



Faculty of Computer Science and Information Technology

**Enhanced Adaptive Neuro-Fuzzy Inference System Classification
Method for Intrusion Detection**

Liu Jia

**Doctor of Philosophy
2024**

Enhanced Adaptive Neuro-Fuzzy Inference System Classification Method for Intrusion Detection

Liu Jia

A thesis submitted

In fulfillment of the requirements for the degree of Doctor of Philosophy

(Software Engineering)

Faculty of Computer Science and Information Technology

UNIVERSITI MALAYSIA SARAWAK

2024

DECLARATION

I declare that the work in this thesis was carried out in accordance with the regulations of Universiti Malaysia Sarawak. Except where due acknowledgements have been made, the work is that of the author alone. The thesis has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.

.....

Signature



Name:

Liu Jia

Matric No.:

20010028

Faculty of Computer Science and Information Technology

Universiti Malaysia Sarawak

Date :

ACKNOWLEDGEMENT

I would like to take this opportunity to those who have contributed directly or indirectly to this research.

My sincere gratitude to the Faculty of Computer Science and Information Technology, for the advice and support given during my period of study in Universiti Malaysia Sarawak.

I would like to thank my two supervisors, Professor Dr. Wang Yin Chai and Associate Professor Dr. Teh Chee Siong, for their guidance. In the process of learning, I benefited a lot from their serious attitude and spirit of learning.

I am also grateful to my parents and husband for their support during my studies. Without them, I would not have been able to complete my studies on time.

Finally, I would like to thank the management of the Universiti Malaysia Sarawak for making it possible for me to complete my study here in Sarawak. Thank you all.

ABSTRACT

Nowadays, the rapid advancement of technologies such as e-commerce, mobile payments, cloud computing, big data and analytics, IoT, AI, machine learning, and social media has greatly contributed to economic growth. However, it has also resulted in an increased frequency of cyberattacks. In order to ensure network security, intrusion detection has become a crucial technology. However, due to the inherent uncertainty in distinguishing between normal and abnormal behaviours, intrusion detection problems can be considered as fuzzy classification problems. Many traditional classification methods are inadequate in handling this fuzziness, leading to suboptimal performance in intrusion detection. To address this issue, the Adaptive-Network-based Fuzzy Inference System (ANFIS) has emerged as a promising approach. ANFIS combines the uncertainty processing ability of fuzzy logic with the learning process of Neural Networks (NNs). It has demonstrated excellent performance in various domains, including intrusion detection. However, due to the constraint of having only 5 layers, ANFIS is unable to identify higher-level and more abstract representations of the data. To address this problem, this study first uses CART (Classification and Regression Tree) to enhance the depth of ANFIS, providing a deeper and interpretable hybrid architecture. The method is then compared to single CART, single ANFIS, BPNN, GA-ANFIS, PSO-ANFIS, and CSO-ANFIS. The results demonstrate a detection rate of 99.86% and a false alarm rate of 0.14% on the intrusion detection dataset, KDDTrain+, which is one of the most widely used benchmark datasets. The proposed method outperforms CSO-ANFIS by exhibiting a 4.06% higher detection rate (improving from 95.80%) and a 2.31% lower false alarm rate (compared to 3.45%). On the KDDTest+ dataset, the proposed method also outperforms single CART and ANFIS in terms of various metrics other than precision. Since the CART tree is a binary tree, it can only represent the relationship between data

through a split based on a single attribute at a single tree node. Therefore, this binary tree cannot analyse complex features of mixed attributes and restricts the CART tree's deep-level feature recognition ability. Although the combination of ANFIS and CART provides deeper feature recognition for ANFIS, it still lacks the ability to recognize deep-level features and mixed-attribute features due to the limitations of both ANFIS and CART. Therefore, deep-level feature analysis, recognizing complex and mixed factors, has essential research value and significance for improving the efficiency and accuracy of intrusion detection. This study further enhances the depth of ANFIS and identifies deep features by using the ResNet (Residual Network) with ANFIS. The proposed hybrid model is able to deepen the structure of ANFIS and provide a deep and hybrid architecture. A comparison study with the proposed method with MVO-ANN, FC-ANN, DN, CSO-ANFIS, single ANFIS and single ResNet is conducted. The results show improved performance with a 10.68% better detection rate (improving from 88.2%) and a 10.68% lower false alarm rate (compared to 11.79%) than a single ANFIS. Furthermore, the proposed method outperforms MVO-ANN, FC-ANN, DN, CSO-ANFIS, single ANFIS, and single ResNet in terms of detection rate, F1-score, and recall rate. Additionally, standard deviation and proposed adaptive K-means algorithms have been employed to minimize the generated rules by ANFIS from the proposed hybrid models. This approach can dynamically minimize the interval number of each continuous attribute. As a result, the number of fuzzy rules generated by ANFIS is reduced, subsequently improving the training and prediction efficiency of the proposed hybrid models.

Keywords: Adaptive Neuro-fuzzy Inference System, Interpretable Hybrid Neural Networks, Classification and Regression Tree, Deep Residual Networks, Intrusion detection

Kaedah Klasifikasi Sistem Inferens Neuro-Fuzzy Adaptive yang Dipertingkatkan untuk Pengesanan Pencerobohan

ABSTRAK

Dalam zaman moden ini, kemajuan pesat dalam teknologi seperti e-dagang, pembayaran mudah alih, pengkomputeran awan, data besar dan analitik, IoT, AI, pembelajaran mesin, dan media sosial telah memberikan sumbangan yang besar kepada pertumbuhan ekonomi. Walau bagaimanapun, ia juga telah mengakibatkan peningkatan frekuensi serangan siber. Bagi memastikan keselamatan rangkaian, pengesanan pencerobohan telah menjadi teknologi yang penting. Namun demikian, disebabkan ketidakpastian dalam membezakan tingkah laku normal dan tidak normal, masalah pengesanan pencerobohan boleh dianggap sebagai masalah klasifikasi yang kabur. Banyak kaedah klasifikasi tradisional tidak memadai dalam menangani ketidakpastian ini, yang membawa kepada prestasi yang kurang memuaskan dalam pengesanan pencerobohan. Bagi menangani isu ini, Sistem Inferens Kabur Berasaskan Rangkaian Adaptif (ANFIS) adalah pendekatan yang sesuai dipilih. ANFIS menggabungkan keupayaan pemprosesan ketidakpastian logik kabur dengan proses pembelajaran Rangkaian Neural (NNs). Ia telah menunjukkan prestasi klasifikasi yang baik dalam pelbagai bidang, termasuk pengesanan pencerobohan. Walau bagaimanapun, disebabkan sistem ini hanya mempunyai 5 lapisan NNs, kekangan ini menyebabkan ANFIS tidak dapat mengenal perwakilan data yang rumit dan lebih abstrak. Untuk mengatasi masalah ini, kajian ini terlebih dahulu menggunakan CART (Classification and Regression Tree) untuk meningkatkan kedalaman ANFIS. Rekabentuk hibrid yang berlapisan lebih dalam ini akan berupaya menerangkan perwakilan data yang rumit dan lebih abstrak. Kaedah ini kemudiannya dibandingkan dengan single CART, single ANFIS, BPNN, GA-ANFIS, PSO-ANFIS, dan CSO-ANFIS. Keputusan menunjukkan kadar pengesanan sebanyak

99.86% dan kadar penggera palsu sebanyak 0.14% diperolehi dengan menggunakan set data pengesanan pencerobohan (KDDTrain+), yang mana set data ini merupakan salah satu set data penanda aras yang paling banyak digunakan. Kaedah yang dicadangkan dapat meningkatkan kadar pengesanan 4.06% lebih tinggi (bertambah baik daripada 95.80%) dan kadar penggera palsu 2.31% lebih rendah (berbanding 3.45%) dengan model CSO-ANFIS. Dengan menggunakan dataset KDDTest+, kaedah yang dicadangkan juga mengatasi prestasi CART tunggal dan ANFIS dari pelbagai ukuran metrik. Disebabkan pepohon CART adalah pepohon binari, ia hanya dapat mewakili hubungan data melalui atribut tunggal pada satu node pepohon. Oleh itu, pepohon binari ini tidak dapat menganalisa sifat data yang mempunyai atribut campuran yang kompleks, ini selanjutnya membatasi kemampuan pengenalan sifat data dengan pepohon daripada CART. Walaupun kombinasi ANFIS dan CART menghasilkan lapisan NNs yang lebih dalam untuk ANFIS, ia masih kurang mampu mengenali sifat data yang mendalam dan atribut data yang bersifat campuran kerana limitasi oleh kedua-dua model ANFIS dan CART. Oleh kerana itu, satu model yang dapat meningkatkan efisiensi dan ketepatan menganalisa sifat data yang mendalam dan atribut data yang bersifat campuran dan kompleks perlu dihasilkan untuk pengesanan pencerobohan. Dalam kajian ini, kedalaman lapisan ANFIS akan ditingkatkan dengan menggunakan ResNet (Residual Network). Model hybrid yang dicadangkan dalam kajian ini mampu memperdalamkan struktur ANFIS. Beberapa kajian perbandingan telah dilaksanakan dengan model hybrid yang dicadangkan dan model-model lain, seperti MVO-ANN, FC-ANN, DN, CSO-ANFIS, ANFIS tunggal dan ResNet tunggal. Keputusan menunjukkan prestasi peringkatan kadar pengesanan sebanyak 10.68% (bertambah baik daripada 88.2%) dan kadar penggera palsu 10.68% lebih rendah (berbanding 11.79%) dengan model ANFI tunggal. Kaedah yang dicadangkan juga mengatasi prestasi MVO-

ANN, FC-ANN, DN, CSO-ANFIS, ANFI tunggal, dan ResNet tunggal dari segi kadar pengesanan, skor F1, dan kadar recall. Selain itu, sisihan piawai dan algoritma k-means adaptif yang dicadangkan juga dapat mengurangkan tanda peraturan (rules) yang dihasilkan oleh ANFIS daripada model hibrid yang dicadangkan. Pendekatan dinamik ini dapat meminimumkan bilangan selang setiap atribut berterusan data yang digunakan. Akibatnya, bilangan tanda peraturan kabur yang dihasilkan oleh ANFIS juga dikurangkan, dan selanjutnya meningkatkan kecekapan latihan dan keupayaan ramalan model hibrid yang dicadangkan. Pendekatan ini secara dinamik dapat meminimumkan bilangan selang setiap atribut berterusan. Akibatnya, bilangan peraturan kabur yang dihasilkan oleh ANFIS dikurangkan, seterusnya meningkatkan kecekapan latihan dan ramalan model hibrid yang dicadangkan.

Kata kunci: *Sistem Inferensi Neuro-Fuzzy Adaptif, Rangkaian Neural Hybrid Boleh Tafsir, Pepohon Klasifikasi dan Regresi, Jaringan Residual Dalam, Pengesanan Pencerobohan*

TABLE OF CONTENTS

	Page
DECLARATION	i
ACKNOWLEDGEMENT	ii
ABSTRACT	iii
ABSTRAK	v
TABLE OF CONTENTS	viii
LIST OF TABLES	xii
LIST OF FIGURES	xiv
LIST OF ABBREVIATIONS	xv
CHAPTER 1 INTRODUCTION	1
1.1 Background	1
1.2 Problem Statement	5
1.3 Research Aim and Objectives	10
1.4 Scope of Work	10
1.5 Thesis Contribution	11
1.6 Organization of Thesis	12
CHAPTER 2 LITERATURE REVIEW	13
2.1 Overview	13

2.2	Hybrid Model for IDS	14
2.3	Fuzzy Logic-based IDS	24
2.3.1	Fuzzy System and Uncertainty	26
2.3.2	Fuzzy Logic-based IDS	32
2.3.3	Hybrid Model with Fuzzy Logic	35
2.4	Neural Network-based IDS	37
2.4.1	Deep Neural Network	39
2.4.2	IDS Based on Neural Networks and Deep Learning	48
2.5	Hybrid Fuzzy with Neural networks	53
2.5.1	Classification of Hybrid Fuzzy with Neural networks	54
2.5.2	ANFIS	56
2.5.3	CART	62
2.5.4	Other Hybrid Fuzzy with Neural Networks	65
2.6	Chapter Summary	69
CHAPTER 3	ENVIRONMENT AND EXPERIMENTAL DESIGN	72
3.1	Overview	72
3.2	Experimental Approaches	72
3.3	Dataset Analysis	77
3.4	Experimental Environment	84
3.5	Evaluation Indicators	85

3.6	Chapter Summary	87
CHAPTER 4 DEEPER AND INTERPRETABLE ANFIS FOR INTRUSION DETECTION		88
4.1	Overview	88
4.2	Pre-processing Techniques	88
4.3	Architecture of Model	91
4.4	Experimental Results	100
4.4.1	Pre-processing Results	101
4.4.2	Results of ANFIS-CART	106
4.5	Interpret The Intrusion Patterns	108
4.6	Discussion	113
4.7	Chapter Summary	116
CHAPTER 5 HYBRID DEEP FUZZY NEURAL NETWORK		118
5.1	Overview	118
5.2	Architecture of Model	119
5.3	Experimental Results	127
5.4	Comparative Analysis	129
5.5	Discussion	132
5.6	Chapter Summary	135
CHAPTER 6 CONCLUSIONS AND RECOMMENDATIONS		136

6.1	Conclusion	136
6.2	Future Works	138
	REFERENCES	142
	APPENDICES	159

LIST OF TABLES

	Page
Table 2.1: Machine Learning Approaches in IDS	17
Table 2.2: Four Types of MFs	28
Table 2.3: Three Types of Fuzzy Inference	30
Table 2.4: Recent Literature of NN in IDS	49
Table 2.5: Two Passes in The Hybrid Learning Algorithm for ANFIS	60
Table 3.1: Distribution of Major Categories in KDDTrain+ and KDDTest+ Subsets	79
Table 3.2: Basic Features	80
Table 3.3: Content Features	81
Table 3.4: Traffic Features Based on Time (Two-second Time Window)	81
Table 3.5: Traffic Features Based on Host (100-connection Window)	82
Table 3.6: Language & Framework of Experiment	84
Table 3.7: Four Situations of Results	85
Table 4.1: Correlation Degree of 41 Attributes towards Target	102
Table 4.2: The Standard Deviation and Original Interval Number of Attributes	104
Table 4.3: Final Interval Number of Selected Continuous Attributes	105
Table 4.4: Performance of Proposed Method on KDDTrain+ Dataset	107
Table 4.5: Performance of Proposed Method on KDDTest+ Dataset	108
Table 4.6: The Classification Pattern of The Nodes	111
Table 4.7: Comparison of Detection Rate and FAR on KDDTrain+ Dataset	114
Table 4.8: Comparison of Accuracy on KDDTrain+ Dataset	114
Table 4.9: Comparison of CART, ANFIS, and Proposed Method on KDDTest+ Dataset	114
Table 4.10: Comparison of Accuracy with Other Existing Approaches on KDDTrain+ Dataset	115

Table 5.1:	Experimental Results for Different Performance Metrics in Five Categories of The KDDTrain+ Dataset	129
Table 5.2:	Experimental Results for Different Performance Metrics in Five Categories of The KDDTest+ Dataset	129
Table 5.3:	Detection Rate, ACC, and FAR Comparison of Different Classifiers on KDDTrain+	130
Table 5.4:	ACC Comparison of Different Classifiers on KDDTrain+	131

LIST OF FIGURES

	Page
Figure 1.1: Intrusion Behaviours and Normal Behaviours	3
Figure 2.1: If-Then Rule	30
Figure 2.2: Classification of Fuzzy IDSs	34
Figure 2.3: Single Artificial Neural Network	38
Figure 2.4: Development of Deep Learning	41
Figure 2.5: Deep learning models	42
Figure 2.6: The Structure of ResNet18	46
Figure 2.7: The Neural-Fuzzy Model	55
Figure 2.8: The Fuzzy-Neural Model	55
Figure 2.9: The Combination of Neural and Fuzzy Model	56
Figure 2.10: Architecture of ANFIS	58
Figure 3.1: Processes in Conducting the Experiment	73
Figure 3.2: Idea of The First Improvement	74
Figure 3.3: Idea of The Second Improvement	76
Figure 3.4: Part of Correlation Matrix of 43 Attributes	83
Figure 4.1: Adaptive K-means Algorithm	90
Figure 4.2: Architecture of ANFIS-CART	93
Figure 4.3: Overlapped Fuzzy Intervals	94
Figure 4.4: The Node for Interpreting the Pattern	109
Figure 5.1: The Architecture of The Proposed Method	121
Figure 5.2: CrossEntropy Convergence Curve of the Proposed Method	128
Figure 5.3: Evaluation Results of ResNet on NSL-KDD Dataset	132

LIST OF ABBREVIATIONS

ABC	Artificial Bee Colony
ACC	Accuracy
AI	Artificial Intelligence
ANFIS	Adaptive Network-based Fuzzy Inference System
ANN	Artificial Neural Network
BP	Back Propagation
BPNN	Back Propagation Neural Network
CART	Classification and Regression Tree
CGS	Centre for Graduate Studies
CNCERT	National Internet Emergency Centre of China
CNN	Convolutional Neural network
DBN	Deep Belief Network
DDoS	Distributed Denial of Service
DNN	Deep Neural Network
DOS	Denial-Of-Service
DT	Decision Tree
FAR	False Alarm Rate
FN	False Negative
FP	False Positive
GA	Generic Algorithm
GAN	Generative Adversarial Network
GBDT	Gradient Boosting Decision Tree
GNN	Graph Neural Network

GoogleNet	GoogleInceptionNet
HIDS	Host-based IDS
HMM	Hidden Markov Model
IDS	Intrusion Detection System
IoT	Internet of Things
K-means	K-means Clustering Algorithm
KNN	K-nearest Neighbour
LAN	Local Area Network
LSTM	Long Short-Term Memory
MF	Membership Function
NIDS	Network-based IDS
NN	Neural Network
NSL-KDD	National Security Lab–Knowledge Discovery and Data Mining
PCA	Principal component analysis
PROBE	Surveillance or Probe
R2L	Remote to Local
RBM	Restricted Boltzmann Machine
ResNet	Residual Network
RNN	Recurrent Neural Network
SAE	Stacked Autoencoder
SOM	Self-Organizing Map
SVM	Support Vector Machine
TN	True Negative
TP	True Positive
U2R	User to Root
UNIMAS	Universiti Malaysia Sarawak

VGG	Visual Geometry Group
WDLSTM	Weight-Dropped Long Short-term Memory
XGBoost	Extreme Gradient Boosting

CHAPTER 1

INTRODUCTION

1.1 Background

The security of the network is becoming increasingly precarious. The Cisco Annual Internet Report (2018–2023) White Paper (Cisco, 2020) reveals a concerning trend of increasing network intrusions. Specifically, there has been a massive 776% surge in attacks with data volumes ranging from 100 Gbps to 400 Gbps between 2018 and 2019. This indicates a clear upward trend in the frequency and severity of cyberattacks (Cisco, 2020).

Furthermore, the report indicates that more than half of the surveyed network operators have encountered infrastructure outages. This highlights the significant impact of cyberattacks on the stability and reliability of network infrastructures, posing serious challenges to the cybersecurity landscape. These outages can have detrimental effects on businesses and individuals, leading to financial losses, service disruptions, and compromised data security.

Given the escalating threat landscape, it is crucial for organizations to prioritize network security measures and implement robust intrusion detection systems. Proactive monitoring, timely threat intelligence, and continuous security updates are vital to safeguarding networks from potential breaches and minimizing the impact of cyberattacks.

In China, the network security scenario is also of great concern. The 50th Statistical Report on China's Internet Development (CINIC, 2022) reveals that the National Internet Emergency Centre of China (CNCERT) received 15,654 reports of network security

incidents in the first half of 2022. Furthermore, China Telecom, China Mobile, and China Unicom detected a total of 16,542 DDoS attacks in the same period.

The rapid progress of technology has presented us with numerous opportunities, but it has also posed significant challenges in the domain of network security. The emergence and widespread adoption of technologies like AI (Artificial Intelligence), e-commerce, social media, machine learning, IoT (Internet of Things), cloud computing, Big Data, and mobile payments have undoubtedly fuelled economic growth. However, alongside these advancements, there has been a corresponding rise in the number and sophistication of cyberattacks (Cisco, 2020).

Cybercriminals have leveraged the potential vulnerabilities brought about by these technologies to launch attacks that are increasingly covert and intelligent. This has rendered traditional intrusion detection methods less effective in identifying and mitigating such attacks. Given the increasing complexity and sophistication of cyber threats, there is an urgent requirement to develop and implement more advanced and adaptive security measures to protect our networks and sensitive data (imarc, 2022).

Consequently, the study of new techniques for intrusion detection has become of paramount importance. Intrusion detection, being a crucial component of network security, has assumed a more critical role in the evolving network landscape. It is imperative that further research and development be carried out in this field to ensure the security of networks against potential threats.

However, intrusion detection is a fuzzy classification problem that involves uncertainty in the classification process. The behaviour of intruders in network security often exhibits similarities to that of legitimate users, as illustrated in Figure 1.1. This overlap

results in inherent fuzziness and uncertainty in intrusion detection. Many traditional machine learning methods are not well equipped to handle this uncertainty (Tsang et al., 2007).

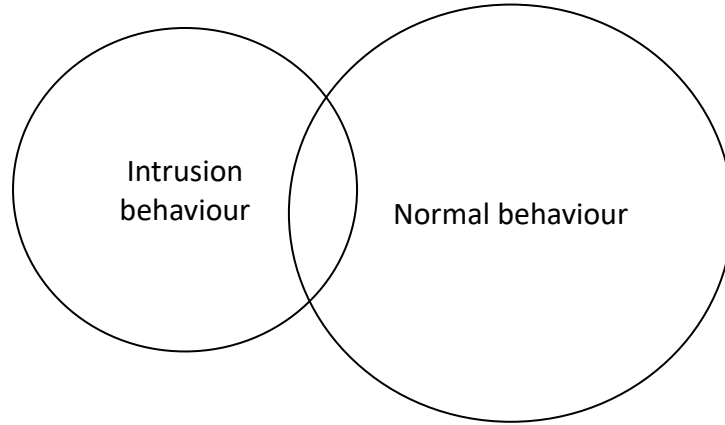


Figure 1.1: Intrusion Behaviours and Normal Behaviours

Since the introduction of fuzzy logic theory by Lotfi Zadeh in 1965, it has been regarded as a powerful approach to address problems involving uncertainty (Masdari & Khezri, 2020). Fuzzy logic is a method that uses fuzzy sets to represent linguistic variables and allows for the specification of intermediate values. Fuzzy logic-based techniques have gained significant popularity and widespread usage in the development of intrusion detection systems., such as fuzzy clustering, fuzzy classifiers, fuzzy feature extraction, and others. The effectiveness of these methods in detecting intrusions in the presence of uncertain data has been demonstrated by researchers (Masdari & Khezri, 2020). While fuzzy logic provides a powerful framework for modelling and handling uncertainty, it is still limited by its inability to fully capture the complex and dynamic nature of the intrusion detection problem. This highlights the need for further research to improve the application of fuzzy logic in intrusion detection, to enhance its ability to handle the uncertainties and ambiguities of the problem and ultimately improve its performance (Alsaadi et al., 2022).

Recently, Neural Network (NN) algorithms have gained considerable attention due to their superior performance in image classification tasks, leading to the development of various NN-based intrusion detection systems (Drewek-Ossowicka et al., 2021). These systems leverage the deep feature extraction abilities of NNs, enabling them to cope with the complex and dynamic network environment. However, the issue of overfitting, which occurs when the model is too complex and fits the training data too closely, resulting in poor generalization performance on unseen data, remains prevalent in NN-based intrusion detection systems applied to fuzzy classification problems. In particular, overfitting can cause the model to overlook important patterns in the data, which are crucial for accurate classification, making it a challenging problem that must be addressed to ensure robust and effective performance (Drewek-Ossowicka et al., 2021).

To address this issue, scholars have proposed various solutions. Some of the most popular methods include regularization techniques such as dropout and early stopping, as well as model architecture modifications such as pruning and ensemble methods (Piotrowski et al., 2020). However, intrusion detection is a fuzzy classification problem. To better fit the fuzzy nature of this problem, the combination of fuzzy logic and neural networks is an inevitable choice (Alsaadi et al., 2022).

Fuzzy schemes have proven to be effective in detecting intrusions and malicious behaviors when dealing with uncertain data (Masdari & Khezri, 2020). Fuzzy logic has emerged as a popular solution for addressing uncertainty in intrusion detection and has demonstrated its effectiveness in this field. Among the different fuzzy approaches applied in intrusion detection, the ANFIS classifier has gained widespread adoption (Masdari & Khezri, 2020).

