77] Cenk Budayan, Irem Dikmen, and M. Birgonul. “Comparing the performance of traditional cluster analysis, self-organizing maps and fuzzy C-means method for strategic grouping”. In: *Expert Systems with Applications* 36 (Nov. 2009), pp. 11772–11781. DOI: 10.1016/j.eswa.2009.04.022.

[78] A. Rauber, D. Merkl, and M. Dittenbach. “The growing hierarchical self-organizing map: exploratory analysis of high-dimensional data”. In: *IEEE Transactions on Neural Networks* 13.6 (2002), pp. 1331–1341. DOI: 10.1109/TNN.2002.804221.

[79] T. Kohonen. “The self-organizing map”. In: *Proceedings of the IEEE* 78.9 (1990), pp. 1464–1480. DOI: 10.1109/5.58325.

[80] ©[2022] IEEE. Reprinted with permission from C. S. Wickramasinghe, K. Amarasinghe, and M. Manic. “Deep Self-Organizing Maps for Unsupervised Image Classification”. In: *IEEE Transactions on Industrial Informatics* (2019), pp. 1–1.

[81] J. Vesanto and E. Alhoniemi. “Clustering of the self-organizing map”. In: *IEEE Transactions on Neural Networks* 11.3 (2000), pp. 586–600. DOI: 10.1109/72.846731.

[82] Christos Ferles, Yannis Papanikolaou, and Kevin Naidoo. “Denoising Autoencoder Self-Organizing Map (DASOM)”. In: *Neural Networks* 105 (May 2018), pp. 112–131. DOI: 10.1016/j.neunet.2018.04.016.

[83] Thore Graepel, Matthias Burger, and Klaus Obermayer. "Self-organizing maps: Generalizations and new optimization techniques". In: *Neurocomputing* 21.1 (1998), pp. 173–190. ISSN: 0925-2312. DOI: https://doi.org/10.1016/S0925-2312(98)00035-6. URL: https://www.sciencedirect.com/science/article/pii/S0925231298000356.

[84] Cenk Budayan, Irem Dikmen, and M. Birgonul. "Comparing the performance of traditional cluster analysis, self-organizing maps and fuzzy C-means method for strategic grouping". In: *Expert Systems with Applications* 36 (Nov. 2009), pp. 11772–11781. DOI: 10.1016/j.eswa.2009.04.022.

[85] Marco Cococcioni, B. Lazzerini, and Sara Volpi. "Robust Diagnosis of Rolling Element Bearings Based on Classification Techniques". In: *IEEE Transactions on Industrial Informatics* (Jan. 2012). DOI: 10.1109/TII.2012.2231084.

[86] Teuvo Kohonen. "Self-Organization of Very Large Document Collections: State of the Art". In: *ICANN 98*. Ed. by Lars Niklasson, Mikael Bodén, and Tom Ziemke. London: Springer London, 1998, pp. 65–74. ISBN: 978-1-4471-1599-1.

[87] J. Vesanto and E. Alhoniemi. "Clustering of the self-organizing map". In: *IEEE Transactions on Neural Networks* 11.3 (2000), pp. 586–600. DOI: 10.1109/72.846731.

[88] Nan Liu, Jinjun Wang, and Yihong Gong. "Deep Self-Organizing Map for visual classification". In: *2015 International Joint Conference on Neural Networks (IJCNN)*. 2015, pp. 1–6. DOI: 10.1109/IJCNN.2015.7280357.

[89] ©[2022] IEEE. Reprinted with permission from Chathurika S. Wickramasinghe et al. "Deep Self-Organizing Maps for Visual Data Mining". In: *2018*

11th International Conference on Human System Interaction (HSI). 2018, pp. 304–310. DOI: 10.1109/HSI.2018.8430845.

[90] Thore Graepel, Matthias Burger, and Klaus Obermayer. "Self-organizing maps: Generalizations and new optimization techniques". In: *Neurocomputing* 21.1 (1998), pp. 173–190. ISSN: 0925-2312. DOI: https://doi.org/10.1016/S0925-2312(98)00035-6. URL: https://www.sciencedirect.com/science/article/pii/S0925231298000356.

[91] Cenk Budayan, Irem Dikmen, and M. Birgonul. "Comparing the performance of traditional cluster analysis, self-organizing maps and fuzzy C-means method for strategic grouping". In: *Expert Systems with Applications* 36 (Nov. 2009), pp. 11772–11781. DOI: 10.1016/j.eswa.2009.04.022.