By combining the benefits of fuzzy logic and neural networks, Adaptive Neuro-fuzzy Inference System (ANFIS) is designed to provide an efficient solution to the problems of uncertainty in intrusion detection (Jang, 1993). On the KDDCUP99 dataset, the use of ANFIS in intrusion detection systems was first reported by Toosi and Kahani (Toosi & Kahani, 2007), who achieved a 95.3% detection rate on the KDDCUP99 dataset using five ANFIS modules. While ANFIS demonstrates high performance in fuzzy classification tasks, it does have certain limitations that warrant further investigation in future research. One of the key limitations is the lack of deep feature recognition capability (T.-L. Nguyen, 2023).

1.2 Problem Statement

Intrusion detection is a security measure designed to identify intrusion behaviors. These detection problems are characterized by a degree of uncertainty in the classification process. Often, the behaviors of intruders overlap with those of legitimate users, leading to inherent uncertainty in intrusion detection. This uncertainty poses a significant challenge for many traditional machine learning methods, as they may struggle to accurately differentiate between intruder and legitimate user behaviors in the presence of such overlap (Tsang et al., 2007).

It is crucial to develop algorithms that can effectively handle this uncertainty in intrusion detection. Fuzzy logic has been shown to be a promising approach (Masdari & Khezri, 2020). In various fuzzy-based approaches, ANFIS is one of the most effective and widely used algorithms for intrusion detection (Masdari & Khezri, 2020).

ANFIS, which stands for Adaptive-Network-based Fuzzy Inference System, is a method that combines fuzzy logic and neural network models to create a powerful tool for intrusion detection. ANFIS's capability to generate fuzzy and overlapping fuzzy rules,

coupled with parameter learning methods akin to neural networks, provides a robust solution for handling uncertainty (Jang, 1993). ANFIS effectively manages the uncertainty inherent in intrusion detection problems, making it a powerful and reliable choice for addressing such challenges.

However, there are some limitations to ANFIS in practical use. One of them is the limited number of layers, which may not be sufficient for deep pattern extraction (T.-L. Nguyen, 2023). Another limitation of ANFIS is the high computational cost associated with generating a large number of rules, which can make it challenging to train the model with high-dimensional data efficiently.

Therefore, the research questions of this study can be summarized as follows:

- i. Feature selection and extraction: How to automatically select and extract effective features for intrusion detection using ANFIS? How to handle high-dimensional feature spaces and utilize feature selection methods to improve detection accuracy?
- ii. Rule generation and optimization: How to automatically generate fuzzy rules for ANFIS and optimize them to enhance model performance? How to handle rule redundancy in ANFIS?
- iii. Deep pattern recognition: How to effectively detect and recognize complex, deep patterns in intrusion detection using ANFIS, considering the challenges of high-dimensional data and complex attack behaviors? How to incorporate other techniques into ANFIS to enhance its ability to detect and interpret deep patterns?

- iv. Interpretability: How to improve the Interpretability of ANFIS in intrusion detection and make the decision-making process more transparent? How to interpret the decisions made by model and evaluate its reliability?
- v. Integration with other technologies: How to integrate ANFIS with other technologies such as decision tree or deep learning to leverage their respective strengths to enhance intrusion detection performance?

These research questions provide guidance for the application of ANFIS in intrusion detection and offer ideas for developing more effective and robust intrusion detection systems.

To address the challenges of deep pattern recognition and model interpretability, the integration of ANFIS and decision tree (DT) can offer significant advantages. By combining these two techniques, we can leverage the complementary strengths of each approach to enhance both the depth of pattern extraction and the interpretability of the model. ANFIS can handle complex and non-linear relationships between input and output variables, making it effective for modeling problems with uncertainty and fuzziness. On the other hand, Decision Trees (DT) are tree-like models used for classification and regression tasks. They operate by recursively partitioning the data space into smaller subspaces, guided by simple decision rules based on feature values. The primary goal of a decision tree is to create a tree structure where each internal node represents a decision based on a specific feature, and each leaf node represents the final classification or regression output. By combining the strengths of ANFIS and DT, the resulting model can provide more accurate and deep pattern extraction, while also maintaining interpretability, which is important for practical use in many fields.

However, DT can be effective for classification tasks, but they may not be sufficient for intrusion detection tasks that involve complex, high-dimensional data (Blockeel et al., 2023). In contrast, deep learning algorithms can handle complex and high-dimensional data, and are capable of extracting deep patterns that are not readily discernible by humans (He et al., 2015). Therefore, to achieve deeper pattern extraction, the combination of ANFIS and deep learning is a viable option. This integration can leverage the complementary strengths of both techniques to enhance intrusion detection performance.

In deep pattern extraction, the term "deep" refers to the capacity to extract complex patterns through a deep architecture with numerous layers and millions of parameters. In some cases, if the pattern extraction is not deep, one model may not be able to capture complex patterns and relationships in the data. The expected outcome of a deep pattern extraction is to improve the accuracy and performance of the model by extracting more abstract and representative patterns from the data. In terms of achieving this capability, deep learning algorithms are known for their ability to extract deep and complex patterns from high-dimensional data. This is made possible by the use of neural networks with many layers, which allow for the creation of hierarchical representations of the input data (He et al., 2015). However, the overfitting problem of deep learning in fuzzy classification problems, such as intrusion detection, needs further research (Alsaadi et al., 2022).

For deeper pattern extraction, the hybrid model, which combines ANFIS and deep learning, aims to synergize the strengths of both approaches. Deep learning is renowned for its capability to extract patterns from high-dimensional data and learn intricate patterns, while ANFIS offers a fuzzy rule-based system that incorporates fuzziness. By integrating the two approaches, the hybrid model has the potential to achieve deeper pattern extraction

ability and improve the accuracy of the resulting model for fuzzy classification problems, thus making it more suitable for increasingly complex network environments.

The primary objective of this research is to enhance the performance of ANFIS in fuzzy classification tasks, particularly in the realm of intrusion detection, through the integration of ANFIS with alternative methods. The researchers aim to overcome the limitations inherent in ANFIS, which include inadequate pattern extraction and the generation of excessive redundant rules. By leveraging the advantages of diverse technologies, this study hopes to improve the performance and interpretability of the model, thus better tackling complex intrusion detection challenges.

ANFIS can generate a large number of rules, particularly when dealing with datasets with many input features or complex patterns. This rule explosion can make the model difficult to interpret and lead to overfitting on the training data. By combining correlation analysis, feature selection techniques, and other methods, the generation of rules can be reduced, overfitting can be mitigated, and the efficiency and usability of the algorithm can be improved.

Furthermore, the limited depth of ANFIS can hinder its ability to capture complex relationships in the data. By integrating ANFIS with other techniques that allow for deeper pattern extraction, such as DT and ResNet, the proposed method seeks to enhance the model's capacity to learn intricate patterns and representations, resulting in enhanced classification accuracy.

Overall, the integration of ANFIS with other advanced techniques is expected to provide a more effective and robust solution for fuzzy classification tasks, especially in the context of intrusion detection. By leveraging the strengths of each method and addressing

their respective limitations, the proposed approach aims to achieve better performance in terms of classification accuracy, while also ensuring interpretability and generalization ability.

1.3 Research Aim and Objectives

The purpose of this study is to enhance fuzzy classification techniques in intrusion detection through the development of hybrid methods. In terms of implementation, the hybrid model is used to improve the weaknesses of the ANFIS model, enhance its ability to recognize deep fuzzy patterns, and retain some of the advantages of ANFIS.

The objectives of the study are as follows:

- i. Optimize the ANFIS model to improve its problem of generating too many rules, making it more suitable for deep hybrid models.
- ii. Explore deep ANFIS hybrid models while retaining the fuzzy and interpretable characteristics of the model. Conduct experiments on the model and evaluate and improve its performance.
- iii. Explore highly deeper ANFIS hybrid models to identify deeper intrusion detection patterns, preserving some of the fuzzy characteristics of the model. Conduct experiments on the model and evaluate its performance.

1.4 Scope of Work

The scope of the study includes the investigation of hybrid techniques to enhance the performance of the ANFIS in fuzzy classification problems, particularly in the context of intrusion detection. The objective of this study is to explore the use of other machine learning

methods to deepen the structure of ANFIS for better deep feature extraction and to improve its fuzzy classification capabilities.

The experimental evaluation will be conducted on the publicly available dataset NSL-KDD (National Security Lab-Knowledge Discovery and Data Mining). The study will focus on reducing the number of rules generated by ANFIS to increase its training efficiency and make it more practical for intrusion detection. The research will also examine the effectiveness of DT and deep learning in combination with ANFIS to enhance its deep pattern extraction capabilities. The performance of the proposed model will be compared with other state-of-the-art techniques, such as FC-ANN, CSO-ANFIS, and GA-ANFIS. Finally, the objective of this study is to gain insights into the performance of the proposed hybrid models and to identify their strengths and limitations.

1.5 Thesis Contribution

Based on the research objectives and scope, the primary contribution of this thesis will be the development of a novel hybrid ANFIS fuzzy classification model for intrusion detection. The proposed model will aim to reduce the number of generated rules by ANFIS, increase its training efficiency, and enhance its deep pattern extraction capabilities by leveraging other machine learning techniques.

The experimental evaluation of the proposed model on the NSL-KDD dataset will provide insights into its performance in comparison to other state-of-the-art techniques. This study's findings will contribute to the advancement of fuzzy classification techniques in intrusion detection and shed light on the potential benefits of hybrid models for improving the accuracy and efficiency of intrusion detection systems.

The study will also contribute to the broader field of machine learning and artificial intelligence by providing insights into the effectiveness of hybrid models for fuzzy classification problems. Finally, the study aims to identify the strengths and limitations of the proposed models, which can inform future research in this area.

1.6 Organization of Thesis

The first chapter provides an overview of the research topic and relevant concepts, as well as outlining the objectives of the thesis. The second chapter conducts a comprehensive literature review of recent developments in fuzzy logic and neural network-based intrusion detection systems. The third chapter focuses on introducing the experimental environment and experimental design. The fourth chapter details the proposed ANFIS-CART (Classification and Regression Tree) scheme and presents the experimental results. The fifth chapter introduces the hybrid ANFIS-ResNet scheme and provides a comprehensive evaluation of its performance. Finally, the last chapter summarizes the findings and contributions of this study, and suggests directions for future research.

CHAPTER 2

LITERATURE REVIEW

2.1 Overview

In the present information era, the network has greatly facilitated communication, connecting individuals, organizations, and countries. An online information society has been established, where communication between people is not limited by distance, and the growth of network data has been explosive. As reported by Internet World Statistics (Jazara, 2009), by June 30, 2022, the number of internet users had reached 5,473,055,736, accounting for 64.2% of the world's total population. The internet has become an essential and integral part of modern society.

However, with the advancement of technology and the expansion of networks, cyber attacks have become increasingly prevalent in recent years (Cisco, 2020). The consequences of these attacks can be severe, leading to loss of privacy and economic damage for individual internet users, as well as posing a risk to corporate offices, banks, hospitals, governments, and millions of organizations through the exposure of sensitive information and potential disruption of online services. In response to the increasing threat of cyberattacks and unauthorized access to computer systems and networks, intrusion detection systems (IDSs) have been developed to safeguard against such malicious activities.

Machine learning algorithms have shown great potential in intrusion detection tasks. However, due to the uncertainty and vagueness of intrusion behaviours, many machine learning methods face great challenges in achieving high accuracy and good performance in intrusion detection. In order to address this problem, fuzzy logic has been widely used in

intrusion detection systems, and it has achieved good results. Among various fuzzy approaches applied in intrusion detection, the ANFIS classifier, as a hybrid model that effectively combines the strengths of fuzzy logic and neural networks, has gained widespread acceptance (Masdari & Khezri, 2020). However, ANFIS also has limitations in high-dimensional data analysis and deep pattern extraction.

This study focuses on recent developments in fuzzy logic and neural network-based intrusion detection systems (IDSs). The purpose of the literature review is to provide an overview of the state-of-the-art in fuzzy logic-based, neural network-based and hybrid IDSs, including the strengths and limitations of the different approaches. The literature review also provides a foundation for the proposed hybrid ANFIS schemes in this study. The review highlights the importance of incorporating both fuzzy logic and neural network techniques in IDSs to achieve high accuracy and robustness in detecting network intrusions.

2.2 Hybrid Model for IDS

As the integration of new technologies like 5G, cloud computing, big data, artificial intelligence, blockchain, and IoT technology continues to advance, the threat of cyber security risks is growing at an unprecedented pace and the task of intrusion detection becomes increasingly complex. On the one hand, these new technologies offer new avenues for potential intrusions, while on the other hand, intrusion detection systems must adapt to keep pace with this rapidly changing landscape.

IDSs can be classified into two categories: signature-based and anomaly-based. Signature-based IDSs use pre-defined patterns to identify known attacks, while anomaly-based IDSs use machine learning techniques to identify intrusions based on the differences between normal and abnormal behaviour patterns.

Machine learning algorithms have been applied to intrusion detection in various ways, such as feature selection, classification, clustering, and anomaly detection. Feature selection algorithms are used to select the most relevant features from a large amount of data for intrusion detection, which can improve the accuracy of intrusion detection. Classification algorithms are used to classify normal and abnormal behaviours, while clustering algorithms group similar behaviours together. Anomaly detection algorithms, on the other hand, detect unusual behaviours that deviate from normal patterns.

Moreover, with the development of neural networks, especially deep learning, deep neural networks (DNNs) have been introduced into intrusion detection, and have achieved better performance than traditional machine learning algorithms in some cases. DNNs can automatically learn complex feature representations from raw data and perform end-to-end intrusion detection without manual feature engineering.

Despite the progress made in intrusion detection with machine learning, there are still challenges that need to be addressed, such as the high false positive rate, the difficulty in generalizing to new types of intrusions, and the lack of interpretability of machine learning models. To overcome these challenges, hybrid approaches that combine multiple machine learning algorithms have been proposed, and have shown promising results in improving the performance of intrusion detection.

Camastra et al. (2013) suggested that the application of machine learning and software computing techniques in intrusion detection can be divided into four categories: supervised learning-based approaches, unsupervised learning-based approaches, statistical modeling-based approaches, and ensemble-based approaches, as detailed below.

- i. Supervised learning-based approaches: This category of approaches focuses on identifying known intrusions more intelligently. The most common methods used in this category are NNs and support vector machines (SVM). These approaches have been effective in detecting known intrusions, but they may not be as effective in detecting unknown intrusions.
- ii. Unsupervised learning-based approaches: Unsupervised learning methods are used to detect unknown intrusions by detecting suspicious activities. Some of the commonly used methods in this category include Self-Organizing Maps (SOMs), rough sets, fuzzy logic, modified SVM, and modified NN. These methods have the advantage of detecting both known and unknown intrusions, but their accuracy may be lower than supervised learning methods.
- iii. Statistical modeling-based approaches: This category of methods uses statistical techniques to monitor user behaviour and identify anomalies within the system usage policies. Bayesian algorithms and Hidden Markov Models (HMMs) are the most commonly used methods in this category. These approaches have been effective in detecting known intrusions, but they may not be as effective in detecting unknown intrusions.
- iv. Ensemble-based approaches: This category of approaches combines the best models from each class to maximize computational efficiency and detection accuracy. Instead of relying on a single model, the ensemble-based approach selects the best model for each class and combines them to achieve better results. These approaches have been shown to be effective in detecting both

known and unknown intrusions, but they may require more computational resources than other methods.

The application of individual and hybrid models is shown in Table 2.1.

Table 2.1: Machine Learning Approaches in IDS

ML	Authors	ML Approach	Datasets	Key Findings
SVM	Kim and Park (2003)	SVM	KDD 99	SVM in IDS
	Mukkamala et al. (2005)	Ensemble of ANNs, SVMs and MARs	DARPA98	NNs, SVMs and MARs in IDS
	Khan et al. (2007)	CT, SVM with hierarchical clustering (DGSOT)	DARPA98	Combination of SVM and DGSOT Compared with Rocchio Bundling and random selection
	Amiri et al. (2011)	MMIFS with LSSVM	KDD 99	Modified mutual information-based feature selection algorithm and linear correlation-based feature selection algorithm for feature selection, linear and non-linear Compared with mRMR and CMIM
	Horng et al. (2011)	Hierarchical Clustering and SVM	KDD 99	BIRCH hierarchical clustering algorithm for reducing data Feature selection based on “leave-one-out” Compared with ESC-IDS, KDD 99 winner, KDD99 runner-up, Multi-classifier, Association rule
	Lin et al. (2012)	SVM, DT, SA	KDD 99	SVM and SA for feature selection DT and SA for finding decision rules for new attacks The parameters for DT and SVM are adjusted by SA Compared with the hybrid process of DT, SA and feature selection, the hybrid process of SVM, PSO and feature selection, the hybrid process of SVM, SA and feature selection, SVM, DT
	Li et al. (2012)	K-means, ACO, SVM	KDD 99	Improve the efficiency of intrusion detection Feature removal method ACO for training data selection Combination of clustering, colony, SVM
	Chandrasekhar and Raghuvver (2013)	K-means, Neuro Fuzzy and SVM classifier	KDD 99	K-means for generating different training subsets Different neuro-fuzzy models are trained in these subsets SVM for detecting intrusion Compared with KDD 99 Winner, Multi-class SVM, Layered conditional random fields, Columbia Model, DT, BPNN
	Kim et al. (2014)	C4.5 and one class SVM	NSL KDD 99	C4.5 for building subsets SVM is set for each subset Compared with two Conventional hybrid methods, DT, 1-class SVM
	Kuang et al. (2014)	KPCA and SVM with GA (N-KPCA-GA-SVM)	KDD 99	KPCA for reducing dimension and shortening training time SVM for estimating intrusion N-RBF for reducing noise GA for optimizing SVM Compared with KPCA-GA-SVM, PCA-GA-SVM, Single-SVM, RBFNN

Table 2.1 continued

SVM	Feng et al. (2014)	CSVAC (SVM+CSOACN)	KDD 99	Combination of SVM and CSOACN
	Al-Qatf et al. (2018)	STL, SVM	NSL-KDD	Unsupervised feature learning Self-taught learning Compared with J48, naive Bayesian, random forest, and SVM
	Pu et al. (2021)	Sub-Space Clustering (SSC) and One Class Support Vector Machine (OCSVM)	NSL-KDD	Unsupervised method Compared with SSC_EA, DBSCAN, K-means
	Kalita et al. (2023)	SVM, MFO	NSL-KDD	Non-stationary environment SVM of optimizing hyperparameters dynamically for intrusion detection MFO as the base optimization algorithm
DT	Amor et al. (2004)	NB vs DT	KDD 99	Compare the performance between NB and DT
	Xiang et al. (2008)	Bayesian clustering, C4.5	KDD 99	Combination of C4.5 and Bayesian clustering
	Sangkatsanee et al. (2011)	C4.5	RLS09 (self) KDD 99	DT outperform the other techniques Compared with Ripper Rule, NN, Bayesian Network, Naïve Bayes, Radial Basis Function NN
	Lin et al. (2012)	SVM, DT, SA	KDD 99	SVM and SA for feature selection DT and SA for finding decision rules for new attacks The parameters for DT and SVM are adjusted by SA Compared with the hybrid process of DT, SA and feature selection, the hybrid process of SVM, PSO and feature selection, the hybrid process of SVM, SA and feature selection, SVM, DT
	Sivatha et al. (2012)	Wrapper (GA, Neuro tree and C4.5)	KDD 99	GA for feature selection Combination of neurotree and C4.5 for classification Compared with Decision Stump, C4.5, Naïve Bayes, Random Forest, Random Tree, REP Tree
	Elhag et al. (2015)	GFS, Fuzzy association rule (FARC-HD)	KDD 99	First combination of GFS and OVO for IDS Genetic Fuzzy Systems for the smoother borderline between the concepts and improving precision for rare attack events Compared with C4.5, FARCHD, MOGFIDS, GFS-GCCL, GFS-Pitts, Boost-FAR
	Shone et al. (2018)	Non-symmetric deep AE random forest	NSL-KDD KDD 99	Based on stacked AE and Random forest Compared with DBN
	Aljawarneh et al. (2018)	Vote algorithm, J48, Meta pagging, Random tree, REPTree, AdaBoostML, Decision stump, Naïve Bayes	NSL-KDD	Variety of classifiers: J48, Meta pagging, Random tree, REPTree, AdaBoostML, Decision stump, Naïve Bayes Comparison of the presented mechanism with respect to possible attack types Binary and multiclass classification
	Bhattacharya et al. (2020)	PCA-firefly, XGBoost	Kaggle intrusion dataset	PCA-firefly for reducing the dimensionality Compared with KNN, naïve Bayes, random forest, SVM, and XGBoost
	Li et al. (2020)	AE random forest, AP	CSE-CIC-IDS 2018	AP feature grouping Compared with KitNet
	Li et al. (2022)	Bayesian possibilistic clustering, fuzzy decision trees	Dataset collected from real IoT systems	Bayesian possibilistic clustering module for feature processing An ensemble module motivated by reinforcement learning and adaptive boosting that dynamically fits the streaming data A collection of fuzzy decision trees that provide readable decision boundaries IoT

Table 2.1 continued

DT	Louk and Tama (2023)	Combination of bagging and GBDT	NSL-KDD UNSW-NB15 HIKARI-2021	Bagging and bootstrap replicates to generate different training instances GBDT to be trained on each sample Compared with GBM, LightGBM, CatBoost, and XGBoost
NB	Koc et al. (2012)	Hidden Naïve Bayes (HNB)	KDD 99	Naïve Bayes is extended by using directed arcs to represent attribute dependencies as a Bayesian network model Compared with JPip, NBTree, LBk, SVM, J48, MLP, Decision Table, SMO, BayesNet, OneR, Naïve Bayes
	Aljawarneh et al. (2018)	Vote algorithm, J48, Meta pagging, Random tree, REPTree, AdaBoostML, Decision stump, Naïve Bayes	NSL-KDD	Variety of classifiers: J48, Meta pagging, Random tree, REPTree, AdaBoostML, Decision stump, Naïve Bayes Comparison of the presented mechanism with respect to possible attack types Binary and multiclass classification
KNN	Li et al. (2007)	Active learning based TCM-KNN	KDD 99	High detection rate, low false positive using fewer data
	Lin et al. (2015)	CANN with KNN	KDD 99	CANN for feature representation Two distances are measured and used for representing each sample by KNN Compared with KNN and SVM
	Pajouh et al. (2019)	Component analysis, linear discriminate analysis, naïve bayes, CF-KNN	NSL-KDD	Component analysis and linear discriminate analysis for reducing the dimension Naïve Bayes for classifying anomaly, then use CF-KNN to refine to normal instances Compared with Two-tier, SVM with BIRCH, ESC-IDS, Association rule IDS, HFR-MLR, Fuzzy classification, SVM, decision tree IoT environment
Fuzzy logic	Wang et al. (2007)	Rough set theory, association rule	KDD 99	Data pre-processed by RST, then discovering association rules in RST framework Compared with best results of KDD 99 contest
	Toosi and Kahani (2007)	Neural Fuzzy & GA (ESC-IDS)	KDD 99	GA to optimize the structure of fuzzy decision engine
	Tajbakhsh et al. (2009)	Fuzzy Association rules	KDD 99	Fuzzy association rules for building classifiers Association hyper-edge for item reduction Compared with clustering, KNN, SVM, H-SOM
	Wang et al. (2010)	FC-ANN	KDD 99	Combination of ANN and fuzzy clustering Solve the detection stability of ANN Fuzzy clustering for generating different training subsets ANN for formulating different base models Fuzzy aggregation for aggregate the results Compared with DT, naïve bayes, BPNN
	Boughaci et al. (2012)	FPSO	KDD 99	Fuzzy rule improved by PSO Compared with FGA, FLS, Hybrid EFS, C4.5, 5-NN, EFRID, NB, Naïve Bayes
	Lin et al. (2022)	Fuzzy neural network blockchain	NS3 Simulator system	Fuzzy neural network to detect packets with possible malicious content SDN and block chain for detecting threats and attacks IoT
	Savva et al. (2022)	Fuzzy logic controller	Contiki OS and the Cooja simulator tool	Jamming intrusion detection Jamming Index (JI) as the system's output IoT

Table 2.1 continued

Fuzzy logic	Saridou et al. (2022)	Fuzzy Set theory and the H-indexing space filling curve	5000 malicious and benignfiles of different sizes	Signature-Agnostic IDS SAGMAD, a binary visualisation methodology based on the pre-existing binvis SAGMAD leverages an existing colour scheme to optimise its capabilities for more effective pattern recognition
	Xu et al. (2022)	Graphs and logic fuzzy	Real data Austin Texas	Fuzzy Optimum-Path Forest (Fuzzy OPF) Fuzzy OPF for threat escapes the regular traffic of an IoT network IoT
	Li et al. (2022)	Bayesian possibilistic clustering, fuzzy decision trees	Dataset collected from real IoT systems	Bayesian possibilistic clustering module for feature processing An ensemble module motivated by reinforcement learning and adaptive boosting that dynamically fits the streaming data A collection of fuzzy decision trees that provide readable decision boundaries IoT
	Alrayes et al.(2023)	ANFIS, HGSO	N-BaIoT	Metaheuristics feature selection with fuzzy logic enabled intrusion detection system HGSO for feature selection ANFIS for classification BBA for adjusting parameter in ANFIS IoT
GA	Chen et al. (2007)	FBT with PSO and GA	DARPA98	Allow input variables selection, overlayer connections and different activation functions FNT is developed using an evolutionary algorithm Parameters are optimized by swarm optimization algorithm
	Toosi and Kahani (2007)	Neural Fuzzy & GA (ESC-IDS)	KDD 99	GA to optimize the structure of fuzzy decision engine
	Sivatha et al. (2012)	Wrapper (GA, Neuro tree and C4.5)	KDD 99	GA for feature selection Combination of neurotree and C4.5 for classification Compared with Decision Stump, C4.5, Naïve Bayes, Random Forest, Random Tree, REP Tree
	Kuang et al. (2014)	KPCA and SVM with GA (N-KPCA-GA-SVM)	KDD 99	KPCA for reducing dimension and shortening training time SVM for estimating intrusion N-RBF for reducing noise GA for optimizing SVM Compared with KPCA-GA-SVM, PCA-GA-SVM, Single-SVM, RBFNN
	Kumar and Kumar (2015)	Three phases MOGA	KDD 99 ISCX	Generate a pool of non-inferior solutions First phase, approximate a Pareto front of non-inferior individual solutions Second phase, the entire solution set is further refined and approximate another Pareto front of ensembled solutions Third phase, voting for fusing individual solutions
	Zhang et al. (2019)	Combination of GA and DBN (GA-DBN)	NSL-KDD	Iterations of GA to generate the number of hidden layers and neurons Compared with FC-ANN, TANN, SA-DT-SVMS, BPNN
	Balyan et al. (2022)	HNIDS, enhanced genetic algorithm and swarm optimization (EGA-PSO), improved random forest (IRF)	NSL-KDD	Hybrid EGA-PSO to enhance the minor data samples IRF elimates the less significant attributes

Table 2.1 continued

NN	Mukkamala et al. (2005)	Ensemble of ANNs, SVMs and MARs	DARPA98	NNs, SVMs and MARs in IDS
	Zhang et al. (2005)	SHIDS, PHIDS	KDD 99	Serial and multiple RBF NN & clustering
	Tong et al. (2009)	RBF NN, Elman Network	DARPA98	RBF NN to be real-time pattern classification Elman Network to restore the memory of past events Compared with MLP/leaky bucket, Elman/leaky bucket, SOM/RPROP
	Wang et al. (2010)	FC-ANN	KDD 99	Combination of ANN and fuzzy clustering Solve the detection stability of ANN Fuzzy clustering for generating different training subsets ANN for formulating different base models Fuzzy aggregation for aggregate the results Compared with DT, naïve bayes, BPNN
	Kumar and Yadav (2015)	Neural Network (41, 29, 5)	KDD 99	Compared with the distance Sum-Based Hybrid Method (Guo et al., 2014), SOM-ANN (Ibrahim et al., 2013)
	Singh et al. (2015)	Online sequential extreme learning machine (OS-ELM)	NSL-KDD Kyoto 2006+	Reduced computational time and memory requirements. Multiple topologies of the proposed neural network architecture are proposed.
	Alheeti et al. (2015)	NN	Generated with simulation tools: SUMO and MOVE	IDS method for DOS detection in VANET infrastructure
	Niyaz et al. (2015)	Sparse AE, Soft-max regression	NSL-KDD	Unsupervised feature learning by sparse AE Self-taught learning Soft-max regression as the classifier
	Alom et al. (2016)	DBN	NSL-KDD	Better accuracy than SVM and DBN-SVM Only a subset of NSL-KDD database was used as a training input
	Kim et al. (2016)	LSTM+RNN	KDD 99	Experimentation on NN parameters Results compared with GRNN, PNN, RBNN, KNN, SVM, Bayesian
	Hodo et al. (2016)	NN	Internet packets	IoT security against DOS and DDOS
	Pandeeswari and Kumar (2016)	Fuzzy means clustering (FMC), ANN	KDD 99	Cloud environments Fuzzy means clustering of input data Results compared with Naïve Bayes and ANN
	Erfani et al. (2016)	DBN, SVM	Six real-life and two synthetic datasets (not network related)	Unsupervised dimensionality reduction by DBN Anomaly detection by SVM No network traffic data
	Kang and Kang (2016)	DBN, DNN	Generated with OCTANE simulation software	For automotive communication
	Tang et al. (2016)	Deep NN (three hidden layers)	NSL-KDD	Six feature subsets from NSL-KDD Compared with: J48, Naïve bayes, NB Tree, Random Tree, MLP, SVM Software-defined networking
	Saied et al. (2016)	NN	Generated with the simulation of DDOS attacks with tools	Supervised method for detection of known and unknown DDOS attacks in real-time
	Ashfaq et al. (2017)	NN, Fuzziness based algorithm	NSL-KDD	Semi-supervised method Compared with: J48, Naïve bayes, NB Tree, Random Forest, Random Tree, MLP
	Yin et al. (2017)	Recurrent NN	NSL-KDD	Different topologies of recurrent NN Compared with: J48, Naïve bayes, NB Tree, Random Forest, MLP, SVM

Table 2.1 continued

NN	Du et al. (2017)	LSTM	HDFS logs OpenStack logs VAST challenge 2011 (testing)	System logs as training data Verification of VAST challenge 2011 dataset
	Wang et al. (2017)	CNN	10 types of normal traffic, Gmail 10 types of malware traffic, Cridex	Taking network traffic data as images New dataset USTC-TFC2016
	Shone et al. (2018)	Non-symmetric deep AE random forest	NSL-KDD KDD 99	Based on stacked AE and Random forest Compared with DBN
	Fernandez et al. (2018)	LSTM, DBN, SAE	CTU	5G networks Compared the performance between CPU and GPU Compared different frameworks
	Tang et al. (2018)	GRU-RNN	NSL-KDD	GRU-RNN is the first to use for an IDS in the SDN environment Compared with VanillaRNN, SVM, DNN
	Benmessahel et al. (2018)	Combination of new EA (MVO) and ANN (ENN)	NSL-KDD UNSW- NB15	Use MVO to train feedback forward ANN Compared with ANN, LR, DT, NB, EM
	Zhang et al. (2019)	Combination of GA and DBN (GA-DBN)	NSL-KDD	Iterations of GA to generate the number of hidden layers and neurons Compared with FC-ANN, TANN, SA-DT-SVMS, BPNN
	Ieracitano et al. (2020)	Statistical techniques and AE	NSL-KDD	Compared with LSTM, MLP, L-SVM, Q-SVM, LDA, QDA
	Hassan et al. (2020)	Combination of CNN and WDLSTM (CNN-WDLSTM)	UNSW-NB15 ISCX2012	CNN for extracting feature WDLSTM for preventing overfitting Compared with CNN, Inception CNN, LSTM, GRU, TSD model
	Li et al. (2020)	AE random forest, AP	CSE-CIC-IDS 2018	AP feature grouping Compared with KitNet
	Fatani et al. (2022)	SI, CNN, AQU	CIC2017 NSL-KDD BoT-IoT KDD99	CNN to extract feature AQU (developed SI) to select features Compared with FFA, PSO, WOA, MFO, TSO, MVO, Bat algorithm, GWO IoT
	Zhou et al. (2022)	GNN	UNSW-SOSR2019	A hierarchical adversarial attack (HAA) to realize adversarial attack strategy Shadow GNN with saliency map technique to generate adversarial examples Node selection algorithm based on RWR to select a set of more vulnerable nodes IoT devices Compared with iRWR, RSM and GCRW1
	Shiomoto (2023)	AAE-based	NSL-KDD	AE extracts variable vectors to hold the features of the input data Encoder to hold the features representing the class information and the other features Compared with MLP-based DNN
	Thakkar and Lohiya (2023)	DNN	NSL-KDD UNSW_NB-15 CIC-IDS-2017	Fusion of statistical importance using standard deviation and difference of mean and median to select features
Other	Wang and Battiti (2006)	PCA	DARPA98	Each network connection is transformed into an input data vector PCA to reduce the dimensionality
	Casas et al. (2012)	Sub Space Clustering, DBSCAN, EAR (UNIDS)	Public MAWI KDD 99	Unsupervised classification Combination of Sub-Space clustering and Multiple Evidence Accumulation techniques
	De la Hoz, et al. (2015)	PCA, FDR	NSL-KDD	Unsupervised hybrid method

Table 2.1 continued

Other	Ozay et al. (2016)	Various machine learning techniques	Generated test system for smart grid	Performance and accuracy comparison of new ML methods for smart grid attack detection
	Mazini et al. (2019)	ABC, AdaBoost	NSL-KDD ISCXIDS2012	ABC for feature selection AdaBoost for evaluating and classifying Compared with AdaBoost, SVM+BRICH, LGP-BA+SVM, SVM+C4.5, Layered MPPO, OS-ELM, BC+SVM, AdaBoost+J48+IG, KNN+K-means, SVM+ELM+K-means
	Gupta and Agrawal (2021)	Uncertain C capitals clustering (UCC), Stochastic relaxation algorithm, Ada boost ensemble machine learning	Real-life environment	UCC and stochastic relaxation cluster data Ada boost ensemble machine learning as the classifier Compared with DT, Random forest, RBF
	Abu et al. (2023)	LS-PIO, ensemble learning approach	BoT-IoT UNSW-NB15 NLS-KDD KDDCUP99	IoT Enhance the PIO by adding a local search algorithm (LS)

As seen from the above table, the application of individual machine learning algorithms in intrusion detection has evolved into utilizing the strengths of various algorithms to achieve better results over time (Kocher & Kumar, 2021).

In the initial stage of research, the application of machine learning in intrusion detection was primarily focused on the use of a single algorithm. The objective of these studies was to reduce the dimensionality of the data and to evaluate the performance of different algorithms for intrusion detection. As time progressed, the research focus shifted towards the combination of multiple algorithms to leverage the strengths of each and achieve improved performance.

It also can be seen that, the use of neural networks in intrusion detection has attracted great attention due to their ability to learn complex relationships between features and detect patterns in large amounts of data (Kocher & Kumar, 2021). These algorithms can be trained on different types of data such as network packets, system logs, and even application behaviour data, making them suitable for different types of intrusion detection problems.

Various deep neural network algorithms, including DNN (Deep Neural Network), LSTM (Long Short-Term Memory), CNN (Convolutional Neural Network), and DBN (Deep Belief Network), have been extensively studied and have demonstrated promising results.

Moreover, the use of ensemble-based approaches, which combines the results of multiple algorithms, has also been increasingly popular in recent years. These approaches are designed to overcome the limitations of individual algorithms and improve the overall performance of the intrusion detection system. Some of the commonly used ensemble-based approaches in intrusion detection include fuzzy logic, Adaboost, Bagging, Random Forest, and Boosting.

Additionally, due to the inherent fuzziness and uncertainty in intrusion detection problems, there has been a growing interest in addressing these challenges through innovative techniques, especially in IoT devices.

In conclusion, the field of intrusion detection has seen a lot of advancements in recent years, with various algorithms and techniques being developed and applied. As the amount of data generated continues to grow, it is expected that the application of machine learning and software computing techniques in intrusion detection will continue to evolve and provide more effective solutions for securing networks and systems.

2.3 Fuzzy Logic-based IDS

The concept of intrusion detection can be traced back to the works of Anderson (1980) and Denning (1987). Anderson defined intrusion as the potential unauthorized attempt to access, manipulate, or disrupt information and systems, while Denning proposed

the use of real-time intrusion detection systems to identify and prevent such malicious activities.

In intrusion detection, it is often difficult to distinguish between the behaviour of intruders and that of legitimate users due to their overlap, as shown in Figure 1.1. As Stallings and Brown (2017) noted, intrusion detection technology serves as a preventive measure to detect intrusions before any damage occurs. However, the inherent uncertainty and overlap in the behaviour patterns can result in a certain rate of misjudgement, a unique characteristic of intrusion detection. To address this challenge, it is important to consider the fuzzy nature of intrusion detection during the classification process to enhance the overall performance.

In the field of computer science and artificial intelligence, the study of uncertainty is an important and challenging research topic. Fuzzy logic is a method for dealing with uncertainty and imprecision, and it has been successfully applied in various fields such as control systems, decision-making, and pattern recognition. Norbert Wiener, the founder of cybernetics, once pointed out that the ability to use fuzzy concepts is one of the characteristics that sets humans apart from machines. Fuzziness refers to the ambiguity and vagueness that exists in human cognition and language communication. Unlike randomness, which reflects objective uncertainty, fuzziness mainly arises from the subjective understanding and interpretation of the extension of concepts. Through the use of fuzzy logic and related techniques, it is possible to model and reason about uncertain and imprecise information, and to enable machines to emulate some aspects of human reasoning and decision-making.

2.3.1 Fuzzy System and Uncertainty

Classical logic is based on the premise that truth and falsehood are binary, absolute, and mutually exclusive. However, in the late 19th century, scholars started to acknowledge the presence of uncertainty in physical systems. This realization challenged the notion of absolute truth and led to the development of alternative approaches that could account for uncertainty and fuzziness in reasoning and decision-making processes. Since the early 20th century, many-valued logic has acknowledged additional truth values, typically considered as intermediate states between true and false and referred to as degrees of truth. The classical sets, being too simplistic, could not extend to many-valued predicates. Lotfi Zadeh (1965) reintroduced the connection between fuzzy sets and many-valued logic in his seminal paper in 1965.

The introduction of fuzzy sets was primarily motivated by the need to represent and handle classes of objects that lack precise definitions. Such classes are commonly encountered in human activities that involve natural language, and fuzzy sets provide novel and potentially valuable perspectives on human cognition, reasoning, communication, decision-making, and more. One of the most significant applications of fuzzy sets is in the representation of knowledge expressed by natural language statements. This type of knowledge has gained new importance due to the ability to represent and manipulate it in a mathematically rigorous manner using fuzzy set theory.

Since Mamdani's pioneering work on applying fuzzy logic to steam engine control in 1974, fuzzy control has been extensively applied in various fields such as industrial process control, robotics, transportation, and more, and has demonstrated excellent performance. Fuzzy control leverages the control experience of human experts, offering the

advantages of good robustness and high control performance for the control of nonlinear and complex objects. Additionally, fuzzy logic has been applied in areas such as clustering analysis, fault diagnosis, expert systems, and image recognition, among others.

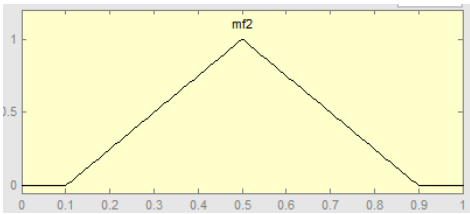
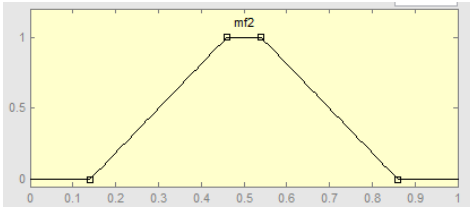
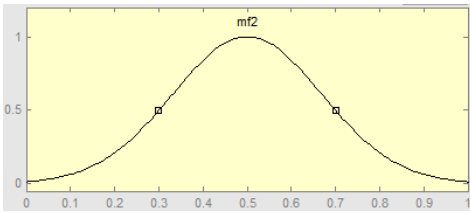
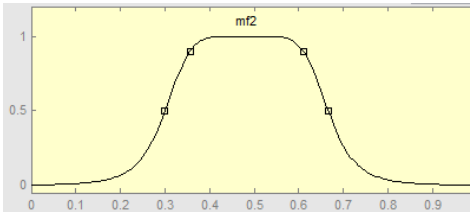
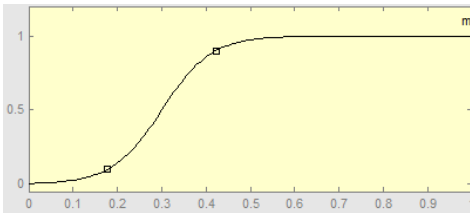
To represent uncertainty in real-world situations, fuzzy logic departs from the traditional assumption that the degree to which elements in the domain conform to the concept is either 0 or 1, and instead assigns a real number between 0 and 1 to the domain, as shown in Equation 2.1 (Siddique & Adeli, 2013).

$$\mu_A: U \rightarrow [0,1] \quad \text{Equation 2.1}$$

The range of values for a fuzzy set is denoted by U. In the context of fuzzy sets, the membership function (MF) is a function that defines the degree of membership or fuzziness of an element in U to a particular fuzzy set A. It associates each point in U with a real number in the interval [0, 1]. The membership function plays a crucial role in characterizing the degree of membership of elements in a fuzzy set.

When the real number domain R is taken as the domain, the membership function is called fuzzy distribution. There are five common types of fuzzy distribution, as shown in Table 2.2.

Table 2.2: Four Types of MFs

MF	Equation	Distribution Curve
Triangular	$\mu(x) = \max(\min(\frac{x-a}{b-a}, \frac{c-x}{c-b}), 0)$	
Trapezoidal	$\mu(x) = \max(\min(\frac{x-a}{b-a}, 1, \frac{d-x}{d-c}), 0)$	
Gaussian	$\mu(x) = e^{-\frac{1}{2}(\frac{x-m}{\sigma})^2}$	
Generalised Bell	$\mu(x) = \frac{1}{1 + \frac{x-m}{\sigma} ^{2a}}$	
Sigmoidal	$\mu(x) = \frac{1}{1 + e^{-a(x-c)}}$	

Let's consider a fuzzy set A in the universal set U. The membership function of fuzzy set A is denoted as $\mu_A(x)$, where x represents an element in U. The membership function assigns a degree of membership, denoted by $\mu_A(x)$, to each element x in U, indicating the extent to which x belongs to fuzzy set A. In other words, $\mu_A(x)$ represents the grade of membership of x in A.

Equation 2.2 illustrates this concept, where $\mu_A(x)$ is the membership function of fuzzy set A, and x belongs to the universal set U. The values assigned by the membership function, denoted by $\mu_A(x)$, provide a continuum of grades of membership, indicating the varying degrees to which elements belong to the fuzzy set A.

$$A = \{x, \mu_A(x) \mid x \in U\} \quad \text{Equation 2.2}$$

There is no absolute conclusion that the value is True or False in which category, only the probability in each category.

Fuzzy logic utilizes fuzzy sets as the basis and employs If-Then rule statements to express the conditional statements in fuzzy logic systems. These If-Then rule statements are used to represent the relationships and dependencies between the input variables and the output variables, as shown in Figure 2.1. The rule is shown as Equation 2.3.

$$\text{If } \langle \text{fuzzy proposition} \rangle \text{ Then } \langle \text{fuzzy proposition} \rangle \quad \text{Equation 2.3}$$

For instance, let's consider the If-Then rule: "If Temperature is High, Then Pressure should be High." In this rule, "Temperature" and "Pressure" are the variables, and "High" is a linguistic term representing a fuzzy set. The membership function $\mu_A(\text{Temperature})$ describes the degree to which a given value of Temperature belongs to the fuzzy set "high"

Similarly, the membership function $\mu_A(\text{Pressure})$ describes the degree to which a given value of Pressure belongs to the fuzzy set "high". In this case, there may be an overlap between the fuzzy sets "high" and "low" since the linguistic terms are not mutually exclusive. The membership function allows us to define this overlap by assigning membership degrees to each fuzzy set based on the given input value.

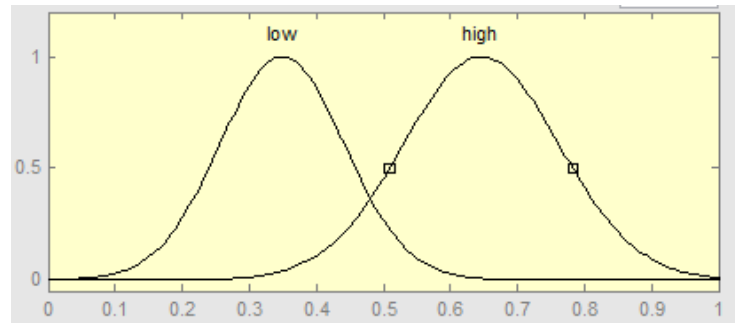


Figure 2.1: If-Then Rule

The inference mechanism of fuzzy logic is a nonlinear mapping from a given input space to an output space. There are three types of fuzzy inference mechanisms widely used in fuzzy systems and their applications, which are different from their fuzzy rules, aggregations and defuzzification procedures. They are Mamdani fuzzy inference, Sugeno fuzzy inference and Tsukamoto fuzzy inference, as shown in Table 2.3.

Table 2.3: Three Types of Fuzzy Inference

Fuzzy Inference	Model
Mamdani	If x is A_i^k and y is B_j^k then z is C_l^k
Sugeno	If x is A_i^k and y is B_j^k then $z^k = f(x, y)$
Tsukamoto	If x is A_i^k and y is B_j^k then z is C_l^k

The general main components of fuzzy logic have four components:

- i. Data Fuzzification: If the input data is concrete, the data should be fuzzed through membership functions.
- ii. Selection of Inference Mechanism: Inference mechanism provides a nonlinear mapping which is the basis of decisions. The fuzzy inference mechanism includes membership functions, fuzzy logic operators and if-then rules.
- iii. Set of Rules: A set of rules based on expert knowledge, which is the rule base of a fuzzy logic controller.
- iv. Defuzzification: Turn the fuzzy control actions into non-fuzzy control actions.

When designing fuzzy systems, two key considerations are the construction of membership functions (MFs) and the rule base.

- i. Construction of Membership Functions (MFs)

Membership functions define the degree of membership of an input value to a particular linguistic term or fuzzy set. The shape and parameters of the membership functions greatly influence the behaviour of the fuzzy system. It is important to carefully design the MFs to accurately capture the underlying characteristics of the input variables. The choice of MF shapes (such as triangular, trapezoidal, Gaussian, etc.) and the placement of their parameters (such as the centre, width, and slope) should reflect the domain knowledge and the specific problem at hand. Fine-tuning the MFs is crucial to ensure the system's effectiveness and interpretability.

ii. Construction of the Rule Base:

The rule base consists of a set of If-Then rules that express the relationships between the input variables and the output variable of the fuzzy system. Each rule typically consists of an antecedent (If-part) and a consequent (Then-part). The antecedent contains linguistic terms defined by the MFs and is used to evaluate the input variables. The consequent represents the output linguistic term or fuzzy set. The rule base should be constructed based on expert knowledge or through data-driven approaches. It is important to carefully select and formulate the rules to capture the desired behaviour and logic of the system. The rule base can be constructed manually or through automated methods such as rule extraction or rule learning algorithms.

Overall, the construction of accurate and appropriate membership functions and the design of a well-structured rule base are crucial steps in designing effective fuzzy systems that can model complex relationships and make intelligent decisions based on uncertain or imprecise inputs.

2.3.2 Fuzzy Logic-based IDS

IDSs based on fuzzy logic use fuzzy sets to model the uncertainty and imprecision of intrusion detection. Fuzzy logic provides a powerful mechanism for dealing with uncertain and vague information, which makes it well-suited for detecting intrusions in complex and dynamic network environments.

In a fuzzy logic-based IDS, the input features, such as network traffic patterns and system logs, are converted into fuzzy sets using membership functions. The membership

functions map the input features to a degree of membership in a fuzzy set, which represents the likelihood of an intrusion occurring.

Fuzzy rules are then constructed to relate the input features to the output, which is the degree of intrusion. The rules are often formulated using the IF-THEN structure, where the antecedent (IF part) specifies the input features, and the consequent (THEN part) specifies the degree of intrusion.

The fuzzy inference engine then combines the fuzzy rules and the input features to generate a degree of intrusion. The degree of intrusion is often represented as a linguistic variable, such as "low," "medium," or "high," which provides a more interpretable output than a numerical value.

Fuzzy logic-based IDS have shown promising results in detecting a wide range of attacks, including DDoS attacks, port scanning, and SQL injection attacks. They have the advantages of being flexible, robust, and adaptive to changes in the network environment. However, they require careful design of membership functions and fuzzy rules, as well as accurate training data to ensure good performance.

According to the fuzzy algorithms and techniques employed, the application of fuzzy logic in intrusion detection can be categorized into nine categories (Masdari & Khezri, 2020), as presented in Figure 2.2.