[92] Teuvo Kohonen et al. *Chapter 10 Self-organization of very large document collections*.

[93] Teuvo Kohonen. "The self-organizing map". In: *Neurocomputing* 21.1 (1998), pp. 1–6. ISSN: 0925-2312. DOI: https://doi.org/10.1016/S0925-2312(98)00030-7. URL: https://www.sciencedirect.com/science/article/pii/S0925231298000307.

[94] T. Kohonen et al. "Engineering applications of the self-organizing map". In: *Proceedings of the IEEE* 84.10 (1996), pp. 1358–1384. DOI: 10.1109/5.537105.

[95] Andreas Rauber, Dieter Merkl, and Michael Dittenbach. "The Growing Hierarchical Self-Organizing Map: Exploratory Analysis of High-Dimensional Data". In: *Neural Networks, IEEE Transactions on* 13 (Nov. 2002), pp. 1331–. DOI: 10.1109/TNN.2002.804221.

[96] Y. Lecun et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. DOI: 10.1109/5.726791.

[97] Matthew D. Zeiler and Rob Fergus. “Visualizing and Understanding Convolutional Networks”. In: *Computer Vision – ECCV 2014*. Ed. by David Fleet et al. Cham: Springer International Publishing, 2014, pp. 818–833. ISBN: 978-3-319-10590-1.

[98] Ritwik K. Kumar, Arunava Banerjee, and B. Vemuri. “Volterrafaces: Discriminant analysis using Volterra kernels”. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition* (2009), pp. 150–155.

[99] Pengfei Zhu et al. “Multi-scale Patch Based Collaborative Representation for Face Recognition with Margin Distribution Optimization”. In: *Computer Vision – ECCV 2012*. Ed. by Andrew Fitzgibbon et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 822–835. ISBN: 978-3-642-33718-5.

[100] Dheeru Dua and Casey Graff. *UCI Machine Learning Repository*. 2017. URL: http://archive.ics.uci.edu/ml.

[101] Alexander Vergara et al. “Chemical gas sensor drift compensation using classifier ensembles”. In: *Sensors and Actuators B: Chemical* 166-167 (2012), pp. 320–329. ISSN: 0925-4005. DOI: https://doi.org/10.1016/j.snb.2012.01.074. URL: https://www.sciencedirect.com/science/article/pii/S0925400512002018.

[102] ©[2022] IEEE. Reprinted with permission from Chathurika S. Wickramasinghe, Kasun Amarasinghe, and Milos Manic. “Parallalizable deep self-organizing maps for image classification”. In: *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*. 2017, pp. 1–7. DOI: 10.1109/SSCI.2017.8285443.

[103] G.E. Hinton and R.R. Salakhutdinov. "Reducing the Dimensionality of Data with Neural Networks". In: *Science (New York, N.Y.)* 313 (Aug. 2006), pp. 504–7. DOI: 10.1126/science.1127647.

[104] *White Paper on Artificial Intelligence: a European approach to excellence and trust*. Feb. 2020. URL: https://ec.europa.eu/info/publications/white-paper-artificial-intelligence-european-approach-excellence-and-trust_en.

[105] Robin.materese@nist.gov. *Artificial intelligence*. Mar. 2021. URL: https://www.nist.gov/artificial-intelligence.

[106] Thelma.allen@nist.gov. *AI Foundational Research - Explainability*. Jan. 2021. URL: https://www.nist.gov/artificial-intelligence/ai-foundational-research-explainability.

[107] Yoshua Bengio Yann LeCun and Geoffrey Hinton. "Deep learning". In: *Nature* (2015).

[108] Andrea Morichetta, Pedro Casas, and Marco Mellia. "EXPLAIN-IT". In: *Proceedings of the 3rd ACM CoNEXT Workshop on Big DAta, Machine Learning and Artificial Intelligence for Data Communication Networks - Big-DAMA '19* (2019). DOI: 10.1145/3359992.3366639. URL: http://dx.doi.org/10.1145/3359992.3366639.

[109] O. Loyola-González et al. "An Explainable Artificial Intelligence Model for Clustering Numerical Databases". In: *IEEE Access* 8 (2020), pp. 52370–52384. DOI: 10.1109/ACCESS.2020.2980581.