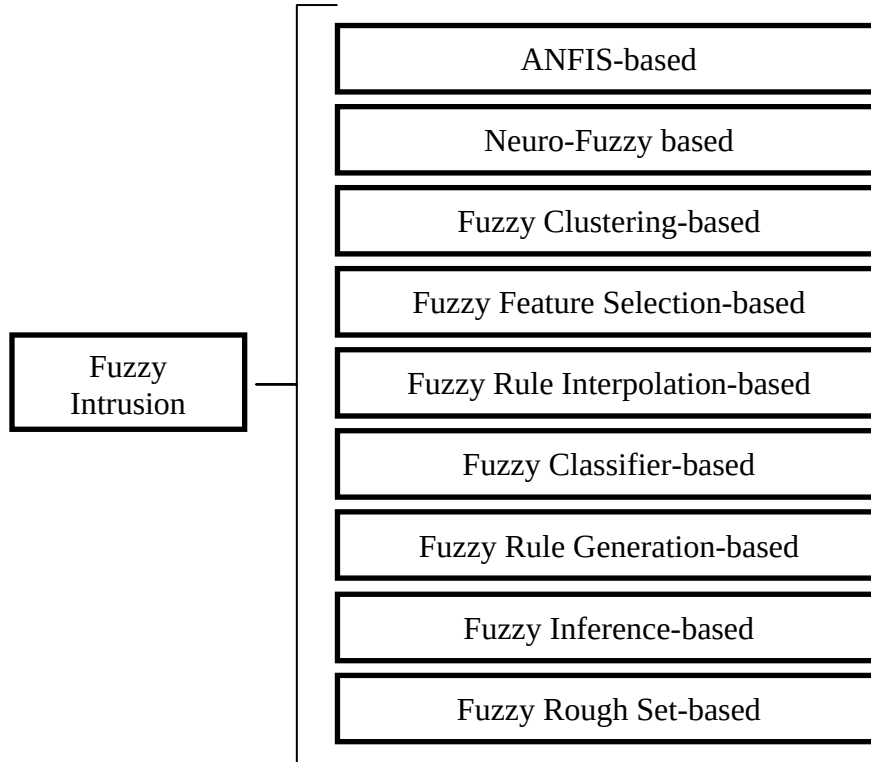


Figure 2.2: Classification of Fuzzy IDSs

In earlier applications of fuzzy logic in IDS, certain algorithms were commonly used to reduce data dimensionality, and simple association rule mining was employed to identify various intrusion classes. For instance, Wang et al. (2007) employed rough set theory to horizontally reduce the KDDCUP99 dataset, and proposed an improved association rule discovery system based on rough set theory. This method utilized a semi-naïve algorithm to cluster different decision classes, and redefined association rule mining within the framework of rough set theory. Subsequently, Tajbakhsh et al. (2009) utilized association hyper-edges to reduce items and employed ABC (Association Based Classification) as the descriptive model for each class.

2.3.3 Hybrid Model with Fuzzy Logic

As the research on intrusion detection systems continues, combining fuzzy logic with other algorithms has become a research hotspot. One example of this is the work of Wang et al. (2010), who used fuzzy clustering to enhance the stability of ANN-based IDS. The proposed approach, called FC-ANN, aims to enhance the performance of intrusion detection systems by combining ANNs and fuzzy clustering. The procedure involves using fuzzy clustering to generate subsets for training different ANNs, and then aggregating the results using a fuzzy aggregation module. Results of experiments on the KDDCUP99 dataset show that FC-ANN outperforms other well-known methods in terms of detection precision and stability.

Boughaci et al. (2012) propose a fuzzy particle optimization algorithm (FPSO) for intrusion detection, which optimizes a fuzzy rule-based knowledge base using a PSO algorithm. The method is evaluated using the KDDCUP99 intrusion dataset and compared to other existing techniques. The results demonstrate the effectiveness of the proposed approach in generating high-quality solutions.

Chandrasekhar and Raghuveer (2013) propose the utilization of data mining techniques, specifically neuro-fuzzy and radial basis SVM (Support Vector Machine), to enhance the performance of Intrusion Detection Systems (IDSs). The approach involves four major steps, including the use of k-means clustering to generate training subsets, training different neuro-fuzzy models, forming a vector for SVM classification, and performing classification using radial SVM to detect intrusion. The approach is evaluated using the KDDCUP99 dataset, and the results show that it outperforms other well-known methods in terms of sensitivity, specificity, and detection accuracy. The proposed approach has the

potential to improve the security of financial and business web applications in the era of the internet.

Elhag et al. (2015) proposed a novel and robust linguistic FRBCS based on association discovery. This method first combines GFS and OVO to develop a robust and interpretable IDS and uses FARCHD (a linguistic fuzzy association rule mining classifier) to learn the model for each class.

Pandeewari and Kumar (2016) used FMC (Fuzzy means clustering) to create different classes, trained different ANNs in each class and combined the results of different ANNs using fuzzy aggregation.

Ashfaq et al. (2017) extended the divide-and-conquer methodology for fuzzy classification and employed NNRw (Neural network with random weights) as the classifier in their study.

Wang et al. (2022) use TF-IDF and Z-Score to normalize and standardize the data to construct a fuzzy triadic background containing quadratic characteristics. In their study, the (i)-induced operator is used to construct the fuzzy triadic concept set based on the fuzzy triadic background and transformed into a fuzzy attribute triadic concept set.

Gorzalczany and Rudzinski (2022) present a new approach for intrusion detection in IoT systems that is accurate and interpretable. The approach uses a fuzzy rule-based classifier and a multi-objective evolutionary optimization algorithm to optimize the accuracy-interpretability trade-off of the IoT intrusion detection systems. The main contribution of the paper is the design of accurate and interpretable IoT IDSs using the MQTT-IOT-IDS2020 dataset. The proposed approach is compared to seven alternative methods and is shown to

outperform them in terms of interpretability of intrusion-detection decisions while remaining competitive or superior in terms of accuracy.

Alrayes et al. (2023) present a MFSFL-IDS model for IoT devices. This model uses Henry gas solubility optimization (HGSO) algorithm to select features. ANFIS technique is utilized for intrusion classification in the network. Meanwhile, the binary bat algorithm (BBA) is exploited for adjusting parameters involved in the ANFIS. This model achieves a maximum accuracy of 99.80%, compared with recent approaches.

Fuzzy logic has several advantages when combined with other algorithms for intrusion detection. Firstly, it allows for a more flexible system, where uncertainty and imprecision in the data can be more effectively handled. This can lead to better accuracy in identifying intrusions while minimizing false positives. Additionally, when combined with other algorithms such as decision trees, artificial neural networks or support vector machines, the system can benefit from the strengths of both approaches, leading to higher detection rates and more robust performance. Furthermore, most fuzzy logic algorithms are easily interpretable, which can enhance the interpretability of intrusion detection systems. Overall, the combination of fuzzy logic with other algorithms can provide a powerful and effective tool for intrusion detection in various contexts.

2.4 Neural Network-based IDS

Neural Networks (NNs) have been widely used in IDSs due to their ability to learn and adapt to new situations. NNs are trained on a dataset of normal and abnormal traffic patterns and can then classify new traffic as normal or potentially malicious based on the learned patterns. NNs have shown promise in improving detection accuracy and reducing false positives.

Neural Networks are a subfield of machine learning, originating from research dating back to the 1940s. The Neural Network paradigm emulates the cognitive process of the human brain, as illustrated in in Figure 2.3.

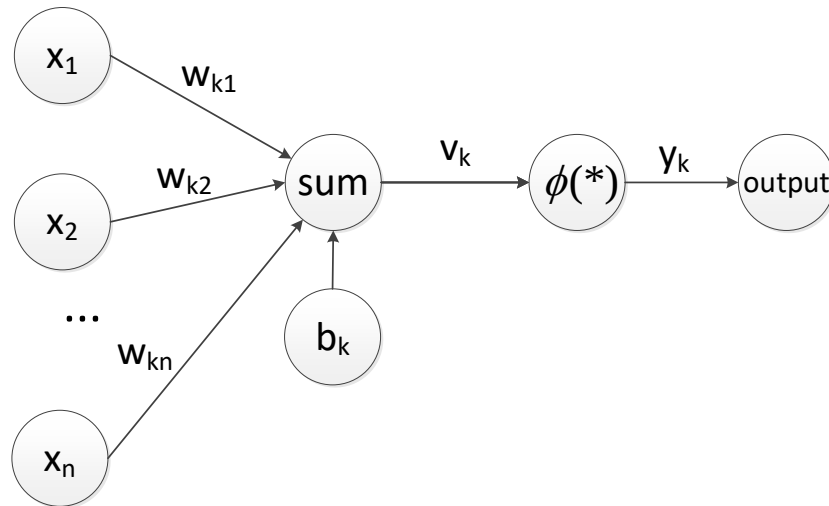


Figure 2.3: Single Artificial Neural Network

The development of neural networks has gone through three waves (LeCun et al., 2015). The first wave, from the 1940s to the 1960s, was the cybernetics wave, which focused on training individual neurons. The second wave, known as connectionism, occurred from 1980 to 1995 and utilized backpropagation to train neural networks. The third and current wave began in 2006 and is characterized by the emergence of deep learning, which involves training more complex neural networks with a larger number of hidden layers.

The third wave of deep learning has shown outstanding performance in the field of image recognition. In 2012, a team led by Hinton (2017) made a breakthrough in image recognition by winning the prestigious ImageNet competition with their deep learning model, AlexNet. In the same year, DNN (deep neural network) developed by Professor Andrew Ng and Jeff Dean achieved remarkable performance in image recognition, reducing the error

rate in ImageNet evaluation from 26% to 15%. The success of deep learning algorithms in the competition drew widespread attention from both academia and industry in the field of deep learning.

Deep learning has achieved remarkable success in the field of image recognition, leading to the widespread adoption of deep learning in various fields, including intrusion detection. In the field of intrusion detection, neural networks, and in particular deep neural networks, have shown great promise in detecting various types of attacks with high accuracy. Deep learning models have the ability to automatically learn and extract features from the input data, which is particularly useful for detecting attacks with complex patterns and behaviours.

With the increasing availability of datasets, such as NSL-KDD and UNSW-NB15, researchers have been able to train deep learning models for intrusion detection with high accuracy. In addition, there have been efforts to optimize deep learning models for intrusion detection, such as using transfer learning, ensemble learning, and adversarial training.

2.4.1 Deep Neural Network

Artificial neurons were introduced in 1943 when McCulloch and Pitts (1943) proposed a mathematical model to simulate the behaviour of biological neurons. Their model, known as the McCulloch-Pitts neuron, aimed to capture the fundamental characteristics of neurons and their ability to process and transmit signals. In 2006, Hinton (2006) showed that a neural network called a deep belief network could be efficiently trained using a strategy called greedy layer-wise pretraining. This change greatly improves the ability of neural networks. Meanwhile, it popularized the use of the term “deep learning” to emphasize that researchers were now able to train deeper neural networks than had been

possible before and to focus attention on the theoretical importance of depth. Deep neural networks outperformed competing for other AI systems in various fields. Today, deep learning is getting more attention in many other fields (intrusion detection etc.).

Deep learning algorithms have shown promise in enhancing intrusion detection systems (IDSs) by capturing complex underlying patterns in data. Various deep learning algorithms, such as artificial neural networks (ANN) (Hodo et al., 2016b), Long Short-Term Memory (LSTM) (Althubiti et al., 2019), and Convolutional Neural Networks (CNN) (Xiao et al., 2019), have been proposed and applied in IDSs. The success of deep learning in image recognition has sparked interest in its application to other domains, including intrusion detection.

Deep learning is a machine learning model that utilizes multiple layers of nonlinear transformations of features, thus reflecting its depth. The most commonly used deep learning model is the multi-layer neural network, where each layer of the neural network performs nonlinear mapping for the input. By stacking multiple layers of nonlinear mapping, deep neural networks can compute abstract features to aid in classification. The learning process is achieved by mapping the input to the new feature space layer by layer, from low to high, and as a result, it exhibits a hierarchy and distributed abstraction characteristic. This enables the fitting of complex nonlinear functions and processing of high-dimensional nonlinear input data. As deep learning algorithms continue to develop, they have become increasingly deeper and more intelligent, as shown in Figure 2.4.

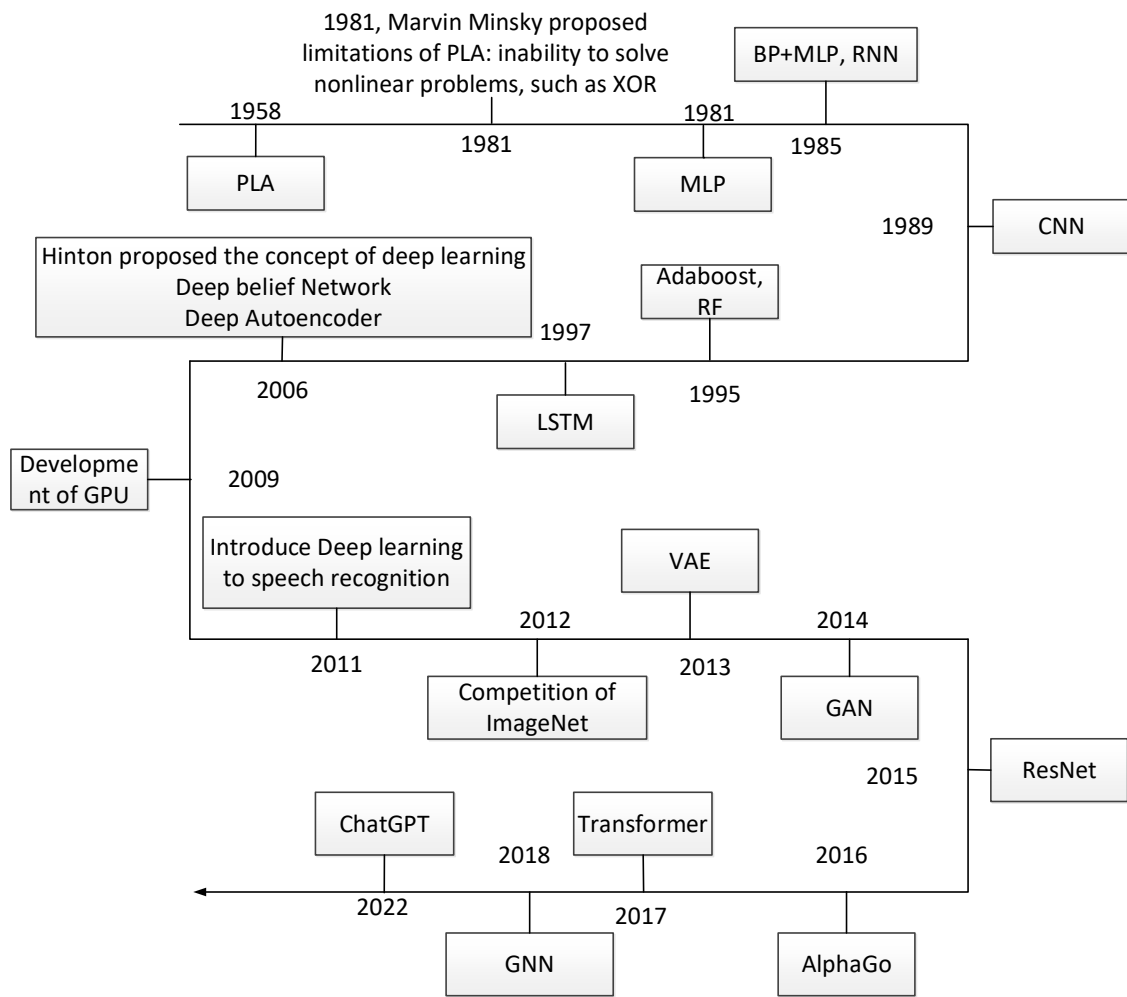


Figure 2.4: Development of Deep Learning

The learning characteristics of deep learning structure are:

- i. Deep learning networks can be pre-trained using unsupervised learning methods such as autoencoders or restricted Boltzmann machines. This pre-training step helps initialize the weights in the network and can lead to better performance on supervised learning tasks.
- ii. Pre-training is often performed layer by layer, where each layer is trained using unsupervised learning, and the output of the lower layer is used as the

input for the higher layer. This stepwise pre-training can help the network learn features at different levels of abstraction.

- iii. After pre-training, the network can be fine-tuned using a supervised learning algorithm, typically using a backpropagation-based optimization method. The entire network is trained end-to-end, adjusting all the weights to minimize a specific loss function.

The learning capabilities of neural networks enable them to learn intricate representations of data by utilizing multiple layers of abstraction. This ability has led to impressive performance in various machine learning tasks, including image recognition, speech recognition, and natural language processing.

Currently, there exist numerous network structures in deep learning, some of them are shown in Figure 2.5.

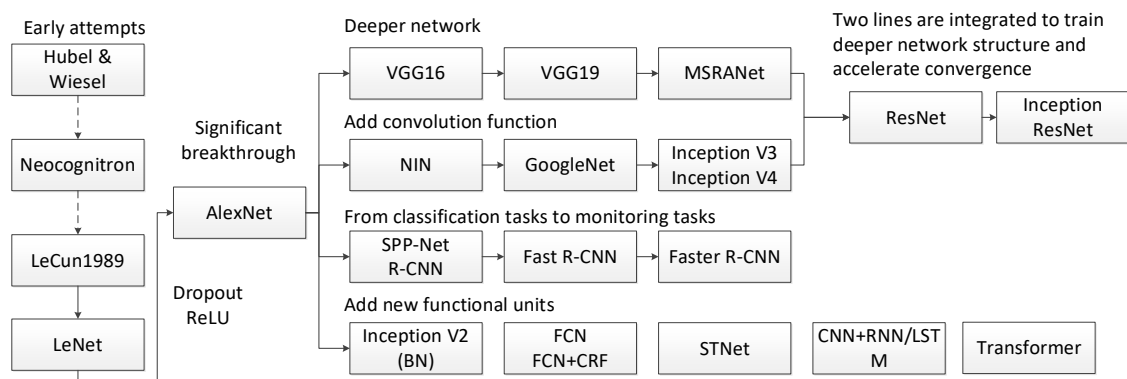


Figure 2.5: Deep learning models

AlexNet (Krizhevsky et al., 2017) is a pioneering deep learning network architecture that has had a profound impact on the field, paving the way for breakthrough research. Its simplicity and effectiveness have made it an important milestone in the development of deep learning. The architecture of AlexNet consists of a convolutional layer and a pooling layer, followed by a fully connected layer. Prior to AlexNet, CPUs were typically used to train neural networks, but AlexNet made training much faster by using GPUs, reducing training time by a factor of 10. Although AlexNet was proposed several years ago and deep learning has made significant progress since then, it remains a critical starting point for applying deep neural networks to a wide range of tasks, including computer vision and speech recognition.

CNN (LeCun et al., 1998) is a deep learning architecture that leverages convolution operations to extract local image features, and pooling layers to reduce the dimensionality of the data. This reduces the dimensionality of large image volumes while effectively retaining important image features. CNNs have found extensive applications in various areas, including face recognition and autonomous driving. LeNet was one of the earliest CNN structures proposed.

RNNs (Zaremba et al., 2014) are a class of artificial neural networks in which the interconnections between units form a directed cycle, endowing it with the ability to exhibit dynamic temporal behaviour. Unlike feedforward neural networks, RNNs can utilize their internal memory to process arbitrary input sequences, making them suitable for tasks such as unsegmented, connected handwriting recognition or speech recognition.

LSTM (Hochreiter & Schmidhuber, 1997) is a type of RNN that addresses the vanishing gradient problem and allows for bidirectional data flow in the network. LSTM is particularly effective for processing and predicting events with long intervals and temporal

delays in time series data, making it well-suited for sequence learning tasks. Its ability to model long-term dependencies has made it a popular choice for various applications, including speech recognition and natural language processing. Additionally, LSTM has shown superior performance compared to traditional sequence learning methods, such as Hidden Markov models.

DBN (Deep Belief Network) was proposed by Hinton et al. (2006). It is a generative model that can not only identify and classify data features but also generate data by training the weights among its neurons to maximize the probability of generating the training data. RBMs (Restricted Boltzmann Machines) are the key components of DBN. Each layer of the RBM network is trained separately and unsupervised, retaining as much feature information as possible when feature vectors are mapped to different feature spaces. In the last layer of DBN, a supervised learning algorithm such as backpropagation is used to train the entity relationship classifier with the output feature vector of RBM as input. Additionally, error information is propagated from the top to bottom layer of RBMs, fine-tuning the whole DBN network. The training process of the RBM network can be regarded as initializing the weight parameters of a deep BP (backpropagation) network, overcoming the shortcomings of the BP network, such as the tendency to fall into local optima and long training times due to random initialization of weight parameters.

The VGG (Visual Geometry Group) network (Simonyan & Zisserman, 2014), introduced by researchers at the University of Oxford's Visual Geometry Group (hence the name VGG), is characterized by its pyramidal shape, with a wide bottom layer closest to the image and a deep top layer. The network contains subsequent convolution and pooling layers, with the pooling layer responsible for narrowing the layer. The researchers proposed

multiple networks of this type, varying in architectural depth. VGG is a suitable architecture for task-specific benchmarks, although its main disadvantage is that training the model from scratch can be very slow.

The GoogleNet, also known as the Google InceptionNet (Szegedy et al., 2016), is an architecture developed by Google researchers. GoogleNet achieved a strong performance and won the ImageNet 2014 challenge. Compared to VGG's 19 layers, GoogleNet is deeper with 22 layers. The researchers proposed a new approach called initial modules which is a significant departure from the previous sequential architecture. In each layer, there are many types of feature extractors. This approach provides more options for the network to solve the task, whether by convolving the input or directly pooling it. The final architecture consists of several initial modules stacked on top of each other.

The training of GoogleNet is slightly different from previous architectures as most of the top layers have their own output layer. This facilitates faster convergence since the layers can be jointly and in parallel trained. A pre-trained GoogleNet model is smaller than VGG. Although GoogleNet has no inherent drawbacks, it introduced further changes to the architecture to improve performance. One of these changes is called the Xception Network, in which the limit for adding Inception modules has been increased, making it theoretically infinite.

ResNet (He et al., 2016) is a seminal deep learning architecture that has significantly advanced the field. The architecture is composed of multiple residual modules, which serve as the fundamental building blocks of ResNet. A residual module has two options: it can either apply a set of functions to the input or skip this step entirely. These residual modules are then stacked on top of each other to form an end-to-end network. ResNet's main

advantage is its ability to use hundreds or even thousands of residual layers for training the network, which is not possible with traditional sequential networks, where increasing the number of layers often leads to performance degradation.

ResNet18 is a type of residual neural network (ResNet) architecture, specifically an 18-layer deep residual network. The key idea behind ResNet is the use of residual blocks, which introduce "shortcut connections" that allow information to skip layers, bypassing the vanishing gradient problem. This allows the network to learn residual functions, which map input data to output data through layers of non-linear transformations. ResNet18 consists of 18 layers arranged into blocks, with each block containing multiple convolutional layers. The architecture uses "bottleneck" blocks that contain three convolutional layers instead of the standard two layers, helping to reduce the number of parameters while maintaining high representational power (He et al., 2016).

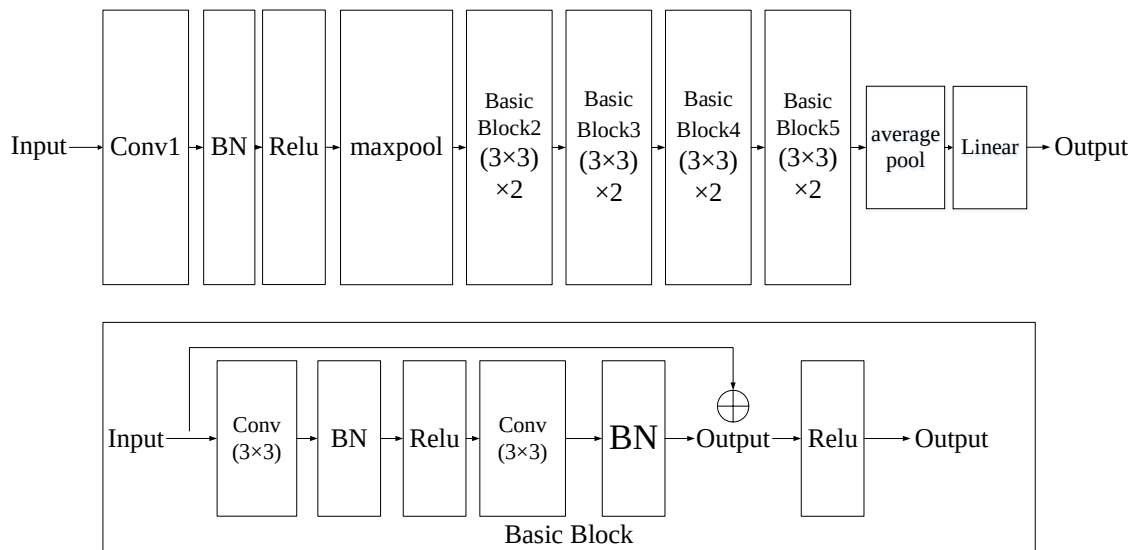


Figure 2.6: The Structure of ResNet18

The Generative Adversarial Network (GAN) architecture (Goodfellow et al., 2020), is a unique neural network model that is capable of generating new images that do not exist in the training dataset, but are realistic enough to belong to that dataset. This model employs two different networks, namely the generator and the discriminator, which compete with each other. The generator is responsible for producing the generated images, whereas the discriminator's task is to differentiate between the generated and required images. The generator continues to modify the generated images until they are able to deceive the discriminator, thereby generating realistic and novel images.