[110] Enguerrand Horel et al. *Explainable Clustering and Application to Wealth Management Compliance*. 2020. arXiv: 1909.13381 [stat.AP].

[111] Michal Moshkovitz et al. "Explainable k-Means and k-Medians Clustering". In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, 13–18 Jul 2020, pp. 7055–7065. URL: http://proceedings.mlr.press/v119/moshkovitz20a.html.

[112] Kacper Sokol and Peter Flach. "Explainability fact sheets". In: *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency* (Jan. 2020). DOI: 10.1145/3351095.3372870. URL: http://dx.doi.org/10.1145/3351095.3372870.

[113] Sanjoy Dasgupta et al. *Explainable k-Means and k-Medians Clustering*. 2020. arXiv: 2002.12538 [cs.LG].

[114] Nave Frost, Michal Moshkovitz, and Cyrus Rashtchian. *ExKMC: Expanding Explainable k-Means Clustering*. 2020. arXiv: 2006.02399 [cs.LG].

[115] Quoc Phong Nguyen et al. "GEE: A Gradient-based Explainable Variational Autoencoder for Network Anomaly Detection". In: *2019 IEEE Conference on Communications and Network Security (CNS)*. 2019, pp. 91–99. DOI: 10.1109/CNS.2019.8802833.

[116] Marion Neumeier et al. *Variational Autoencoder-Based Vehicle Trajectory Prediction with an Interpretable Latent Space*. 2021. arXiv: 2103.13726 [cs.LG].

[117] Mariana Curi et al. "Interpretable Variational Autoencoders for Cognitive Models". In: *2019 International Joint Conference on Neural Networks (IJCNN)*. 2019, pp. 1–8. DOI: 10.1109/IJCNN.2019.8852333.

[118] Q. Zhang et al. “A Tensor-Train Deep Computation Model for Industry Informatics Big Data Feature Learning”. In: *IEEE Transactions on Industrial Informatics* 14.7 (July 2018), pp. 3197–3204.

[119] V W Ajin and Lekshmy D Kumar. “Big data and clustering algorithms”. In: *2016 International Conference on Research Advances in Integrated Navigation Systems (RAINS)*. 2016, pp. 1–5. DOI: 10.1109/RAINS.2016.7764405.

[120] M. Kang et al. “A Hybrid Feature Selection Scheme for Reducing Diagnostic Performance Deterioration Caused by Outliers in Data-Driven Diagnostics”. In: *IEEE Transactions on Industrial Electronics* 63.5 (May 2016), pp. 3299–3310.

[121] Yazhou Ren et al. “Semi-supervised deep embedded clustering”. In: *Neurocomputing* 325 (2019), pp. 121–130.

[122] Jaime Zabalza et al. “Novel segmented stacked autoencoder for effective dimensionality reduction and feature extraction in hyperspectral imaging”. In: *Neurocomputing* 185 (2016), pp. 1–10.

[123] Aicke Hinrichs, Joscha Prochno, and Mario Ullrich. “The curse of dimensionality for numerical integration on general domains”. In: *Journal of Complexity* 50 (2019), pp. 25–42.

[124] Dejiao Zhang et al. “Deep Unsupervised Clustering Using Mixture of Autoencoders”. In: *CoRR* abs/1712.07788 (2017).

[125] Chen Xing, Li Ma, and Xiaoquan Yang. “Stacked denoise autoencoder based feature extraction and classification for hyperspectral images”. In: *Journal of Sensors* 2016 (2016).

[126] Wenjun Sun et al. “A sparse auto-encoder-based deep neural network approach for induction motor faults classification”. In: *Measurement* 89 (2016), pp. 171–178.

[127] Yasi Wang, Hongxun Yao, and Sicheng Zhao. “Auto-encoder based dimensionality reduction”. In: *Neurocomputing* 184 (2016). RoLoD: Robust Local Descriptors for Computer Vision 2014, pp. 232–242.

[128] L. Bottou et al. “Scaling Learning Algorithms toward AI”. In: *Large-Scale Kernel Machines*. MITP, 2007, pp. 321–359. ISBN: null.

[129] Deyu Bo et al. “Structural Deep Clustering Network”. In: *Proceedings of The Web Conference 2020*. WWW ’20. Taipei, Taiwan: Association for Computing Machinery, 2020, pp. 1400–1410. ISBN: 9781450370233. DOI: 10.1145/3366423.3380214. URL: https://doi.org/10.1145/3366423.3380214.