Autoencoder (Hinton et al., 2006) is a type of artificial neural network used for unsupervised learning of efficient coding. It consists of two parts: an encoder and a decoder. The encoder learns to represent the input data in a lower dimensional space, while the decoder learns to reconstruct the original input data from the encoded representation. This process of dimension reduction and reconstruction helps in learning useful features from the input data. Autoencoder models can be deepened by adding multiple layers and trained on large datasets to learn complex features. Once the model is trained, it can be fine-tuned for supervised learning tasks such as classification or image conversion. U-Net (Weng & Zhu, 2021) is a popular autoencoder architecture that uses the encoder-decoder structure for image segmentation tasks. It has skip connections between the encoder and decoder to improve the quality of the output segmentation.

Indeed, there are many other deep learning models that have been developed and applied to various problems. ResNeXt (Xie et al., 2017), is a deep neural network that combines the concepts of Inception and ResNet, and is claimed to be one of the most advanced object recognition technologies available. R-CNN (Ren et al., 2017) is a widely

used deep learning architecture for target detection problems, and is considered to be one of the most influential deep learning architectures in this domain. YOLO (Redmon et al., 2016) is another popular deep learning model that is specifically designed to solve the problem of real-time image detection. SqueezeNet (Iandola et al., 2016) is a compact deep neural network that is more suitable for low-bandwidth scenarios such as mobile platforms. Finally, SegNet (Badrinarayanan et al., 2017) is a deep learning model that is known for its excellent performance in image segmentation tasks. These models have shown remarkable performance on various deep learning problems and continue to be extensively researched and developed in the deep learning community.

Deep learning has several advantages, such as its ability to automatically learn complex patterns and representations from data, its capacity to handle large and high-dimensional datasets, and its ability to generalize to new, unseen data. However, deep learning also presents some challenges. One major issue is the lack of interpretability of deep learning models, which are often considered to be black box models. This opacity makes it difficult to understand how the model makes its decisions, limiting its transparency and interpretability, which is a critical consideration in domains where decisions must be justified, such as in legal or medical contexts. Another drawback of deep learning models is their tendency to overfit if not appropriately regularized, resulting in poor generalization performance on unseen data.

2.4.2 IDS Based on Neural Networks and Deep Learning

In the field of network security, the IDS is a critical tool for identifying and preventing network attacks. The goal of an IDS is to identify events that deviate from a baseline model of normal behaviour and alert the system administrator. Traditional IDS

approaches, such as SVMs, ANNs, and Bayesian Networks, have limitations in terms of accuracy. These techniques often suffer from a high False Alarm Rate (FAR), leading to a large number of false positives.

The advancements in deep learning have revolutionized intrusion detection by introducing new approaches that outperform traditional machine learning techniques in terms of accuracy and performance. One major advantage of deep learning is its capability to process raw data and automatically learn high-level features, eliminating the need for extensive manual feature engineering. Additionally, deep learning models excel at learning from large datasets, a crucial factor in network security where the number of potential attack scenarios is vast and constantly evolving.

Table 2.4 provides a concise overview of recent research focusing on the utilization of neural networks and deep learning for intrusion detection.

Table 2.4: Recent Literature of NN in IDS

Type of NN	Authors	NN	Combined Algorithms	Datasets	Key Findings
DBN	Alom et al. (2016)	DBN		NSL-KDD	Better accuracy than SVM and DBN-SVM Only subset of NSL-KDD database used as a training input
	Erfani et al. (2016)	DBN	SVM	Six real-life and two synthetic datasets (not network related)	Unsupervised dimensionality reduction by DBN Anomaly detection by SVM No network traffic data
DBN	Kang and Kang (2016)	DBN		Generated with OCTANE simulation software	For automotive communication
	Zhang et al. (2019)	DBN	GA	NSL-KDD	Iterations of GA to generate the number of hidden layers and neurons Compared with FC-ANN, TANN, SA-DT-SVMS, BPNN

Table 2.4 continued

DBN	Fernandez et al. (2018)	LSTM, DBN	SAE	CTU	5G networks LSTM for implementing NAD (network anomaly detection) DBN and SAE for ASD (anomaly symptom detection) time measurement process Compared the performance between CPU and GPU Compared different frameworks
ANN	Tong et al. (2009)	Combina- tion of RBF NN and Elman NN		DARPA98	RBF NN to be real-time pattern classification Elman Network to restore the memory of past events Compared with MLP/leaky bucket, Elman/leaky bucket, SOM/RPROP
	Wang et al. (2010)	NN	Fuzzy clustering	KDD 99	Combination of ANN and fuzzy clustering Solve the detection stability of ANN Fuzzy clustering for generating different training subsets ANN for formulating different base models Fuzzy aggregation for aggregate the results Compared with DT, naïve bayes, BPNN
	Alheeti et al. (2015)	NN		Generated with simulation tools: SUMO and MOVE	IDS method for DOS detection in VANET infrastructure
	Kumar and Yadav (2015)	NN (41, 29, 5)		KDD 99	Compared with distance Sum-Based Hybrid Method (Guo et al., 2014), SOM-ANN (Ibrahim et al., 2013)
	Hodo et al. (2016)	NN		Internet packets	IOT security against DOS and DDOS
	Pandeewari and Kumar (2016)	NN	Fuzzy means clustering (FMC)	KDD 99	Cloud environments Fuzzy means clustering of input data Results compared with: Naïve bayes and ANN
	Saied et al. (2016)	NN		Simulation of DDOS attacks with tools	Supervised method for detection of known and unknown DDOS attacks in real time
	Tang et al. (2016)	NN (three hidden layers)		NSL-KDD	Six feature subsets from NSL-KDD Compared with: J48, Naïve bayes, NB Tree, Random Tree, MLP, SVM Software defined networking
	Ashfaq et al. (2017)	NNR _w (NN with random weights)	Divide-and conquer strategy	NSL-KDD	Semi-supervised method Extend divide-and-conquer method NNR _w as classifier Compared with: J48, Naïve bayes, NB Tree, Random Forest, Random Tree, MLP
	Benmessahel et al. (2018)	NN	MVO	NSL-KDD UNSW- NB15	Use MVO to train a feed back forward ANN Compared with ANN, LR, DT, NB, EM
	Thakkar and Lohiya (2023)	DNN	Statistical method	NSL-KDD UNSW_NB-15 CIC-IDS-2017	Fusion of statistical importance using standard deviation and difference of mean and median to select features
RNN	Kim et al. (2016)	LSTM+ RNN		KDD 99	Experimentation on NN parameters Results compared with: GRNN, PNN, RBNN, KNN, SVM, Bayesian
	Yin et al. (2017)	RNN		NSL-KDD	Different topologies of recurrent NN Compared with: J48, Naïve bayes, NB Tree, Random Forest, MLP, SVM

Table 2.4 continued

RNN	Du et al. (2017)	LSTM		HDFS logs OpenStack logs VAST challenge 2011 (testing)	System logs as training data Verification on VAST challenge 2011 dataset
	Fernande et al. (2018)	LSTM, DBN	SAE	CTU	5G networks LSTM for implementing NAD (network anomaly detection) DBN and SAE for ASD (anomaly symptom detection) time measurement process Compared the performance between CPU and GPU Compared different frameworks
	Tang et al. (2018)	RNN (GRU)		NSL-KDD	GRU-RNN is first use for an IDS in the SDN environment Compared with VanilaRNN, SVM, DNN
	Hassan et al. (2020)	Combina tion of CNN and WDLST M (CNN- WDLST M)		UNSW-NB15 ISCX2012	CNN for extracting feature WDLSTM for preventing overfitting Compared with CNN, Inception CNN, LSTM, GRU, TSD model
CNN	Wang et al. (2017)	CNN		10 types of normal traffic, Gmail 10 types of malware traffic, Cridex	Taking network traffic data as images New dataset USTC-TFC2016
	Hassan et al. (2020)	Combina tion of CNN and WDLST M (CNN- WDLST M)		UNSW-NB15 ISCX2012	CNN for extracting feature WDLSTM for preventing overfitting Compared with CNN, Inception CNN, LSTM, GRU, TSD model
	Fatani et al. (2022)	CNN,	SI, AQU	CIC2017 NSL-KDD BoT-IoT KDD99	CNN to extract feature AQU (developed SI) to select features Compared with FFA, PSO, WOA, MFO, TSO, MVO, Bat algorithm, GWO IoT

The table reveals an increasing interest in the use of deep learning for intrusion detection in recent years. Researchers have proposed various neural network architectures, including Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), and Deep Belief Networks (DBN), to detect different types of network attacks. These models have been evaluated on datasets like KDDCUP99 (Tavallae et al., 2009), UNSW-NB15 (Moustafa & Slay, 2015), and NSL-KDD (Tavallae et al., 2009), and have shown promising results in terms of accuracy, detection rate, and false alarm rate.

Deep learning has become a prominent approach in intrusion detection due to its exceptional performance. Researchers have shown great interest in applying deep learning techniques in this field, as they have proven to be effective in analyzing and classifying network traffic, enabling the detection of complex and evolving intrusion patterns. The outstanding performance of deep learning models in intrusion detection has led to their widespread adoption and further exploration (Lansky et al., 20621).

For instance, experiments on the CSE-CIC-IDS2018 dataset by Ferrag et al. (2020) demonstrated the strengths of various deep learning algorithms. DNN (Deep Neural Network) achieved the highest true negative rate (96.915%), RNN (Recurrent Neural Network) showed the highest detection rate for seven attack types, and CNN (Convolutional Neural Network) excelled in four attack types, with the highest overall accuracy (97.376%) using 100 hidden nodes and a learning rate of 0.5. Comparing generative/unsupervised models, CNN exhibited a superior mean false alarm rate compared to DNN and RNN. In deep discriminative models, DA (Deep Autoencoder) performed better than RBM (Restricted Boltzmann Machine), DBF (Deep Belief Network), and DBM (Deep Boltzmann Machine) in terms of mean false alarm rate.

Furthermore, when comparing CNN with Naïve Bayes, ANN (Artificial Neural Network), SVM (Support Vector Machine), DNN, RNN, RF (Random Forest), and others, CNN achieved the highest overall detection rate (97.28%). In the generative/supervised models, the DA model stood out with the highest overall detection rate (98.18%) compared to RF, Naïve Bayes, SVM, ANN, RBM, DBN (Deep Belief Network), and SVM.

These findings highlight the power and potential of deep learning algorithms in intrusion detection, showing their ability to outperform traditional models in various aspects

of performance. As a result, deep learning continues to be an area of active research and development in the field of intrusion detection.

Although deep learning has shown promising results in improving the accuracy and efficiency of intrusion detection systems, some challenges need to be addressed (Kocher & Kumar, 2021). For example, the black box nature of deep learning models makes it difficult to interpret the results, which is a significant issue in applications where transparency and explainability are essential, such as in medical diagnosis or legal decision-making. Additionally, the overfitting problem of deep learning in fuzzy classification problems, such as intrusion detection, needs further research.

2.5 Hybrid Fuzzy with Neural networks

The combination of fuzzy logic and neural networks, known as hybrid Fuzzy with Neural networks, has emerged as a powerful approach in various applications, including pattern recognition, classification, and control systems. This hybrid approach aims to leverage the strengths of both fuzzy logic and neural networks while mitigating their limitations.

Fuzzy logic provides a mechanism for handling uncertainty and imprecision in data, allowing for the representation of vague and fuzzy concepts. On the other hand, neural networks offer the ability to learn from data and make predictions based on learned patterns.

The hybrid Fuzzy with Neural networks approach has been successfully applied to a wide range of problems, including fault diagnosis, control systems, and pattern recognition. In the context of intrusion detection, researchers have explored different ways of incorporating hybrid Fuzzy with Neural networks, such as fuzzy rule-based classification

with neural networks, hybrid neural-fuzzy systems, and the use of Adaptive Neuro-fuzzy Inference Systems (ANFIS).

These approaches have shown promising results in improving the accuracy and efficiency of intrusion detection systems. By combining the uncertainty handling capabilities of fuzzy logic with the learning and predictive capabilities of neural networks, hybrid Fuzzy with Neural networks have the potential to enhance the performance of intrusion detection systems, enabling the detection of complex and evolving intrusion patterns.

2.5.1 Classification of Hybrid Fuzzy with Neural networks

Currently, there are three main ways to combine neural networks and fuzzy techniques: the first is the Neural-Fuzzy models; the second is the Fuzzy-Neural models; and the third is the combination of neural and fuzzy models.

The Neural-Fuzzy model combines neural networks and fuzzy control, and uses neural networks to implement the decision-making process of fuzzy control. It takes the fuzzy control method as the "sample" for offline training and learning of the neural network. All the samples are learned, and the neural network becomes a smart and flexible fuzzy rule table, with self-learning and adaptive capabilities, as shown in Figure 2.7.

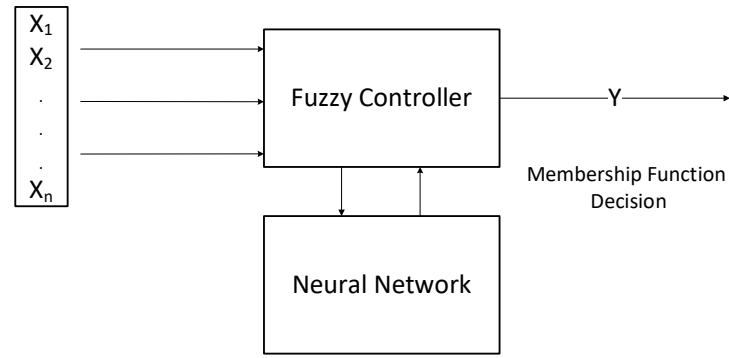


Figure 2.7: The Neural-Fuzzy Model

The Fuzzy-Neural model, on the other hand, takes neural networks as the main component, and divides the input space into different forms of fuzzy inference combinations. It first performs fuzzy logical judgment on the system and uses the output of the fuzzy controller as the input of the neural network, which has intelligent control characteristics with self-learning ability. The combination of neural and fuzzy models, as shown in Figure 2.8.

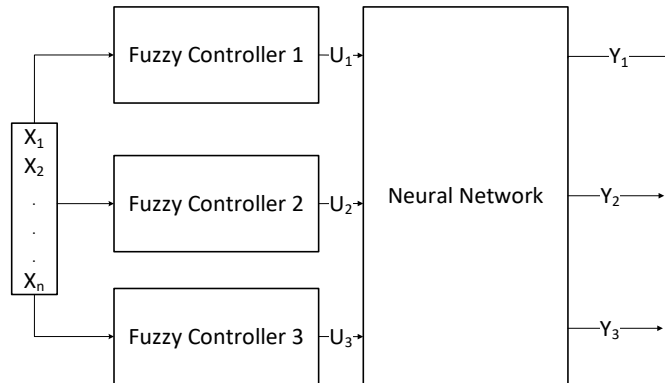


Figure 2.8: The Fuzzy-Neural Model

The combination of neural and fuzzy model integrates neural networks and fuzzy control methods by having each method process input information directly based on the

different nature of the input variables, allowing for more effective utilization of their respective control characteristics on the controlled object, as shown in Figure 2.9.

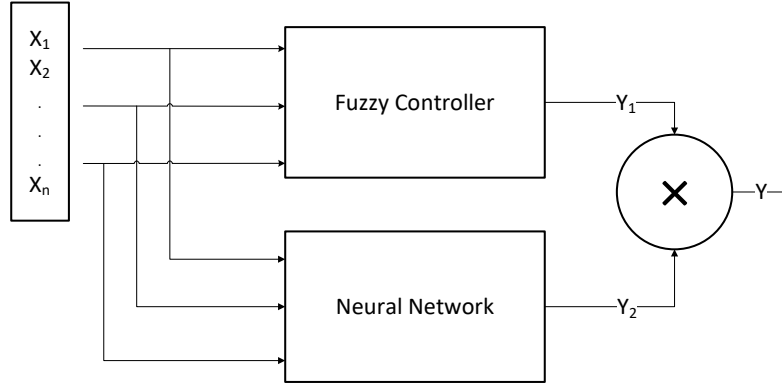


Figure 2.9: The Combination of Neural and Fuzzy Model

2.5.2 ANFIS

ANFIS (Adaptive Neuro-Fuzzy Inference System) is a hybrid approach that combines the learning capabilities of neural networks with the reasoning capabilities of fuzzy logic. It utilizes fuzzy if-then rules to represent complex nonlinear relationships between input and output variables. The key idea behind ANFIS is to adaptively adjust the parameters of these fuzzy rules using a backpropagation algorithm.

With the development of NNs, ANFIS was developed to combine the benefits of both NNs and fuzzy logic in a single system by Jang (1993). Neural networks have the ability to learn from data and extract hidden patterns, but their outputs are typically numeric and lack transparency. On the other hand, fuzzy logic can model and reason with imprecise and uncertain information in a transparent and interpretable manner, but its rule bases and membership functions are usually designed based on expert knowledge or trial and error.

ANFIS (Adaptive Neuro-fuzzy Inference System) is a hybrid computational model that combines the fuzzy logic's ability to handle uncertainty with the learning capability of

neural networks. One of the key advantages of ANFIS is that it determines the fuzzy membership function and rules through a learning process based on known data, rather than relying solely on human experience or intuition.

This learning-based approach is particularly valuable in systems with properties that are not fully understood or are highly complex. By learning from the available data, ANFIS can capture the underlying patterns and relationships in the data, even when they are not explicitly defined by human experts. This makes ANFIS well-suited for applications where there is a lack of comprehensive domain knowledge or when the system's behaviour is influenced by multiple interacting factors.

ANFIS aims to overcome the limitations of both neural networks and fuzzy logic by using a hybrid learning approach to automatically learn both the fuzzy rule base and the corresponding membership functions from data. By doing so, ANFIS can achieve a high level of accuracy and transparency, making it well-suited for many real-world applications, including intrusion detection systems.

ANFIS represents both Sugeno and Tsukamoto fuzzy models. The architecture of ANFIS is shown in Figure 2.10. There are five layers in ANIFIS:

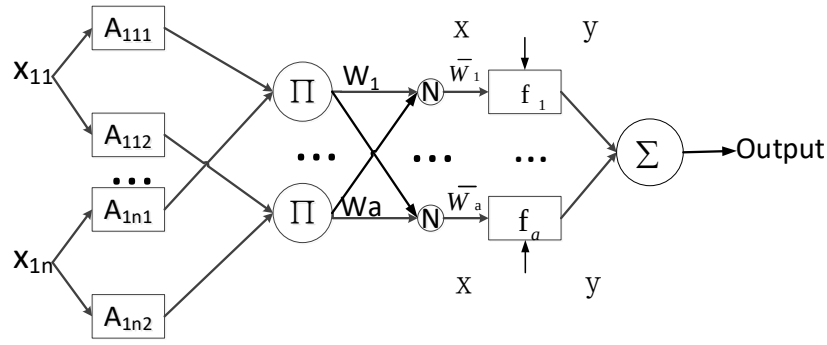


Figure 2.10: Architecture of ANFIS

Layer 1: Calculation of membership grades using the Gaussian membership function. Each input is evaluated by the Gaussian function, which is characterized by the mean (center) and sigma (width) parameters.

$$\mu_{A_{1nj}}(x_{1n}) = e^{-\frac{(x_{1n} - \text{mean}_n)^2}{\text{sigma}_n^2}} \quad \text{Equation 2.4}$$

Where x_{1n} represents the input to the node A_{1n} , which corresponds to attribute n . The linguistic label j denotes the membership function associated with attribute n , such as "low" or "high." A_{1nj} represents the degree to which the input x_{1n} satisfies the linguistic label A_{1nj} , indicating the membership value.

The premise parameters, mean_n and sigma_n , play a crucial role in the Gaussian membership function. The mean_n parameter represents the center or midpoint of the Gaussian function, determining where the function reaches its peak. The sigma_n parameter represents the width or spread of the Gaussian function, indicating how quickly the function decreases as the input moves away from the centre.

These premise parameters, mean_n and sigma_n , are learning parameters that can be adjusted during the learning process. By modifying the values of these parameters, the shape

and characteristics of the Gaussian functions can be altered. This flexibility allows ANFIS to adapt and fit the input data more effectively, enhancing its ability to capture complex relationships and handle uncertainty in fuzzy classification tasks.

Layer 2: Firing strengths. Each node in this layer calculates the firing strength of the specific i -th rule based on the membership grades obtained in Layer 1.

$$\mathbf{w}_i = \mu_{A_{11j}}(x_{11}) \times \dots \times \mu_{A_{1nj}}(x_{1n}) \quad \text{Equation 2.5}$$

Where i is the node number of Layer2. Each node output represents the firing strength of a rule.

Layer 3: Normalization of firing strengths. The firing strengths obtained in Layer 2 are normalized to ensure that they sum up to 1, representing the relative importance of the i -th rule.

$$\bar{w}_i = \frac{\mathbf{w}_i}{\sum_{i=1}^a \mathbf{w}_i} \quad \text{Equation 2.6}$$

Where a is the number of nodes in Layer2.

Layer 4: Adaptive nodes with linear functions. Each node in this layer calculates the weighted value of the consequent part of each rule. The consequent parameters (p_i, q_i, r_i) are the learning parameters that determine the linear function's behaviour.

$$\bar{w}_i \mathbf{f}_a = \bar{w}_i (\mathbf{p}_i \mathbf{x} + \mathbf{q}_i \mathbf{y} + \mathbf{r}_i) \quad \text{Equation 2.7}$$

Layer 5: Aggregation of rule outputs. The overall output is generated by aggregating the outputs of all the fired rules, taking into account their firing strengths.

$$\text{output} = \sum_i \bar{w}_i \cdot f_i \quad \text{Equation 2.8}$$

In ANFIS, the learning process involves adjusting the premise and consequent parameters. The premise parameters, including the mean and sigma values of the Gaussian membership functions, are updated using gradient descent during the backward propagation process. The consequent parameters, (p_i, q_i, r_i) , are estimated using the least-squares method during the forward-passing process.

This hybrid learning mechanism, which combines forward-passing and backward propagation, allows ANFIS to learn from data and optimize its parameters to approximate the desired output, as shown in Table 2.5. By adjusting the premise and consequent parameters, ANFIS adapts to the given input-output relationship, making it a powerful tool for modelling complex systems.

Table 2.5: Two Passes in The Hybrid Learning Algorithm for ANFIS

Parameters	Forward Pass	Backward Pass
Premise Parameters	Fixed	Gradient Descent
Consequent Parameters	Least-squares estimator	Fixed
Signals	Node outputs	Error signals

In practical applications, the ANFIS learning algorithm emphasizes the aspect of parameter learning in the controller, since the network structure has already been established and only the premise and conclusion parameters require adjustment.

There are four methods used to update parameters, listed below in increasing order of complexity:

- i. Gradient Descent Only: In this approach, all parameters, including both premise and consequent parameters, are updated using gradient descent.
- ii. Gradient Descent and Once Pass of LSE (Least Squares Estimate): Initially, the least squares estimate is used to obtain the initial values of the consequent parameters. This estimate is performed only once at the beginning of the training process. Afterward, gradient descent is employed to update all parameters, including both premise and consequent parameters. The least squares estimate provides an initial approximation, and then gradient descent is utilized for further fine-tuning.
- iii. Gradient Descent and LSE: This approach combines both gradient descent and the least squares estimate. The training process involves iteratively updating the parameters using gradient descent while also periodically performing the least squares estimate to refine the consequent parameters. The combination of these two methods allows for a more comprehensive and refined learning process.
- iv. Sequential (Approximate) LSE Only: Only the recursive (approximate) least squares estimate is used.

Toosi and Kahani (2007) introduced the use of ANFIS in intrusion detection by proposing an evolutionary method. They trained multiple ANFIS modules to detect intrusive

activity and incorporated a fuzzy inference module for the final decision-making process based on empirical knowledge.

ANFIS has gained attention as a promising hybrid model that combines the strengths of fuzzy logic and neural networks. However, it also has limitations, as illustrated in Section 1.2. One limitation is the restricted number of layers in ANFIS, which may hinder its ability to represent complex relationships effectively. Additionally, ANFIS can generate a large number of rules, making it challenging to interpret and understand the decision-making process.

These limitations highlight the importance of ongoing research and development in the field of hybrid models to address these challenges and enhance the performance of ANFIS in practical applications.

2.5.3 CART

CART provides a natural, intuitive way to visualize and understand the decision-making process. This makes it easier for analysts to understand how the model is making predictions, which in turn builds trust in the system.

CART (Breiman et al., 2017) is a popular decision tree algorithm that has several advantages over other decision tree algorithms, such as ID3 (Quinlan, 1986) and C4.5 (Quinlan, 1993). One of its key strengths is its ability to handle both categorical and continuous data types. CART constructs binary decision trees by recursively splitting the data set into two subsets based on the attribute that provides the most information gain or the smallest Gini index. Another advantage of CART is its ability to handle missing values, where the algorithm uses surrogate splits to replace missing values during the construction of the decision tree.