[130] K. He et al. “Deep Residual Learning for Image Recognition”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

[131] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: *CoRR* abs/1512.03385 (2015).

[132] Alireza Zaeemzadeh, Nazanin Rahnavard, and Mubarak Shah. “Norm-Preservation: Why Residual Networks Can Become Extremely Deep?” In: *IEEE transactions on pattern analysis and machine intelligence* (2020).

[133] K. He et al. “Identity Mappings in Deep Residual Networks”. In: *CoRR* abs/1603.05027 (2016).

[134] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. "Training Very Deep Networks". In: *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*. NIPS'15. Montreal, Canada: MIT Press, 2015, pp. 2377–2385.

[135] G. Huang et al. "Densely Connected Convolutional Networks". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 2261–2269.

[136] Mykola Pechenizkiy. "The Impact of Feature Extraction on the Performance of a Classifier: kNN, Naïve Bayes and C4.5". In: *Advances in Artificial Intelligence*. Ed. by Balázs Kégl and Guy Lapalme. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 268–279. ISBN: 978-3-540-31952-8.

[137] S. Khalid, T. Khalil, and S. Nasreen. "A survey of feature selection and feature extraction techniques in machine learning". In: *2014 Science and Information Conference*. 2014, pp. 372–378.

[138] Hee Sun Park, René Dailey, and Daisy Lemus. "The Use of Exploratory Factor Analysis and Principal Components Analysis in Communication Research". In: *Human Communication Research* 28.4 (Jan. 2006), pp. 562–577.

[139] Clemens Reimann, Peter Filzmoser, and Robert G. Garrett. "Factor analysis applied to regional geochemical data: problems and possibilities". In: *Applied Geochemistry* 17.3 (2002), pp. 185–206.

[140] P. Hansen. "The truncatedSVD as a method for regularization". In: *BIT Numerical Mathematics* 27 (1987), pp. 534–553.

[141] Chen Lu et al. “Fault diagnosis of rotary machinery components using a stacked denoising autoencoder-based health state identification”. In: *Signal Processing* 130 (2017), pp. 377–388.

[142] G. E. Hinton and R. R. Salakhutdinov. “Reducing the Dimensionality of Data with Neural Networks”. In: *Science* 313.5786 (2006), pp. 504–507.

[143] Chris Ding and Xiaofeng He. “K-means Clustering via Principal Component Analysis”. In: *Proceedings of the Twenty-first International Conference on Machine Learning*. ICML ’04. New York, NY, USA: ACM, 2004, pp. 29–.

[144] Fengfu Li, Hong Qiao, and Bo Zhang. “Discriminatively boosted image clustering with fully convolutional auto-encoders”. In: *Pattern Recognition* 83 (2018), pp. 161–173.

[145] David Balduzzi et al. “The Shattered Gradients Problem: If resnets are the answer, then what is the question?” In: (Feb. 2017).

[146] Qingchen Zhang et al. “A survey on deep learning for big data”. In: *Information Fusion* 42 (2018), pp. 146–157.

[147] Y. LECUN. “THE MNIST DATABASE of handwritten digits”. In: *http://yann.lecun.com/exd* (). URL: https://ci.nii.ac.jp/naid/10027939599/en/.

[148] Alex Krizhevsky. “Learning Multiple Layers of Features from Tiny Images”. In: *University of Toronto* (May 2012).

[149] Han Xiao, Kashif Rasul, and Roland Vollgraf. “Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms”. In: *CoRR* abs/1708.07747 (2017). arXiv: 1708.07747. URL: http://arxiv.org/abs/1708.07747.

[150] François Chollet et al. *Keras*. 2015.

[151] ©[2022] IEEE. Reprinted with permission from Chathurika Wickramasinghe, Daniel Marino, and Milos Manic. "Deep Embedded Clustering with ResNets". In: *2021 14th International Conference on Human System Interaction (HSI)*. 2021, pp. 1–6. DOI: 10.1109/HSI52170.2021.9538747.

[152] M. M. Saeed, Z. Aghbari, and Mohammed Alsharidah. "Big data clustering techniques based on Spark: a literature review". In: *PeerJ Computer Science* 6 (2020).

[153] Meenu Dave and Hemant Gianey. "Different clustering algorithms for Big Data analytics: A review". In: *2016 International Conference System Modeling Advancement in Research Trends (SMART)*. 2016, pp. 328–333. DOI: 10.1109/SYSMART.2016.7894544.