Additionally, CART has a straightforward interpretation, where the decision tree can be easily visualized and understood. However, one of the main limitations of CART is its tendency to overfit the training data, especially for complex decision trees.

It follows a binary splitting process, where each internal node of the tree represents a decision based on a selected attribute and its corresponding threshold value. The Gini index is commonly used in CART to evaluate the impurity of a node and determine the best attribute to split the data. The Gini index is a measure of inequality that ranges from 0 to 1, and it is used to determine the optimal feature to split the data. A Gini index of 0 means that the distribution is perfectly equal, while a Gini index of 1 means that the distribution is completely unequal.

To determine the Gini index of a feature in the dataset, the Gini index is calculated for all values belonging to that feature, as shown in Equation 2.9. The feature with the smallest GiniGain is selected as the split point, and this process is recursively continued until one of the following conditions is met:

- i. All subsets belong to the same class: If all the instances in a given subset have the same class label, no further splitting is necessary, and a leaf node is created with that class label.
- ii. All features have been used: If all the available features have been used for splitting the data, but the subsets still contain instances from multiple classes, a leaf node is created with the majority class label or by using some other predefined criterion.

For example, the Gini index for a given feature is calculated based on the probability of an instance being misclassified if it is randomly assigned to a class based on the distribution of classes in that feature. The Gini index can be computed as Equation 2.9.

$$Gini(p) = \sum_{k=1}^K p_k(1 - p_k) = 1 - \sum_{k=1}^K p_k^2 \quad \text{Equation 2.9}$$

Where p_k represents the probability of an instance belonging to the k -th class. $Gini(p)$ represents the Gini index of the probability distribution.

Equation 2.10 shows how the Gini index of a sample set D can be calculated by summing the Gini index of each class subset C_k within the sample set.

$$Gini(D) = 1 - \sum_{k=1}^K \left(\frac{|C_k|}{|D|} \right)^2 \quad \text{Equation 2.10}$$

Where $Gini(D)$ is the Gini index of sample set D . C_k represents the sample subset belonging to the k -th class in sample set D .

If sample set D is segmented on feature A 's certain value a , obtaining parts D_1 and D_2 (sample set D 's two segments divided according to feature A). The GiniGain of set D under feature A is shown as Equation 2.11.

$$Gain_Gini(D, A) = \frac{|D_1|}{|D|} Gini(D_1) + \frac{|D_2|}{|D|} Gini(D_2) \quad \text{Equation 2.11}$$

where $Gain_Gini(D, A)$ represents the uncertainty of set D after $A = a$ segmentation. The greater the value, the greater the uncertainty of the sample set.

For feature A , the $Gain_Gini$ after dividing the dataset into two parts by every feature value is calculated respectively, and the minimum value is selected as the optimal binary scheme obtained by feature A , as shown in Equation 4.6.

$$\min_{i \in A}(\text{Gain_Gini}(D, A)) \quad \text{Equation 2.12}$$

Then, for sample set D , the optimal binary scheme of all features is calculated, and the minimum value is selected as the optimal binary scheme of sample set D , as shown in Equation 2.13.

$$\min_{A \in \text{Attribute}} (\min_{i \in A}(\text{Gain_Gini}(D, A))) \quad \text{Equation 2.13}$$

2.5.4 Other Hybrid Fuzzy with Neural Networks

Regarding other research apart from ANFIS on the combination of fuzzy logic and neural networks, Toosi and Kahani (2007) propose the integration of multiple soft computing techniques, including neuro-fuzzy networks, fuzzy inference approach, and genetic algorithms, to enhance the accuracy of intrusion detection systems. Their study outlines the use of parallel neuro-fuzzy classifiers for initial classification, followed by the fuzzy inference system, which utilizes the output of the neuro-fuzzy classifiers to make the final decision on whether an activity is normal or intrusive. To further optimize the performance of the system, a genetic algorithm is employed to optimize the structure of the fuzzy decision engine. The proposed method is evaluated using the KDDCUP99 intrusion detection dataset. The study suggests that the incorporation of various soft computing techniques can significantly improve the accuracy and efficiency of intrusion detection systems.

Wang et al. (2010) present a novel approach, FC-ANN, that integrates ANN with fuzzy clustering to enhance the performance of IDS. The main goal of this approach is to improve the detection rate, reduce false positives, and increase system stability. The FC-ANN approach in intrusion detection involves utilizing fuzzy clustering to generate distinct

training subsets. These subsets are then employed to train separate artificial neural network (ANN) models, resulting in the creation of diverse base models. Finally, a meta-learner, the fuzzy aggregation module, is used to aggregate the results of the different base models. Experimental results on the KDDCUP99 dataset show that the FC-ANN approach outperforms traditional methods such as decision trees, naïve Bayes, and BPNN in terms of detection precision and stability.

Pandeewari and Kumar (2016) have proposed a hybrid algorithm, Fuzzy C-Means clustering algorithm and Artificial Neural Network (FCM-ANN), for improving the accuracy of IDSs in cloud computing systems. They have named their proposed system, which operates at the hypervisor layer, as Hypervisor Detector. The system's performance has been evaluated using DARPA's KDDCUP99 dataset. The experimental results of this study highlight the superior performance of the Hypervisor Detector compared to the Naïve Bayes classifier and Classic ANN algorithm in detecting anomalies, even for low-frequency attacks. The Hypervisor Detector exhibits high accuracy and low false alarm rates, showcasing its effectiveness in enhancing the accuracy and performance of Intrusion Detection Systems (IDS) in cloud computing environments. This study emphasizes the potential of hybrid algorithms, such as the Hypervisor Detector, in addressing the unique challenges and improving the security of cloud computing systems.

Ashfaq et al. (2017) introduced an innovative fuzziness-based semi-supervised learning (SSL) approach for Intrusion Detection Systems (IDSs). The proposed method leverages unlabeled samples in conjunction with a supervised learning algorithm to enhance the performance of the classifier. The approach employs a single hidden layer feed-forward neural network (SLFN) to generate a fuzzy membership vector for each sample. Based on

these fuzzy membership values, the unlabeled samples are categorized into three groups: low fuzziness, mid fuzziness, and high fuzziness. Each category is then separately incorporated into the original training set, and the classifier is retrained accordingly. Experimental evaluations conducted on the NSL-KDD dataset demonstrate the effectiveness of incorporating unlabeled samples from low and high fuzziness groups into the training set. This integration significantly improves the performance of the classifier compared to traditional classifiers, such as Naive Bayes, Support Vector Machine, and Random Forests.

Lin et al. (2022) focus on the application of fuzzy logic-based techniques for the detection of network traffic anomalies and Distributed Denial of Service (DDoS) attacks. The survey systematically categorizes the reviewed approaches based on the specific fuzzy techniques utilized to address network anomalies and DDoS attacks. The survey underscores the wide array of fuzzy network anomaly detection approaches, which employ a diverse set of techniques such as classifiers, feature selection/extraction methods, and statistical and clustering algorithms. These methods leverage the power of fuzzy logic to handle uncertainties and imprecisions inherent in network data, facilitating robust and accurate detection of anomalous traffic.

Alrayes et al. (2023) introduce a novel approach called MFSFL-IDS for intrusion detection in the Internet of Things (IoT) environment, leveraging a combination of metaheuristics feature selection and fuzzy logic techniques. The main objective of MFSFL-IDS is to identify and prevent intrusions, ensuring enhanced security for IoT devices. The MFSFL-IDS model consists of several stages. Initially, data pre-processing is performed to transform the raw data into a usable format. Next, the authors employ the Henry gas solubility optimization (HGSO) algorithm as a feature selection technique to identify

relevant feature vectors from the data. Subsequently, the Adaptive Neuro-Fuzzy Inference System (ANFIS) is employed for intrusion recognition and classification within the network. ANFIS leverages fuzzy logic principles to model complex relationships between input features and intrusion classes, enabling accurate and efficient intrusion detection. To optimize the parameters in the ANFIS model, the binary bat algorithm (BBA) is applied. BBA is a metaheuristic optimization algorithm that iteratively searches for the optimal parameters to enhance the performance of the ANFIS model.

Neural networks are recognized for their capacity to learn from data and generate predictions, while fuzzy logic is acknowledged for its ability to manage uncertainty and imprecision in data. In the hybrid Fuzzy with Neural Networks, a fuzzy logic system is incorporated into a neural network to assist in handling uncertainty in the data. The fuzzy logic system offers the ability to reason about imprecise and uncertain data, while the neural network provides robust pattern recognition and modeling capabilities. The two components are combined in a manner that capitalizes on the strengths of each to improve overall system performance.

However, the hybrid Fuzzy with Neural Networks poses several challenges, as described in Section 1.2, including the need to optimize the integration of the two approaches to achieve better performance. Another significant challenge is the interpretability of resulting models, as neural networks are often seen as black box models. Overfitting is also a common problem, particularly in deep fuzzy classification problems. Therefore, further research is necessary to address these challenges and to explore the potential of hybrid models for applications such as intrusion detection and other related fields.

In conclusion, the hybrid approach of combining Fuzzy Logic with Neural Networks shows great promise in addressing complex and uncertain problems, particularly in the domain of intrusion detection. By leveraging the strengths of both fuzzy logic and neural networks, this approach enhances the accuracy and robustness of intrusion detection systems, enabling them to effectively handle the dynamic and evolving nature of network security threats. The integration of fuzzy logic's ability to handle uncertainty and imprecision with the learning capabilities of neural networks provides a powerful solution for tackling intrusion detection challenges. This hybrid approach holds significant potential for advancing the field of network security and improving the overall effectiveness of intrusion detection systems.

2.6 Chapter Summary

This study analyses the characteristics of intrusion detection and reviews the application of various machine learning algorithms in intrusion detection. For the fuzzy characteristics of intrusion detection systems, this study intends to adapt the fuzzy characteristics of intrusion detection through hybrid ANFIS model to enhance the performance in intrusion detection.

The classification and recognition of intrusion detection have fuzzy characteristics. Intrusion detection is a preventive technology aimed at detecting unauthorized access attempts before damage occurs, and can provide real-time or near-real-time alarms (Stallings & Lawrie, 2017). According to Anderson (1980), this technology assumes that there is a significant difference between the patterns of legitimate users and intruders. By establishing the normal pattern of legitimate users and the abnormal pattern of intruders through historical record observation, intrusions can be identified and blocked in advance, resulting in lower

damage. However, as the behaviours of intruders often overlap with those of legitimate users, intrusion detection involves a certain degree of misjudgement and uncertainty. Accounting for this factor during classification can lead to improved intrusion detection performance.

Fuzzy logic is a powerful weapon that adapts to the fuzzy characteristics of intrusion detection. Fuzzy logic systems are designed to mimic human-like reasoning through the use of linguistic expressions and have demonstrated success in fuzzy classification problems like intrusion detection.

In various fuzzy-based intrusion detection systems, ANFIS is one of the most utilized algorithms. ANFIS integrates the learning capabilities of neural networks with the uncertainty-handling abilities of fuzzy logic. It allows for flexibility in dealing with varying degrees of uncertainty and vagueness present in network traffic and attack patterns. This adaptability is crucial in complex network environments where attack patterns and behaviors can vary widely and are often difficult to define precisely. It incorporates the learning capabilities of neural networks, allowing it to adapt and refine its models based on new data and evolving attack patterns. This ability to learn and adapt makes ANFIS a dynamic tool that can evolve and improve over time, providing more accurate and effective intrusion detection. Meanwhile, ANFIS's advantage of having fewer layers leads to better interpretability. This means that the model's structure is simpler and easier to understand, making it more transparent and reliable for users.

Nevertheless, ANFIS does have limitations, including a restricted number of layers and the generation of numerous rules. The limited number of layers in ANFIS can indeed be a constraint when dealing with complex intrusion detection tasks that require a deeper and

more hierarchical representation of data. The shallow architecture of ANFIS might not be able to effectively capture the hierarchical patterns or long-range dependencies present in network traffic or attack behavior. Additionally, the training process of ANFIS often results in a large number of generated rules. While this increased flexibility and coverage can be beneficial, it also brings challenges in terms of rule management and training efficiency in high-dimensional data.

To overcome these limitations, researchers have attempted to enhance the performance by using other algorithms. For instance, some studies have employed evolutionary algorithms, particle swarm optimization, and ant colony optimization to improve the learning and decision-making process of the ANFIS algorithm. These enhanced algorithms have shown promising results in various applications, including intrusion detection, fault diagnosis, and prediction. However, it does not solve the problem of limited levels in ANFIS, and there is still a lack of ability to deal with complex intrusion behaviors.

In summary, ANFIS is an effective algorithm for adapting to fuzzy characteristics in intrusion detection. It exhibits learning capabilities, robustness, and ease of interpretability. However, it also has certain limitations, such as a lack of deep pattern recognition capabilities and an excessive number of redundant rules. Currently, combining ANFIS with other algorithms provides a possibility for enhancing its performance in intrusion detection, and it also offers a feasible approach to address the increasingly complex issues of intrusion detection.

CHAPTER 3

ENVIRONMENT AND EXPERIMENTAL DESIGN

3.1 Overview

This chapter provides an overview of the dataset, environment, metrics, and experimental methodologies utilized in this study. The modified algorithm's architecture and experimental procedure are first introduced. Next, the NSL-KDD dataset, used in the experiments, is presented. Finally, the software tools, platforms and evaluation indicators employed in the study are outlined.

3.2 Experimental Approaches

To compensate for the limited layers of ANFIS and maintain interpretability, as discussed in Section 1.2, CART-enhanced ANFIS was proposed. Meanwhile, in order to identify increasingly complex intrusion behaviors in complex network environments, as mentioned in Section 1.2, ResNet-enhanced ANFIS was proposed, generating a deep fuzzy network structure. Both of these approaches involve deepening the structure of ANFIS, with the ResNet-enhanced ANFIS having more layers than the CART-enhanced ANFIS.

However, due to the interpretable structures of the two algorithms combined in CART-enhanced ANFIS, its structure is more interpretable. ResNet-enhanced ANFIS aims to achieve deeper levels and employs a black-box ResNet algorithm to enhance ANFIS, resulting in a deeper network structure.

The experimental process is illustrated in Figure 3.1 and consists of three main stages: data extraction, data classification, and performance testing. In the data pre-processing stage, feature selection methods are employed to minimize the number of

attributes. Once this is completed, the classification stage commences, where proposed hybrid ANFIS model is trained. The performance of this model is then evaluated in the performance testing stage. Depending on the results of this testing, the proposed hybrid ANFIS model will undergo further improvement.

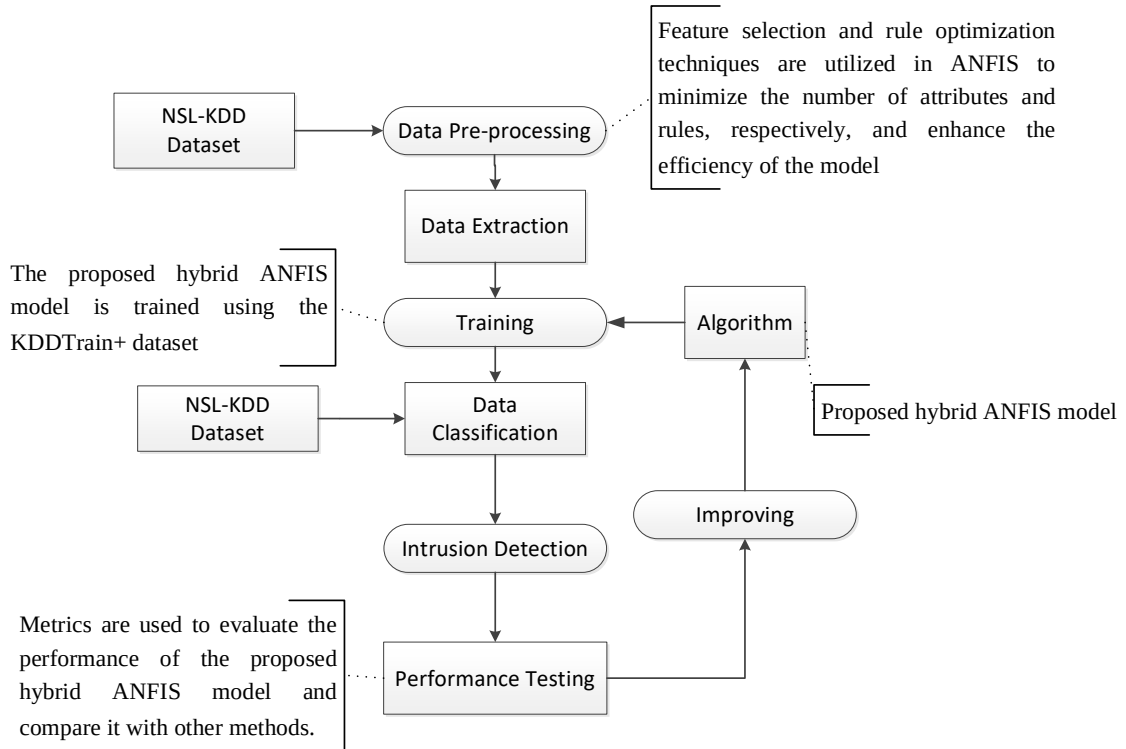


Figure 3.1: Processes in Conducting the Experiment

Data generation in this study includes converting symbolic features to numerical values and normalizing numerical features. The NSL-KDD dataset contains several character-type features, such as protocol_type, service, flag, land, target, and so on. In this research, these features are numerically encoded based on the order of occurrence in the dataset. The original dataset contains numerical features with varying ranges, therefore all features are normalized prior to training. Each numerical feature is standardized, and then scaled to the range of [0,1].

During the data classification stage, the KDDTrain+ dataset is split into two parts. The first part, which comprises 80% of the dataset, is used as the training set for the algorithm model. The second part, which is the full KDDTrain+ dataset, is used as the validation dataset. This ensures that the training set and validation dataset have the same distribution. Then the algorithm model is trained by training dataset.

In the performance testing stage, the trained model will be evaluated using the validation dataset and test dataset to assess its performance. The experimental results will then be analyzed and compared to determine the effectiveness of the proposed methods.

The first improvement aims to enhance the deep pattern extraction ability of ANFIS by combining the advantages of ANFIS and CART to design a deeper and more interpretable architecture. This is based on a study of the application of ANFIS and CART in intrusion detection, as shown in Figure 3.2.

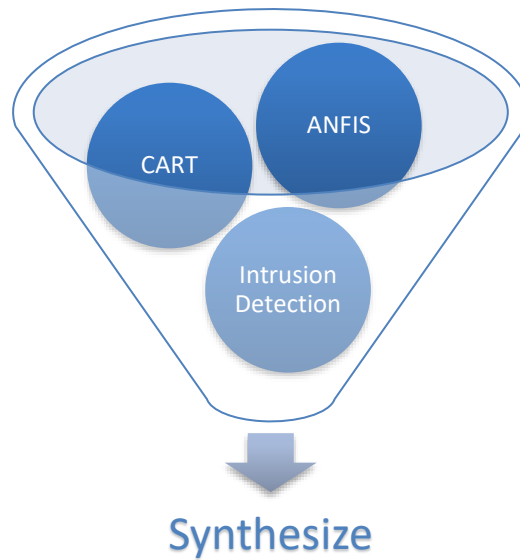


Figure 3.2: Idea of The First Improvement

ANFIS is a fuzzy neural network that is particularly suitable for dealing with problems with uncertainty and ambiguity. It uses fuzzy logic and fuzzy set theory to deal with ambiguity, and adapts through the learning ability of neural networks. In intrusion detection, ANFIS can effectively extract useful features from complex network traffic data, and classify intrusion behaviors based on these features.

CART is a decision tree learning method that uses the construction of a tree to classify or regress data. CART has good performance in dealing with complex problems and the generated model is easy to understand and interpret. In intrusion detection, CART can be used to construct an efficient classifier to distinguish normal network traffic from abnormal traffic.

By combining CART and ANFIS, we enhance each other's depth and pattern recognition capabilities, designing a deeper and more interpretive network structure to improve ANFIS's pattern extraction ability. The goal of this improvement is to leverage ANFIS's fuzzy logic processing capabilities and CART's classification capabilities to create a more powerful and flexible model that can effectively detect intrusions in complex network environments.

The second improvement combines the advantages of ANFIS and deep learning algorithms to design a highly deep structure for intrusion detection, as shown in Figure 3.3.

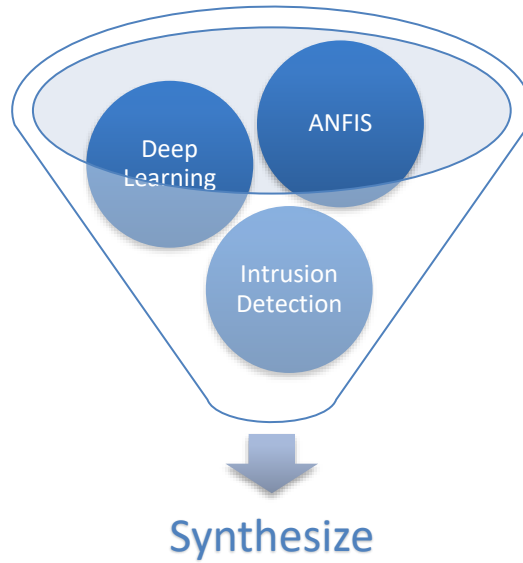


Figure 3.3: Idea of The Second Improvement

Deep learning is a powerful machine learning method that uses deep neural networks to process and analyze data. In the field of intrusion detection, deep learning algorithms have been proven to be effective in identifying complex network attacks. By combining ANFIS with deep learning algorithms, we design a highly deep network structure to enhance the ability of intrusion detection. The purpose of this combination is to leverage the fuzzy logic processing capabilities of ANFIS and the powerful learning capabilities of deep learning algorithms to jointly address the constantly changing intrusion behaviors in complex network environments.

Specifically, ANFIS deals with the uncertainty and ambiguity of data at the fuzzy logic level, while deep learning algorithms extract high-level features and patterns from data by constructing multilayer neural networks. This combination allows us to design deeper network structures to analyze network traffic data more deeply and identify potential intrusions more accurately.

3.3 Dataset Analysis

The NSL-KDD dataset (Tavallaee et al., 2009), which is widely used as a benchmark in intrusion detection research, is publicly available and can be accessed from the official website of the Canadian Institute for Cybersecurity (CIC) at <https://www.unb.ca/cic/datasets/nsl.html>. It is a widely used dataset in intrusion detection research, making it a suitable choice for comparing and evaluating different algorithms. It covers a wide range of attacks and normal traffic scenarios, providing a diverse and representative dataset for testing intrusion detection algorithms.

This dataset is an improved and refined version of the KDDCUP99 dataset from the University of New Brunswick, which was initially derived and processed from the DARPA98 IDS data that contained various intrusions that were simulated in a military network environment. The LAN simulation captured nine weeks of raw TCP dump data and was designed by Lincoln Labs to emulate a typical U.S. Air Force LAN while simultaneously experiencing multiple attacks (Huang, 2018).

The NSL-KDD dataset consists of 8 files, as outlined below. This dataset includes four subsets, namely KDDTest+, KDDTest-21, KDDTrain+, and KDDTrain+_20Percent, as shown below. Notably, KDDTest-21 and KDDTrain+_20Percent are extracted from KDDTest+ and KDDTrain+, respectively.

- i. KDDTrain+: The full NSL-KDD train set including attack-type labels and difficulty level.
- ii. KDDTrain+_20Percent: A 20% subset of the KDDTrain+.

- iii. KDDTest+: The full NSL-KDD test set including attack-type labels and difficulty level.
- iv. KDDTest-21: A subset of the KDDTest+ which does not include records with difficulty level of 21 out of 21.

The NSL-KDD dataset is widely known for its comprehensive representation of various network attacks and normal traffic. It consists of 43 attributes, including 41 normal attributes, 1 target attribute, and 1 hard attribute. Among the normal attributes, there are 7 discrete attributes and 34 continuous attributes.

The dataset includes two main subsets: KDDTrain+ and KDDTest+. These subsets differ in terms of traffic distribution, making the NSL-KDD dataset suitable for evaluating the generalization ability of intrusion detection algorithms. The KDDTest+ subset contains some new intrusion types that are not present in the KDDTrain+ subset.

The NSL-KDD dataset classifies attacks into four distinct categories: Denial of Service (DoS), PROBE, User to Root (U2R), and Remote to Local (R2L). These categories cover a wide range of attack types and scenarios.

In the KDDTrain+ subset, there are a total of 23 classes, including 22 types of attacks and normal traffic. The KDDTest+ subset expands to 38 classes, consisting of 37 types of attacks and normal traffic. Both subsets maintain the four attack categories: DoS, PROBE, U2R, and R2L, as below:

- i. DoS: This is an attack that attempts to shut down traffic to and from a target system.