[154] Amir Ahmad and Shehroz S. Khan. "Survey of State-of-the-Art Mixed Data Clustering Algorithms". In: *IEEE Access* 7 (2019), pp. 31883–31902. DOI: 10.1109/ACCESS.2019.2903568.

[155] Mamta Mittal et al. "Clustering approaches for high-dimensional databases: A review". In: *WIREs Data Mining and Knowledge Discovery* 9.3 (2019), e1300. DOI: https://doi.org/10.1002/widm.1300. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/widm.1300. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/widm.1300.

[156] Erxue Min et al. "A Survey of Clustering With Deep Learning: From the Perspective of Network Architecture". In: *IEEE Access* 6 (2018), pp. 39501–39514. DOI: 10.1109/ACCESS.2018.2855437.

[157] Chunfeng Song et al. "Auto-encoder Based Data Clustering". In: *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*.

Ed. by José Ruiz-Shulcloper and Gabriella Sanniti di Baja. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 117–124. ISBN: 978-3-642-41822-8.

[158] Junyuan Xie, Ross B. Girshick, and Ali Farhadi. "Unsupervised Deep Embedding for Clustering Analysis". In: *CoRR* abs/1511.06335 (2015).

[159] Kaiming He et al. *Identity Mappings in Deep Residual Networks*. 2016. arXiv: 1603.05027 [cs.CV].

[160] Xifeng Guo et al. "Improved Deep Embedded Clustering with Local Structure Preservation". In: *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. IJCAI'17. AAAI Press, 2017, pp. 1753–1759.

[161] HAJI Zakaria et al. "Recent Advancements and Developments for Electric Vehicle Technology". In: *2019 International Conference of Computer Science and Renewable Energies (ICCSRE)*. 2019, pp. 1–6. DOI: 10.1109/ICCSRE.2019.8807726.

[162] Li Zhu et al. "Big Data Analytics in Intelligent Transportation Systems: A Survey". In: *IEEE Transactions on Intelligent Transportation Systems* 20.1 (2019), pp. 383–398. DOI: 10.1109/TITS.2018.2815678.

[163] Juan Guerrero-Ibáñez, Sherali Zeadally, and Juan Contreras-Castillo. "Sensor Technologies for Intelligent Transportation Systems". In: *Sensors* 18.4 (2018).

[164] Azzedine F. M. Boukerche and Rodolfo W. L. Coutinho. "Crowd Management: The Overlooked Component of Smart Transportation Systems". In: *IEEE Communications Magazine* 57 (2019), pp. 48–53.

[165] Yosra Fraiji et al. "Cyber security issues of Internet of electric vehicles". In: *2018 IEEE Wireless Communications and Networking Conference (WCNC)*. 2018, pp. 1–6. DOI: 10.1109/WCNC.2018.8377181.

[166] Eunbi Seo, Hyun Min Song, and Huy Kang Kim. "GIDS: GAN based Intrusion Detection System for In-Vehicle Network". In: *2018 16th Annual Conference on Privacy, Security and Trust (PST)*. 2018, pp. 1–6. DOI: 10.1109/PST.2018.8514157.

[167] Moayad Aloqaily et al. "An intrusion detection system for connected vehicles in smart cities". In: *Ad Hoc Networks* 90 (2019). Recent advances on security and privacy in Intelligent Transportation Systems, p. 101842. ISSN: 1570-8705. DOI: https://doi.org/10.1016/j.adhoc.2019.02.001. URL: https://www.sciencedirect.com/science/article/pii/S1570870519301131.

[168] Manoj Basnet and Mohd. Hasan Ali. "Deep Learning-based Intrusion Detection System for Electric Vehicle Charging Station". In: *2020 2nd International Conference on Smart Power Internet Energy Systems (SPIES)*. 2020, pp. 408–413. DOI: 10.1109/SPIES48661.2020.9243152.

[169] Daniel L. Marino, Chathurika S. Wickramasinghe, and Milos Manic. "An Adversarial Approach for Explainable AI in Intrusion Detection Systems". In: *IECON 2018 - 44th Annual Conference of the IEEE Industrial Electronics Society*. 2018, pp. 3237–3243. DOI: 10.1109/IECON.2018.8591457.