- ii. PROBE: This is an attack that attempts to extract information from the network.
- iii. U2R: This is an attack that starts with a regular user account and attempts to access a system or network as a superuser (root).
- iv. R2L: This is an attack that attempts to gain local access to a remote computer.

The traffic distribution of the KDDTrain+ and the KDDTest+ subsets of the NSL-KDD dataset is presented in Table 3.1.

Table 3.1: Distribution of Major Categories in KDDTrain+ and KDDTest+ Subsets

Category	KDDTrain+ dataset		KDDTest+	
	Count	Percentage	Count	Percentage
Normal	67342	53.46%	9711	43.08%
Probe	45927	36.46%	7457	33.08%
DOS	11656	9.25%	2754	12.22%
U2R	995	0.79%	2421	10.74%
R2L	52	0.04%	200	0.89%
Total	125972	100%	22543	100%

In this study, the training dataset consisted of an 80% subset of KDDTrain+, while the remaining 20% of KDDTrain+ and the complete KDDTest+ subset was used as the validation and test datasets, respectively (Chidananda Murthy et al., 2016).

Both the KDDTrain+ and KDDTest+ datasets contain a total of 43 attributes, including 41 normal features, 1 target attribute, and 1 hard attribute. These 41 normal

features can be classified into four categories. The first category is basic features, which includes 9 attributes, such as duration, protocol type, and number of bytes transmitted, as presented in Table 3.2. The second category is content features, which consists of 13 attributes to identify attacks like U2R and R2L that lack frequent sequence patterns in the data record, as illustrated in Table 3.3. The third category is traffic features based on time, which comprises 9 features that observe connections with similar characteristics as the current connection in the past 2 seconds, as shown in Table 3.4. The fourth category is traffic features based on the host, which consists of 10 features that compare the 100 previous connections to the current connection with the same destination host, as demonstrated in Table 3.5.

Table 3.2: Basic Features

No.	Feature Name	Description	Type
1	duration	length (number of seconds) of the connection	continuous
2	protocal_type	type of the protocol, e.g. tcp, udp, etc.	discrete
3	service	network service on the destination, e.g., http, telnet, etc.	discrete
4	flag	normal or error status of the connection	discrete
5	src_bytes	number of data bytes from source to destination	continuous
6	dst_bytes	number of data bytes from destination to source	continuous
7	land	1 if connection is from/to the same host/port; 0 otherwise	discrete
8	wrong_fragment	number of ``wrong" fragments	continuous
9	urgent	number of urgent packets	continuous

Table 3.3: Content Features

No.	Feature Name	Description	Type
1	hot	number of ``hot" indicators	continuous
2	num_failed_logins	number of failed login attempts	continuous
3	logged_in	1 if successfully logged in; 0 otherwise	discrete
4	num_compromised	number of ``compromised" conditions	continuous
5	root_shell	1 if root shell is obtained; 0 otherwise	discrete
6	su_attempted	1 if ``su root" command attempted; 0 otherwise	discrete
7	num_root	number of ``root" accesses	continuous
8	num_file_creations	number of file creation operations	continuous
9	num_shells	number of shell prompts	continuous
10	num_access_files	number of operations on access control files	continuous
11	num_outbound_cmds	number of outbound commands in an ftp session	continuous
12	is_hot_login	1 if the login belongs to the ``hot" list; 0 otherwise	discrete
13	is_guest_login	1 if the login is a ``guest"login; 0 otherwise	discrete

Table 3.4: Traffic Features Based on Time (Two-second Time Window)

No.	Feature Name	Description	Type
The source host features based on TCP connection			
1	count	number of connections to the same host as the current connection in the past two seconds	continuous
2	serror_rate	% of connections that have ``SYN" errors	continuous
3	rerror_rate	% of connections that have ``REJ" errors	continuous
4	same_srv_rate	% of connections to the same service	continuous
5	diff_srv_rate	% of connections to different services	continuous
6	srv_count	number of connections to the same service as the current connection in the past two seconds	continuous
7	srv_serror_rate	% of connections that have ``SYN" errors	continuous
8	srv_rerror_rate	% of connections that have ``REJ" errors	continuous
9	srv_diff_host_rate	% of connections to different hosts	continuous

Table 3.5: Traffic Features Based on Host (100-connection Window)

No.	Feature Name	Description	Type
The destination host features based on TCP connection			
1	dst_host_count	number of connections to the same host	continuous
2	dst_host_srv_count	number of connections to the same service	continuous
3	dst_host_same_srv_rate	% of connections to the same service	continuous
4	dst_host_diff_srv_rate	% of connections to different services	continuous
5	dst_host_same_src_port_rate	% of connections to the same port	continuous
6	dst_host_src_diff_host_rate	% of connections that to the same service, but have different source host	continuous
7	dst_host_serror_rate	% of connections that have ``SYN" errors	continuous
8	dst_host_srv_serror_rate	% of connections that to the same service and have ``SYN" errors	continuous
9	dst_host_rerror_rate	% of connections that have ``REJ" errors	continuous
10	dst_host_srv_rerror_rate	% of connections that to the same service and have `` REJ " errors	continuous

The NSL-KDD dataset comprises 41 normal features distributed into four categories. However, some of these features may not be relevant to the intrusion detection algorithm and may impede its efficiency, possibly leading to biased results. Figure 3.4 illustrates the Pearson correlation matrix of all 43 attributes, including the 41 normal features, 1 target feature, and 1 hard feature, the full figure is shown in Appendix E.

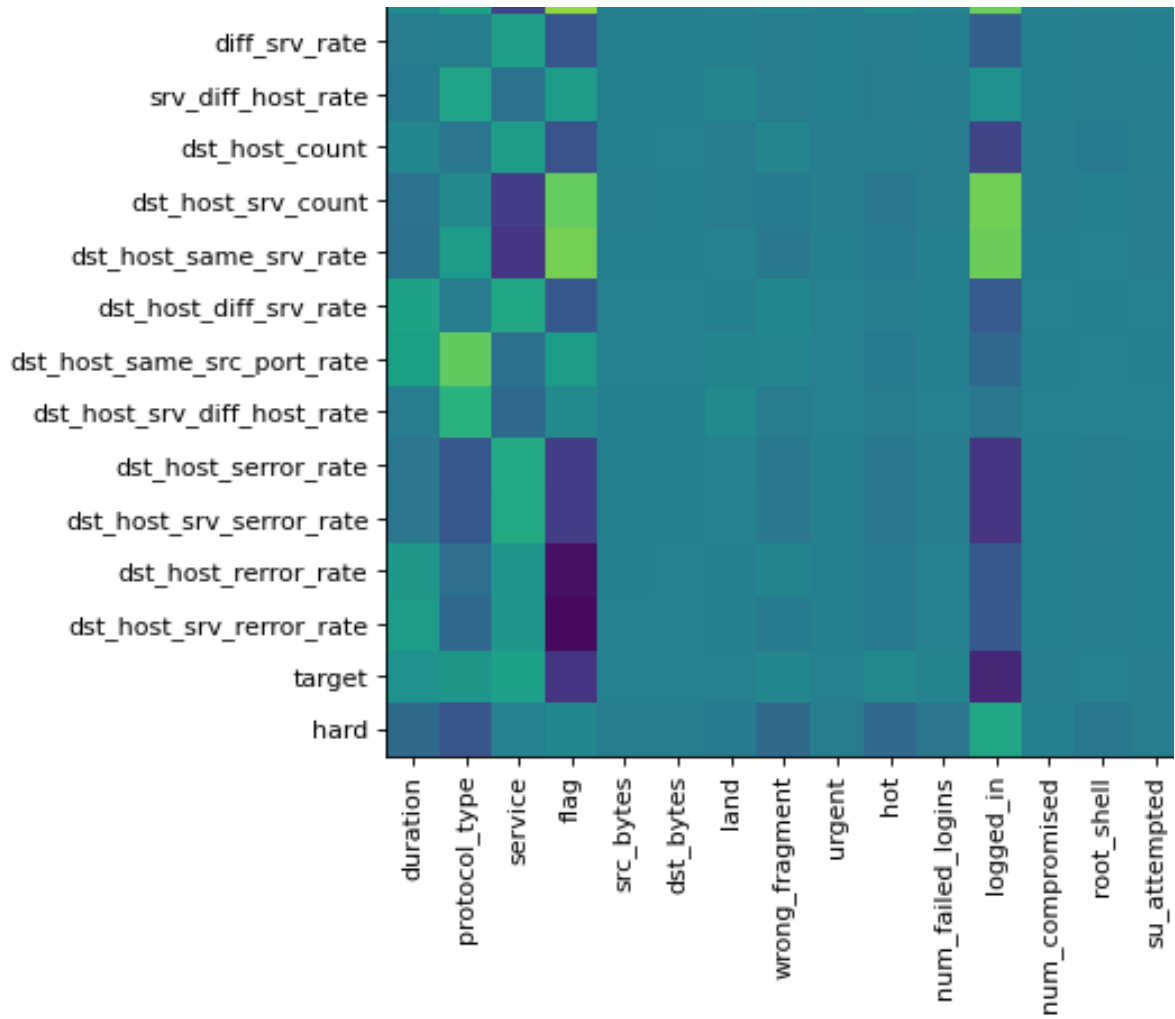


Figure 3.4: Part of Correlation Matrix of 43 Attributes

In this study, attribute selection will be conducted using Pearson correlation analysis to identify the most relevant attributes. The proposed model will then be trained on an 80% subset of the KDDTrain+ dataset. The full KDDTrain+ subset and the KDDTest+ subset will be used as validation and test datasets, respectively, to evaluate the performance of the proposed model.

3.4 Experimental Environment

This study employs the sklearn library as the foundation for implementing fuzzy logic and machine learning algorithms, and PyTorch as the foundation for implementing deep learning, as presented in Table 3.6. The programming language used for this research is Python 3, as indicated in Table 3.6.

Table 3.6: Language & Framework of Experiment

No.	Language & Framework	Version
1	Python	3.7.6
2	sklearn	0.23.1
3	PyTorch	1.5.0

Sklearn is a machine-learning library based on Python language. It covers data pre-processing to training models. Sklearn has methods that can be used for both supervised and unsupervised learning. Most of the functions in sklearn can be classified as Estimators and Transformers. Estimators are models, which are used for the prediction or regression of data. Transformers are used for data processing, such as standardization, dimensional reduction, feature selection, and so on.

PyTorch is a Python-based deep learning framework that was created as an open-source alternative to Torch. It is a tensor library that provides support for manipulating multi-dimensional matrix data. Unlike Tensorflow, which uses a static computational graph, PyTorch's computational graph is dynamic and can be modified in real-time based on the needs of the computation. PyTorch is built around three levels of abstraction that range from

low to high: tensor, automatic differentiation, and neural network. These three levels of abstraction are closely connected and can be modified and operated on simultaneously.

3.5 Evaluation Indicators

In intrusion detection, samples that contain attacks are considered positive samples, while samples without attacks are considered negative samples (Neu et al., 2022). The classification results can be categorized into four situations, as presented in Table 3.7.

Table 3.7: Four Situations of Results

Classification Sample	Positive	Negative
Positive	True Positive (TP)	False Negative (FN)
Negative	False Positive (FP)	True Negative (TN)

The following indicators can be calculated by TP, TN, FP and FN (Neu et al., 2022):

- i. Precision: The metric that measures the proportion of true positive (TP) samples to all positive classifications (TP + FP). It represents the accuracy of positive predictions made by a classification model.

$$Precision = \frac{TP}{TP + FP} \quad \text{Equation 3.1}$$

- ii. Recall rate: Recall rate, also known as sensitivity or true positive rate, measures the proportion of true positive (TP) samples to all samples that should be positive (TP + FN). It represents the ability of a classification model to correctly identify positive samples out of all the samples that should be positive.

$$Recall = \frac{TP}{TP + FN} \quad \text{Equation 3.2}$$

- iii. Accuracy (ACC): Accuracy (ACC) is a commonly used performance metric in classification tasks. It measures the proportion of correct predictions, which includes both true positives (TP) and true negatives (TN), out of all samples.

$$ACC = \frac{TP + TN}{TP + FN + FP + FN} \quad \text{Equation 3.3}$$

- iv. F1-score: The F1-score is a metric that combines precision and recall providing a balanced measure of classification performance. It is the harmonic mean of precision and recall, and it ranges from 0 to 1. The F1-score is particularly useful when dealing with imbalanced datasets, where the number of samples in different classes is significantly unequal.

$$F_1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad \text{Equation 3.4}$$

- v. Macro avg: A measure that calculates the arithmetic average of precision, recall, and F1-score across all classes in a multi-class classification problem.
- vi. Weighted avg: The support-weighted (support refers to the number of the original real data belonging to this class) average of precision, recall score and F1-score for all classes.

This study aims to compare the performance of the proposed hybrid ANFIS model with other algorithms, including the original ResNet, BPNN (Back Propagation Neural Network), GA-DBN, and DNN, using the metrics mentioned before.

3.6 Chapter Summary

This chapter provides an overview of the experimental data analysis, the experimental environment, the evaluation metrics, and the experimental procedures used in this study. The KDDTrain+ dataset was employed for training and validation, while the full KDDTrain+ dataset and the KDDTest+ dataset were used for validation and testing, respectively. The research was conducted in Python, utilizing the sklearn and PyTorch as the fundamental algorithmic support structure. To enhance ANFIS's ability to detect intrusion patterns while maintaining its interpretability, the CART-enhanced ANFIS was proposed. Additionally, the deep learning-enhanced ANFIS was introduced to further identify intrusion behaviors in today's complex network environments by enabling deeper pattern extraction. To establish a performance baseline, the original algorithm's performance was evaluated to clearly visualize the improvement of the new algorithms. Additionally, their performance will be compared with that of other existing algorithms to validate the effectiveness of the proposed methods.

CHAPTER 4

DEEPER AND INTERPRETABLE ANFIS FOR INTRUSION DETECTION

4.1 Overview

The proposed approach aims to enhance the deep pattern extraction ability of ANFIS by utilizing other interpretable algorithms to deepen its structure. Specifically, this study employs the CART (Classification and Regression Tree) algorithm to achieve this goal, considering its high performance, efficiency, and interpretability.

To enhance the interpretability and depth of intrusion detection based on ANFIS, this study proposes a novel approach that combines ANFIS with CART. By co-training these two models, the study aims to identify deep intrusion patterns and create a more interpretable structure. Furthermore, to improve the efficiency of training and prediction, the study utilizes pre-processing techniques such as Pearson Correlation analysis, standard deviation, and an adaptive K-means algorithm. These techniques aid in attribute selection and fuzzy interval decision-making, ultimately reducing the number of fuzzy rules generated by ANFIS. The incorporation of these techniques not only improves the performance of the proposed algorithm but also enhances its efficiency and interpretability in the context of intrusion detection.

4.2 Pre-processing Techniques

This section describes the preprocessing techniques employed in the proposed method, specifically focusing on attribute selection and interval initialization for continuous attributes. The primary goal of preprocessing is to reduce the dimensionality of the data, which in turn enhances the performance of the ANFIS-based intrusion detection system. By carefully

selecting relevant attributes and initializing appropriate intervals for continuous attributes, the pre-processing step helps to streamline the data and improve the efficiency and effectiveness of the subsequent analysis. Ultimately, these pre-processing techniques contribute to the overall performance enhancement of the intrusion detection system based on ANFIS.

The first technique is attribute selection, which plays a crucial role in mitigating overfitting and improving system performance. Pearson correlation analysis is utilized to measure the correlation between each attribute and the target variable. Attributes with high correlation are selected for further analysis. This initial attribute selection is particularly important for problems with a large number of attributes, as it helps reduce the complexity of the system and improve efficiency, as shown in Equation 2.5.

The second technique involves using the standard deviation to determine the initial number of intervals for continuous attributes selected through Pearson correlation analysis. The standard deviation reflects the degree of variation in an attribute. By adding 2 to the exponent of the standard deviation, the initial number of intervals for each continuous attribute is determined. When we take the standard deviation and add 2 to it as the initial number of intervals, we are actually adjusting a factor that can be used to determine the distribution of the data. Adding 2 is to ensure that the change is amplified, but it will not result in too many attribute intervals, allowing us to capture most of the changes in the data.

The third technique focuses on employing an adaptive K-means algorithm (shown in Figure 4.1) to dynamically determine the minimum number of clusters, thereby further minimizing and optimizing the number of intervals. Unlike traditional K-means, which requires the number of clusters to be known in advance, the adaptive K-means algorithm dynamically determines the optimal number of clusters based on the characteristics of the data.

This adaptive approach improves the accuracy and efficiency of clustering, leading to more effective interval determination.

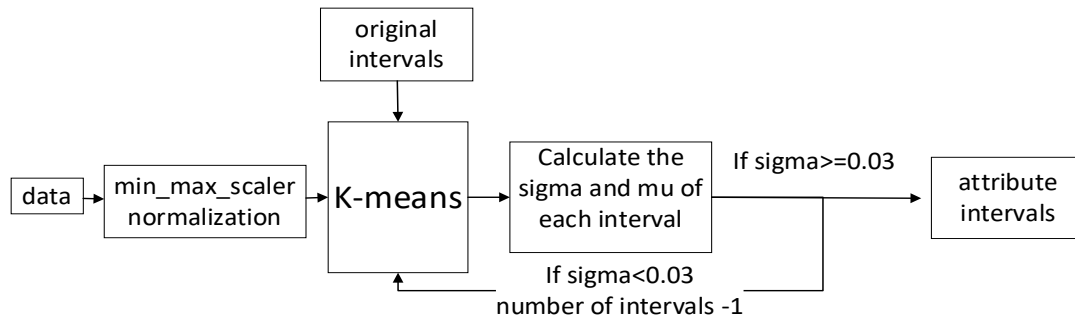


Figure 4.1: Adaptive K-means Algorithm

The adaptive K-means algorithm consists of four stages to determine the appropriate number of clusters and intervals for the data:

Stage 1: This stage involves normalizing all attributes using the min_max_scaler normalization technique. This normalization step eliminates the influence of dimensionality among attributes, ensuring that each attribute is on a similar scale.

Stage 2: The K-means algorithm is applied to calculate the cluster centers for each interval. The initial number of intervals is determined by calculating the exponent of the standard deviation of each attribute and adding 2 to it. This initial number of intervals serves as the starting point for the K-means clustering process.

Stage 3: This stage focuses on dividing the data into intervals and calculating the *sigma* and *mean* values for each interval. The *sigma* value is a measure of the spread of data within an interval. It is calculated as the standard deviation of the data points within that interval. The *sigma* value indicates how much the data varies within the interval. A low *sigma* value indicates that the data points within the interval are close to each other, while a

high *sigma* value indicates that the data points are more spread out. If the *sigma* value of an interval is less than 0.03, indicating insufficient dispersion of data within that interval, the clustering process is repeated by reducing the number of clusters by 1. This iterative step ensures that each interval captures a sufficient amount of data and improves the quality of clustering.

$$mean = \frac{\sum_1^n x_i}{n} \quad \text{Equation 4.1}$$

$$sigma = \sqrt{\frac{\sum_1^n (x_i - mean)^2}{n}} \quad \text{Equation 4.2}$$

Where n is the total amount of data.

Stage 4: This stage is reached when the *sigma* values of all attributes are greater than 0.03, indicating that the data has been properly distributed among the intervals. At this stage, the attributes are assigned to their corresponding intervals, and the calculated *sigma* and *mean* values become the original *sigma* and *mean* values used in the subsequent training of the ANFIS model. This ensures that the sigma and mean values of the ANFIS are at an appropriate level during the initial phase.

By implementing these pre-processing techniques, the proposed method reduces data dimensionality, selects relevant attributes, initializes appropriate intervals for continuous attributes, and optimizes the number of intervals. These steps collectively contribute to improving the performance and interpretability of the intrusion detection system based on ANFIS. The details calculation will be provided in Section 4.4.1- Pre-processing Results.

4.3 Architecture of Model

The proposed method utilizes the CART to deepen the structure of ANFIS and provides interpretability. By utilizing CART, the proposed method can identify deeper, more

nuanced patterns within the data that might be missed by traditional ANFIS. This increased pattern recognition leads to more accurate predictions and detections. Meanwhile, the interpretability of CART complements the opaquer nature of ANFIS, providing a balance between accuracy and interpretability, as illustrated in .

In , the input data $X = \{X_1 \dots X_{total}\}$ is the data with attributes selected by Pearson correlation analysis. *total* is the number of items in X . The X_i represents the i -th item of X . x_{ij} represents the j -th attribute of X_i , $j = 1, 2, \dots, n$. n is the number of attributes.

The selected attribute, x_{ij} , is divided into k overlapping intervals based on the standard deviation and adaptive K-means. The number of intervals, namely k , for the attribute is determined by the standard deviation. The original *mean* and *sigma* for each interval are generated through adaptive K-means, as described in Section 4.2. The *mean* represents the location of the interval and the *sigma* represents dispersion. The resulting k intervals are overlapped, as shown in Figure 4.3. A_{ijk} represents the j -th attribute's k -th fuzzy interval for item i . μ_{ijk} represents the j -th attribute's k -th Gaussian MF at the beginning of training the i -th item.

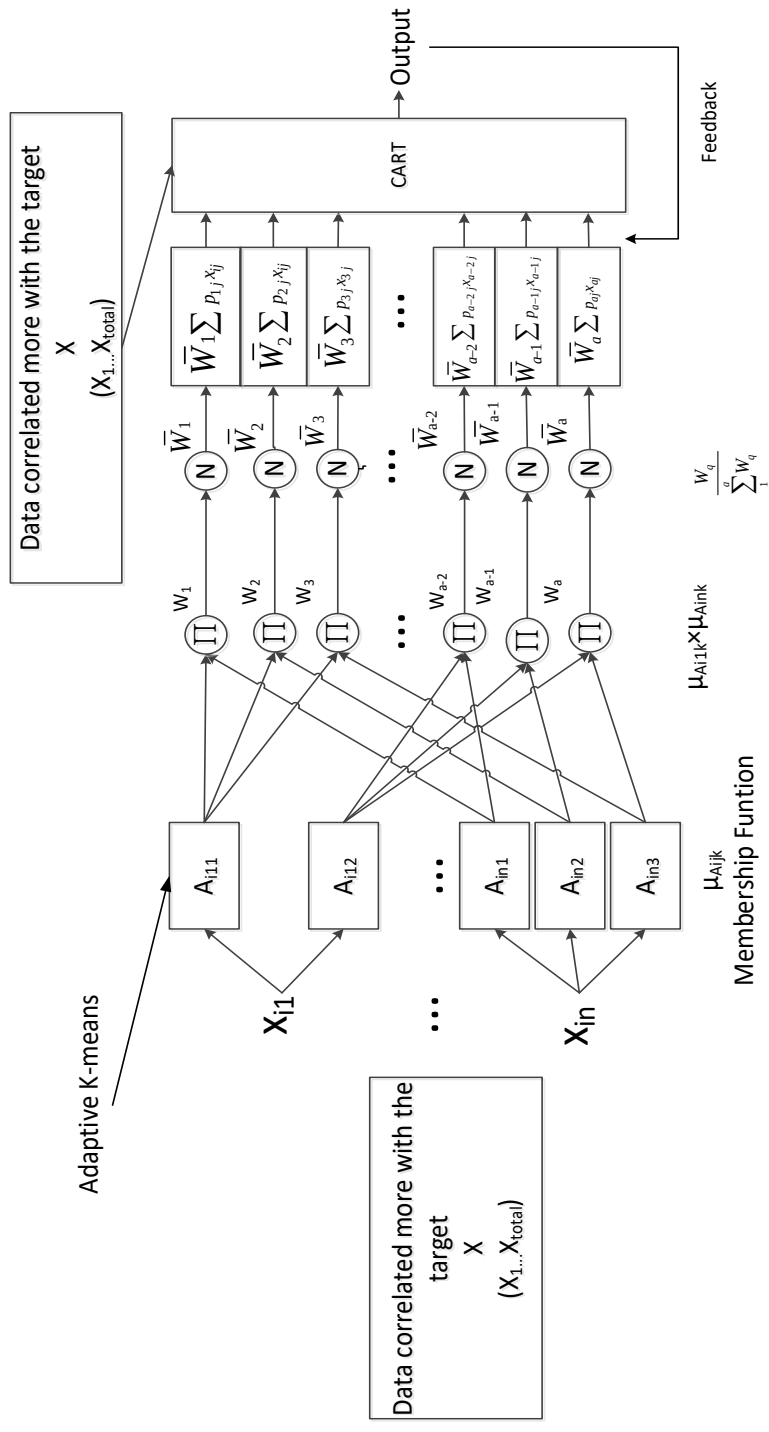


Figure 4.2: Architecture of ANFIS-CART

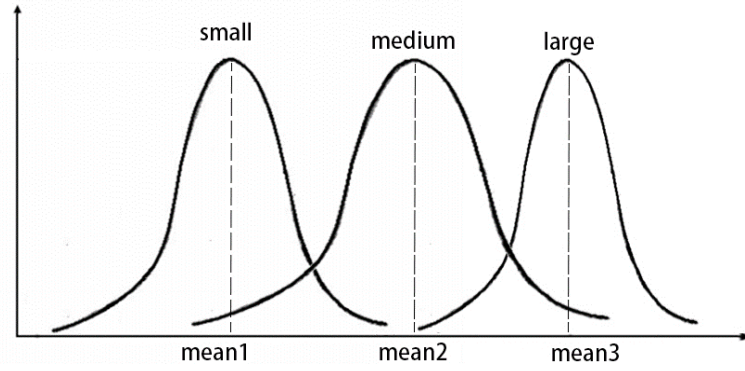


Figure 4.3: Overlapped Fuzzy Intervals

The proposed ANFIS-CART model consists of six layers. The total number of generated rules is a . W_q represents the q -th rule. The model is described below:

Layer 1: Gaussian Membership Function (MF) Formation. In this layer, the *means* and *sigmas* calculated by adaptive K-means are used to form Gaussian MFs. The membership grade of the inputs is then calculated based on these MFs.

$$\mu_{A_{ijk}}(x_{ij}) = e^{-\frac{1}{2}(\frac{x - \text{mean}_{ijk}}{\text{sigma}_{ijk}})^2} \quad \text{Equation 4.3}$$

Where *mean* and *sigma* are the learning parameters.