[170] Ansam Khraisat et al. "Survey of intrusion detection systems: techniques, datasets and challenges". In: *Cybersecurity* 2 (Dec. 2019). DOI: 10.1186/s42400-019-0038-7.

[171] Gisung Kim, Seungmin Lee, and Sehun Kim. "A novel hybrid intrusion detection method integrating anomaly detection with misuse detection". In: *Expert Systems with Applications* 41.4, Part 2 (2014), pp. 1690–1700. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2013.08.066. URL: https://www.sciencedirect.com/science/article/pii/S0957417413006878.

[172] ©[2022] IEEE. Reprinted with permission from Chathurika S. Wickramasinghe et al. "Explainable Unsupervised Machine Learning for Cyber-Physical Systems". In: *IEEE Access* 9 (2021), pp. 131824–131843. DOI: 10.1109/ACCESS.2021.3112397.

[173] Hyunsung Lee, Seong Hoon Jeong, and Huy Kang Kim. "OTIDS: A Novel Intrusion Detection System for In-vehicle Network by Using Remote Frame". In: *2017 15th Annual Conference on Privacy, Security and Trust (PST)*. 2017, pp. 57–5709. DOI: 10.1109/PST.2017.00017.

[174] Miki E. Verma et al. "ROAD: The Real ORNL Automotive Dynamometer Controller Area Network Intrusion Detection Dataset (with a comprehensive CAN IDS dataset survey & guide)". In: *CoRR* abs/2012.14600 (2020). arXiv: 2012.14600. URL: https://arxiv.org/abs/2012.14600.

[175] Andrew Tomlinson et al. "Detection of Automotive CAN Cyber-Attacks by Identifying Packet Timing Anomalies in Time Windows". In: *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*. 2018, pp. 231–238. DOI: 10.1109/DSN-W.2018.00069.

[176] Michael R. Moore et al. "Modeling Inter-Signal Arrival Times for Accurate Detection of CAN Bus Signal Injection Attacks: A Data-Driven Approach to in-Vehicle Intrusion Detection". In: *Proceedings of the 12th Annual Conference on Cyber and Information Security Research*. CISRC '17. Oak Ridge, Tennessee, USA: Association for Computing Machinery, 2017. ISBN: 9781450348553.

[177] Markus Hanselmann et al. “CANet: An Unsupervised Intrusion Detection System for High Dimensional CAN Bus Data”. In: *IEEE Access* 8 (Mar. 2020), pp. 58194–58205. DOI: 10.1109/ACCESS.2020.2982544.

[178] Wonsuk Choi et al. “VoltageIDS: Low-Level Communication Characteristics for Automotive Intrusion Detection System”. In: *IEEE Transactions on Information Forensics and Security* 13.8 (2018), pp. 2114–2129. DOI: 10.1109/TIFS.2018.2812149.

[179] Adrian Taylor, Sylvain Leblanc, and Nathalie Japkowicz. “Anomaly Detection in Automobile Control Network Data with Long Short-Term Memory Networks”. In: *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*. 2016, pp. 130–139. DOI: 10.1109/DSAA.2016.20.

[180] Min-Joo Kang and Je-Won Kang. “Intrusion Detection System Using Deep Neural Network for In-Vehicle Network Security”. In: *PLoS ONE* 11 (2016).

[181] Md Delwar Hossain et al. “An Effective In-Vehicle CAN Bus Intrusion Detection System Using CNN Deep Learning Approach”. In: *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*. 2020, pp. 1–6. DOI: 10.1109/GLOBECOM42002.2020.9322395.

[182] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. “Adversarial Machine Learning at Scale”. In: *CoRR* abs/1611.01236 (2016). arXiv: 1611.01236. URL: http://arxiv.org/abs/1611.01236.

[183] Ram Shankar Siva Kumar et al. “Adversarial Machine Learning-Industry Perspectives”. In: *2020 IEEE Security and Privacy Workshops (SPW)*. 2020, pp. 69–75. DOI: 10.1109/SPW50608.2020.00028.

[184] Nicolas Papernot et al. “Practical Black-Box Attacks against Machine Learning”. In: Apr. 2017, pp. 506–519. DOI: 10.1145/3052973.3053009.

[185] Battista Biggio and Fabio Roli. “Wild patterns: Ten years after the rise of adversarial machine learning”. In: *Pattern Recognition* 84 (2018), pp. 317–331. ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2018.07.023. URL: https://www.sciencedirect.com/science/article/pii/S0031320318302565.

[186] K. Amarasinghe et al. “Framework for Data Driven Health Monitoring of Cyber-Physical Systems”. In: *2018 Resilience Week (RWS)*. 2018, pp. 25–30. DOI: 10.1109/RWEEK.2018.8473535.

[187] Abdelouahid Derhab et al. “Histogram-Based Intrusion Detection and Filtering Framework for Secure and Safe In-Vehicle Networks”. In: *IEEE Transactions on Intelligent Transportation Systems* (2021), pp. 1–14. DOI: 10.1109/TITS.2021.3088998.

[188] Ivo Berger et al. “Comparative Study of Machine Learning Methods for In-Vehicle Intrusion Detection”. In: *CyberICPS/SECPRE@ESORICS*. 2018.

[189] Yoshua Bengio. “Deep Learning of Representations for Unsupervised and Transfer Learning”. In: *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*. Ed. by Isabelle Guyon et al. Vol. 27. Proceedings of Machine Learning Research. Bellevue, Washington, USA: PMLR, 2012, pp. 17–36. URL: https://proceedings.mlr.press/v27/bengio12a.html.

[190] Aditya Siddhant, Anuj Goyal, and Angeliki Metallinou. “Unsupervised Transfer Learning for Spoken Language Understanding in Intelligent Agents”. In: *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference*

and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence. AAAI'19/IAAI'19/EAAI'19. Honolulu, Hawaii, USA: AAAI Press, 2019. ISBN: 978-1-57735-809-1. DOI: 10.1609/aaai.v33i01.33014959. URL: https://doi.org/10.1609/aaai.v33i01.33014959.

[191] Mohanad Abukmeil et al. "A Survey of Unsupervised Generative Models for Exploratory Data Analysis and Representation Learning". In: *ACM Comput. Surv.* 54.5 (July 2021). ISSN: 0360-0300. DOI: 10.1145/3450963. URL: https://doi.org/10.1145/3450963.

[192] Andrea Papenmeier, Gwenn Englebienne, and Christin Seifert. "How model accuracy and explanation fidelity influence user trust". In: *arXiv e-prints*, arXiv:1907.12652 (July 2019), arXiv:1907.12652. arXiv: 1907.12652 [cs.CY].

[193] Chih-Kuan Yeh et al. "On the (In)fidelity and Sensitivity of Explanations". In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach et al. Vol. 32. Curran Associates, Inc., 2019. URL: https://proceedings.neurips.cc/paper/2019/file/a7471fdc77b3435276507cc8f2dc2569-Paper.pdf.

[194] Ahmad Kamal Mohd Nor et al. "Overview of explainable artificial intelligence for prognostic and health management of industrial assets based on preferred reporting items for systematic reviews and meta-analyses". English. In: *Sensors (Switzerland)* 21.23 (Dec. 2021). ISSN: 1424-8220. DOI: 10.3390/s21238020.

[195] Tien N. Nguyen and Raymond Choo. "Human-in-the-Loop XAI-enabled Vulnerability Detection, Investigation, and Mitigation". In: *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. 2021, pp. 1210–1212. DOI: 10.1109/ASE51524.2021.9678840.

"In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of Virginia Commonwealth University's products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to `http://www.ieee.org/publications_standards/publications/rights/rights_link.html` to learn how to obtain a License from RightsLink. If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation."

VITA

Chathurika S. Wickramasinghe (Alias: Chathurika S. Wickramasinghe Brahmana Mudiyanselage) was born on March 20th 1991 in Bokkawala (Kandy), Sri Lanka. She received her B.Sc. in Computer Science from the University of Peradeniya in Sri Lanka in 2016. She joined the Doctor of Philosophy program at Virginia Commonwealth University in Richmond, Virginia, in 2017. She has has over six years of research and development experience, collaborating with universities, the U.S. Department of Energy National Laboratories (US DOE), and Industry partners. He has authored over 22 articles in peer reviewed journals and conferences. She received two awards from the computer science department at VCU: the "early-career research award" in 2018 and the "best paper award" in 2019. Furthermore, she was one of four finalists for the Outstanding Graduate Research Award presented by the College of Engineering, VCU 2021. Her research interests are Interpretable Machine Learning, Neural Networks, Unsupervised Machine Learning, and machine learning for Cyber-physical system security.