Layer 2: Firing Strengths Calculation. Each node in this layer calculates the firing strength of the q -th rule using the membership grades.

$$w_q = \mu_{A_{i1k}}(x_{i1}) \times \dots \times \mu_{A_{ink}}(x_{in}) \quad \text{Equation 4.4}$$

Layer 3: Normalized Firing Strengths Calculation. The normalized firing strength of the q -th rule is calculated in this layer.

$$\bar{w}_q = \frac{w_q}{\sum_{q=1}^a w_q} \quad \text{Equation 4.5}$$

Where a is the number of rules.

Layer 4: Weighted Rules Calculation. $WF_q \in WF = \{WF_1, WF_2, \dots, WF_a\}$, $WF_q = \sum_1^n \bar{w}_q f_{ij}$. The weighted rules set, denoted as WF , formed by combining the weighted rules. Each WF_q is calculated as the sum of the product of the normalized firing strength.

$$\bar{w}_q f_{ij} = \bar{w}_q p_{qj} x_{ij} \quad \text{Equation 4.6}$$

Where p_{qj} is the learning parameter.

Layer 5: Input Preparation for CART. In this layer, the 1 to total item's weighted rules, denoted as all_WF , are concatenated with the selected attributes (X) obtained from Pearson correlation analysis. The concatenated input, denoted as X' , is prepared for training the CART.

$$X' = \text{Concatenate}(all_WF, X) \quad \text{Equation 4.7}$$

Layer 6: X' as the input to train the CART. Then, the result of X_i , namely Y' , is calculated using the after-trained CART. Then, calculate the loss between Y' and the actual Y . The loss is calculated by MSEloss, The MSEloss is calculated by Equation 4.8.

$$\text{loss}(Y', Y) = (Y' - Y)^2 \quad \text{Equation 4.8}$$

In the proposed ANFIS-CART model, the loss calculated in Layer 6 is passed back to Layer 4. The learning mechanism is demonstrated in Section 2.5.2. In this proposed method, the Gradient Descent Only is used to update all the parameters. The optimizer of this step is SGD (Stochastic Gradient Descent) optimizer, the learning rate is 0.0001, the momentum is 0.99.

The SGD optimizer is a popular optimization algorithm used in deep learning to update the weights of neural networks during training. In SGD, the parameters of the network are updated using the gradient of the loss function with respect to the weights.

The learning rate determines the step size of the update, and it is usually set to a small value to ensure that the algorithm converges to a minimum of the loss function. However, this can also result in slow convergence or getting stuck in a local minimum. To address this, variants of SGD have been proposed, such as Momentum, Adagrad, RMSprop, and Adam, which use different update rules or adaptive learning rates to improve convergence and stability.

Momentum is a technique used in optimization algorithms, particularly in SGD, to accelerate the learning process and avoid getting stuck in local minima. In the context of optimization, momentum refers to the accumulated previous gradients of the parameters. During each iteration of the optimization algorithm, the momentum term is added to the current gradient. This has the effect of increasing the step size in the direction of the previous gradient, allowing the optimization algorithm to move more quickly in that direction. Momentum is controlled by a hyperparameter called the momentum coefficient, which determines the weighting given to the previous gradients relative to the current gradient. A momentum coefficient value close to 1 gives more weight to the previous gradients, while a value close to 0 gives equal weight to all gradients. By using momentum, the optimization algorithm can navigate the loss function more efficiently and reach the minimum more quickly than standard SGD.

The study employs the gradient descent method to update the learning parameters (premise parameters and consequent parameters) in the proposed ANFIS-CART model. This

choice is made because the computation complexity of the gradient descent is lower compared to the least squares estimate. Therefore, Algorithm 1, which outlines the specific steps of the learning process, utilizes the gradient descent method for updating the parameters. On the other hand, the CART part of the model employs the best strategy to split each node, ensuring efficient and effective decision tree construction.

Algorithm 1 The ANFIS-CART algorithm

Input: $X \in \mathbb{R}^{total \times n}$, X is the selected attributes by Pearson correlation analysis.

Output: Classification result $Y \in \mathbb{R}^{total}$.

Step 1. $ANFIS(X) \rightarrow (WF)$:

- The original Gaussian membership functions (formed by *means* and *sigmas* after Adaptive K-means).

$$\mu(x) = e^{-\frac{1}{2}(\frac{x-mean}{sigma})^2} \quad \text{Equation 4.9}$$

Where *means* and *sigmas* are the learning parameters.

- w_q , firing strengths of q -th rule, as follows:

$$w_q = \mu_{A_{i1k}}(x_{i1}) \times \dots \times \mu_{A_{ink}}(x_{in}) \quad \text{Equation 4.10}$$

Where $\mu_{A_{ijk}}$ is the k -th MF of x_{ij} . x_{ij} is the j -th attribute of X_i . X_i is the i -th item of X .

n is the number of total attributes. $i = 1, 2, \dots, total$, $j = 1, 2, \dots, n$.

- $\bar{w}_q$, firing strengths of q -th rule, as follows:
-

$$\bar{w}_q = \frac{w_q}{\sum_{q=1}^a w_q} \quad \text{Equation 4.11}$$

Where a is the number of rules.

- $WF = \{WF_1, WF_2, \dots, WF_a\}$, $WF_q = \sum_1^n \bar{w}_q f_{ij}$, weighted rules set, each weighted rule can be calculated as follows:

$$\bar{w}_q f_{ij} = \bar{w}_q p_{qj} x_{ij} \quad \text{Equation 4.12}$$

Where $x_{ij} \in X_i$, p_{qj} is the learning parameter.

Step 2. *CART*(all_WF, X) $\rightarrow$ (Y'):

- Calculate all rules of X through the forward of Step 1, namely all_WF .
- Concatenate all_WF, X as:

$$X' = (all_WF, X) \quad \text{Equation 4.13}$$

- Calculate ***CART***(X') to generate the CART tree. Then according to the CART tree to get the output of X_i , namely Y' .

Step 3. *Backpropagation*(Y', Y):

- Calculate the MSE loss of (Y', Y). MSE stands for Mean Squared Error. It is the average difference between the predicted and actual values in machine learning.
 - Then, calculate the gradient with the function in Layer 4. Use SGD (learning rate=1e-4, momentum=0.99) to update the premise parameters and consequent parameters. SCD stands for Stochastic Gradient Descent. Unlike traditional gradient
-

descent, which uses the entire training set for parameter updates in each iteration, SCD only uses one sample per iteration. This makes SCD more efficient for large datasets.

Step 4. *Training and Verify_Loss*(Y', Y):

- Keep training the algorithm from Step 1 to Step 3.
- Calculate the loss of (Y', Y) on each epoch.
- Early Stopping mechanism (patience=5). The early stopping mechanism in this study was implemented with a patience value of 5 epochs. The selection of this value was based on careful consideration of the trade-off between preventing overfitting and allowing sufficient training time. If the patience value is set too low, there is a risk of premature stopping of the training process, leading to suboptimal performance. Conversely, setting the patience value too high may result in overfitting of the training data and poor generalization performance on the test data. In this context, a patience value of 5 epochs was considered to strike a reasonable balance between these competing considerations. Specifically, the training process is stopped if the validation loss fails to improve for 5 consecutive epochs. This is based on the assumption that, after 5 epochs, the model should be making reasonable progress in terms of reducing the validation loss. If there is no improvement in the validation loss after 5 epochs, it is likely that the model has reached a suboptimal solution or is overfitting the training data.

By Algorithm 1, the membership function is utilized to transform the attribute values into membership degrees belonging to specific intervals. This enables the representation of

attribute values as fuzzy and overlapped intervals, providing the proposed algorithm with fuzzy, overlapped, and generalization capabilities. Subsequently, fuzzy rules are generated based on the product of different interval membership degrees of attributes, representing various combinations of attribute values (e.g., "duration is small," "count is medium," "diff_srv_rate is big").

After normalizing the firing strengths of the rules w_q (rules' firing strengths), weighted fuzzy rules (all_WF) are formed, indicating the contributions of each rule to the target. These weighted fuzzy rules, along with the original input data (X), are concatenated and inputted into the CART component for interpreting deep patterns..

The performance of the proposed method will be evaluated using the validation dataset (KDDTrain+ subset) and the test dataset (KDDTest+ subset). The trained model will be applied to the validation dataset and test dataset to obtain the predicted results ($valid_Y'$ and $test_Y'$). The evaluation indicators described in Section 3.5 will be used to assess the accuracy and performance of the model by comparing the predicted results with the actual target values ($valid_Y$ and $test_Y$). This evaluation process provides insights into the effectiveness of the proposed method in intrusion detection.

4.4 Experimental Results

In this study, the performance of the proposed method was evaluated on the NSL-KDD dataset using six evaluation metrics: precision, recall rate, F1-Score, accuracy, detection rate, and false alarm rate, as described in Section 3.5. The proposed method began by applying attribute selection and interval initialization techniques, specifically the adaptive K-means and standard deviation approaches. These techniques were used to select relevant attributes and initialize overlapping intervals for continuous attributes.

Subsequently, the proposed method was trained on 80% of the KDDTrain+ dataset. The remaining KDDTrain+ subset was used as the validation set, while the KDDTest+ subset served as the test set. The performance of the proposed method was then assessed on these datasets to evaluate its effectiveness in intrusion detection. The evaluation metrics provided insights into the accuracy, detection rate, and false alarm rate of the proposed method, demonstrating its performance and effectiveness in handling intrusion detection tasks.

4.4.1 Pre-processing Results

Table 4.1 displays the Pearson correlation coefficients between each of the 41 normal attributes and the target attribute in the NSL-KDD dataset. The Pearson correlation coefficient measures the linear correlation between two variables and ranges from -1 to 1, where -1 represents a perfect negative correlation, 0 represents no correlation, and 1 represents a perfect positive correlation. The absolute values of the coefficients in Table 4.1 are used to determine the correlation strength of each attribute towards the target attribute.

Based on the threshold of a correlation coefficient greater than 0.1, Table 4.2 displays the 18 continuous attributes and 4 discrete attributes that were selected for the proposed method. In Table 4.2, these four attributes (protocol_type, service, flag, and logged_in) are discrete attributes that have been selected for further analysis in the proposed method. They are believed to have a higher degree of linear correlation with the target attribute and will be used as input for the intrusion detection system. Figure 3.4 shows the total Pearson correlation matrix of the NSL-KDD dataset's 43 attributes. This matrix provides an overview of the correlation between each attribute in the dataset, where the coefficients' colors represent the strength of the correlation.

Table 4.1: Correlation Degree of 41 Attributes towards Target

Attribute	Correlation Degree
dst_host_srv_count	-0.62425
logged_in	-0.57604
hard	-0.557252
dst_host_same_srv_rate	-0.525837
same_srv_rate	-0.510869
flag	-0.501333
srv_count	-0.038856
num_access_files	-0.030234
su_attempted	-0.019483
num_file_creations	-0.014385
num_root	-0.01002
num_compromised	-0.009003
is_host_login	-0.002334
land	0.002944
urgent	0.003941
num_shells	0.005208
root_shell	0.007757
dst_bytes	0.008651
srv_diff_host_rate	0.010454
src_bytes	0.012981
num_failed_logins	0.02312
is_guest_login	0.0306
wrong_fragment	0.054795

Table 4.1 continued

hot	0.065085
duration	0.129443
protocol_type	0.168167
dst_host_count	0.196101
service	0.233783
dst_host_srv_diff_host_rate	0.239645
diff_srv_rate	0.282313
dst_host_error_rate	0.293804
error_rate	0.308667
srv_error_rate	0.309907
dst_host_srv_error_rate	0.310226
dst_host_same_src_port_rate	0.319097
srv_error_rate	0.378666
error_rate	0.381644
dst_host_error_rate	0.382115
dst_host_srv_error_rate	0.384793
count	0.388355
dst_host_diff_srv_rate	0.391327

Table 4.2: The Standard Deviation and Original Interval Number of Attributes

Attribute Name	Standard Deviation	Original Interval Number
protocol_type	5.65E-01	1
service	1.69E+01	1
flag	2.69E+00	1
logged_in	4.89E-01	1
dst_host_srv_diff_host_rate	1.13E-01	1
diff_srv_rate	1.80E-01	1
dst_host_diff_srv_rate	1.89E-01	1
dst_host_rerror_rate	3.07E-01	1
dst_host_same_src_port_rate	3.09E-01	1
dst_host_srv_rerror_rate	3.19E-01	1
rerror_rate	3.20E-01	1
srv_rerror_rate	3.24E-01	1
same_srv_rate	4.40E-01	1
dst_host_serror_rate	4.45E-01	1
dst_host_srv_serror_rate	4.46E-01	1
serror_rate	4.46E-01	1
srv_serror_rate	4.47E-01	1
dst_host_same_srv_rate	4.49E-01	1
dst_host_count	9.92E+01	3
dst_host_srv_count	1.11E+02	4
count	1.15E+02	4
duration	2.60E+03	5

Table 4.2 shows the original number of intervals for the selected continuous attributes (duration, src_bytes, dst_bytes, hot). These attributes initially had 6 intervals each.

To reduce the number of intervals and initialize the *sigma* and *mean* values, standard deviation analysis was performed. The final number of intervals, *sigmas*, and *means* for each attribute are presented in Table 4.3.

Appendix B lists the 192 fuzzy rules generated by the ANFIS-part for the selected continuous attributes (duration, src_bytes, dst_bytes, hot) with interval numbers greater than 1. Each rule specifies the interval numbers for each attribute, allowing for fuzzy and overlapping combinations.

Appendix C provides the weight p_{qj} values for the selected continuous attributes, , as shown in Equation 4.12. These values represent the weights of the attributes in each rule and play a role in determining the contribution of each attribute to the overall decision-making process.

Table 4.3: Final Interval Number of Selected Continuous Attributes

Attribute Name	Final Interval Number	Original mean and Sigma (First is mean, second is sigma)
diff_srv_rate	1	[0.6, 1.8]
dst_host_diff_srv_rate	1	[0.8, 1.9]
dst_host_rerror_rate	1	[1.2, 3.1]
dst_host_same_src_port_rate	1	[1.5, 3.1]
dst_host_same_srv_rate	1	[5.2, 4.5]
dst_host_serror_rate	1	[2.8, 4.4]

Table 4.3 continued

dst_host_srv_diff_host_rate	1	[0.3, 1.1]
dst_host_srv_error_rate	1	[1.2, 3.2]
dst_host_srv_serror_rate	1	[2.8, 4.5]
rerror_rate	1	[1.2, 3.2]
same_srv_rate	1	[6.6, 4.4]
serror_rate	1	[2.8, 4.5]
srv_error_rate	1	[1.2, 3.2]
srv_serror_rate	1	[2.8, 4.5]
dst_host_count	3	[1.0, 0.9], [5.1, 1.3], [9.9, 0.4]
dst_host_srv_count	4	[0.4, 0.4], [2.9, 0.8], [6.2, 1.0], [9.9, 0.4]
count	4	[0.2, 0.2], [2.4, 0.6], [4.8, 0.7], [9.5, 0.8]
duration	4	[0.0, 0.1], [2.1, 0.7], [4.9, 0.9], [8.7, 0.9]

4.4.2 Results of ANFIS-CART

After selecting relevant attributes, the proposed method retained 18 continuous attributes and 4 discrete attributes, effectively reducing the data dimension. The standard deviation analysis and adaptive K-means techniques were then utilized to initialize the number, *means*, and *sigmas* of intervals for each attribute, resulting in the reduction of the number of fuzzy rules generated by ANFIS to 192. This reduction had a positive impact on the training and prediction efficiency of the algorithm. This is because fewer fuzzy rules

mean less computational complexity and faster convergence during training, which in turn improves the speed of the algorithm's predictions.

The algorithm was trained using Algorithm 1, and when tested on the entire KDDTrain+ dataset, it achieved a 99.86% detection rate (DR) and a 0.14% false alarm rate (FAR), as demonstrated in Table 4.4. The proposed method was further evaluated on the full KDDTrain+ subset and the KDDTest+ subset, with the results presented in Table 4.4 and Table 4.5, respectively. These tables provide performance metrics such as precision, recall rate, F1-Score, and accuracy, demonstrating the effectiveness of the proposed method in intrusion detection. The performance comparison with other algorithms is described in Section 4.6.

Table 4.4: Performance of Proposed Method on KDDTrain+ Dataset

Category	Precision (%)	Recall (%)	F1-score(%)	DR(%)	FAR(%)	ACC(%)
Normal	99.84	99.90	99.87	99.90	0.16	99.87
DOS	99.94	99.90	99.92	99.90	0.06	99.91
Probing	99.75	99.83	99.79	99.83	0.24	99.79
R2L	98.67	96.88	97.77	96.88	1.31	97.79
U2R	90.00	69.23	78.26	69.23	7.69	80.77
weighted avg	99.86	99.86	99.86	99.86	0.14	99.86

Table 4.5: Performance of Proposed Method on KDDTest+ Dataset

Category	Precision (%)	Recall (%)	F1-score(%)	DR(%)	FAR(%)	ACC(%)
Normal	68.1	96.88	79.98	96.88	45.37	75.75
DOS	95.6	77.54	85.63	77.54	3.57	86.98
Probing	86.51	78.4	82.25	78.40	12.23	83.08
R2L	83.33	13.25	22.87	13.25	2.65	55.30
U2R	16.33	4.00	6.43	2.05	6.42	41.75
weighted avg	80.58	77.46	74.46	77.46	22.54	77.46

4.5 Interpret The Intrusion Patterns

To interpret the intrusion pattern, one can first analyze the CART part of the model to identify the attributes and rules that contribute to the classification, and then further analyze the rules using the ANFIS part of the model.

In CART, each node in the decision tree represents a test on a particular attribute, and the branch coming out of the node represents the possible outcomes of the test. The leaf nodes represent the final classification of the instance based on the tests performed on its attributes.

The importance of an attribute can be interpreted based on the level of the tree it appears on and the number of instances that it separates. Attributes that appear near the root of the tree and separate a large number of instances are considered to be more important in the classification process.

In addition, CART can also provide measures of variable importance based on the amount of improvement in classification performance that results from each attribute. The Gini importance, for example, measures the total reduction in the impurity measure (e.g.,

Gini index) achieved by splitting on an attribute over all splits in which it appears. Attributes with higher Gini importance are considered more important for the classification.

The CART interprets the rules by ANFIS and other attributes by determining their importance and contribution to the target. By interpreting the CART, one can understand the decision-making process of the proposed method and how it arrives at its classification results. This adds to the interpretability of the proposed method and allows for better understanding and analysis of its performance.

An example can be used to illustrate how the proposed method can be interpreted. Figure 4.4 displays a part of the fully generated CART, where the intrusion types identified by the proposed model are denoted by the red box. In the red box, the intrusions identified by the proposed model are DOS and Probing. By examining the complete CART tree, one can gain a better understanding of the decision-making process of the proposed method.

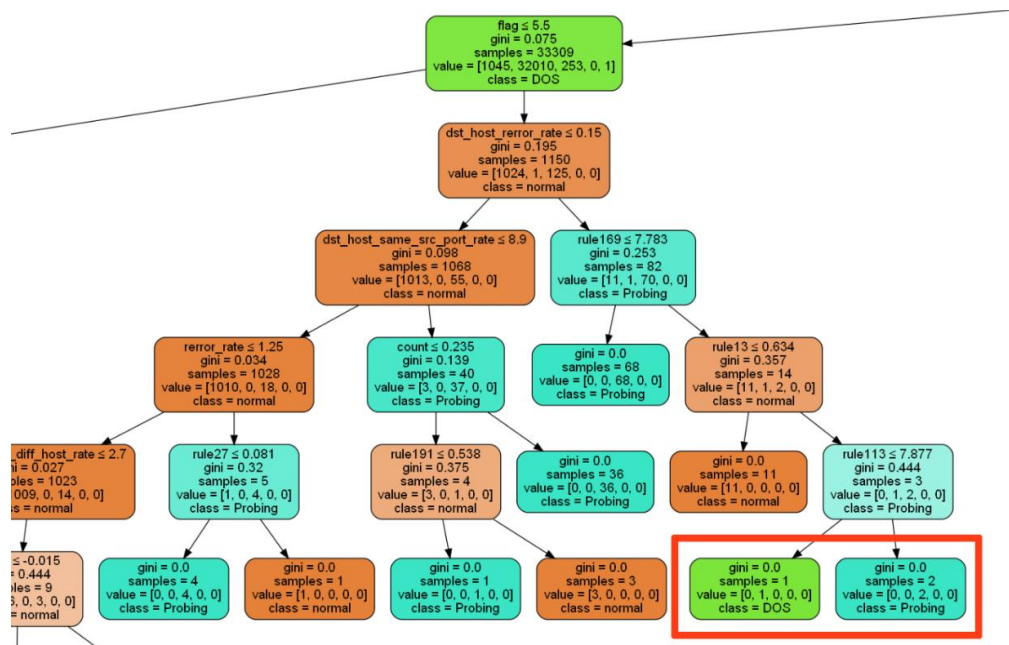


Figure 4.4: The Node for Interpreting the Pattern

For example, in Figure 4.4, if certain conditions such as $\text{same_srv_rate} \leq 4.95$, $\text{dst_host_diff_srv_rate} \leq 1.55$, $\text{flag} > 5.5$, and $\text{dst_host_error_rate} > 0.15$, $\text{rule169} > 7.783$, $\text{rule13} > 0.634$, along with specific rule values, are met, the behaviour is identified as an intrusion. To further identify the intrusion type, rule113 is utilized. If $\text{rule113} \leq 7.877$, the intrusion type is identified as DOS, otherwise it is classified as Probing.

After obtaining the relevant attributes and rules for the discriminant classification results, the rules are further analyzed through the ANFIS-part.

ANFIS has the advantage of interpretability compared to deep artificial neural networks (ANNs) due to its architecture of only having five layers. The generated rules and their contribution to the target are crucial aspects of ANFIS. The contribution of each rule is determined by the firing strengths $\bar{w}_q$ which are calculated by the combination of attributes' fuzzy intervals, as shown in Equation 4.14. A higher firing strength indicates a higher contribution of the rule to the target. The importance of the attributes to the rules is represented by p_{qj} . A higher p_{qj} value signifies higher importance of the attribute to the rule.

$$\bar{w}_q f_{ij} = \bar{w}_q p_{qj} x_{ij} \quad \text{Equation 4.14}$$

The Appendix B and Appendix C list the 192 rules with the four continuous attributes (interval number > 1). The values listed in c are the interval numbers for each attribute, such as 1 is small, 2 is medium, 3 is large. The values listed in Appendix C are the p_{qj} of four continuous attributes. In Appendix C, a part of the components of the 192 rules generated and trained by the proposed method is presented, which represent the various combinations of the selected attributes that correspond to different situations, such as high duration with low count and low dst_host_count , etc. In each situation, i.e., under each rule, the

contribution of various attributes to the rule varies, which is reflected in the value of p_{qj} . Then, the rules of ANFIS, which are the sum of weighted attributes (weighted by the p_{qj}), are weighted using the firing strength ($\bar{w}_q$), and input into the CART to capture additional patterns.

In Table 4.6, the rules used in the first step to identify intrusions, namely rule169, rule13, and rule113, are presented. These rules are obtained from the proposed model, which combines ANFIS and CART for intrusion detection.

Table 4.6: The Classification Pattern of The Nodes

Path	Dividing Value
same_srv_rate	≤ 4.95
dst_host_diff_srv_rate	≤ 1.55
flag	> 5.5
dst_host_rerror_rate	> 0.15
rule169 (duration is 3, count is 2, dst_host_count is 0, dst_host_srv_count is 1)	> 7.783
rule13 (duration is 0, count is 1, dst_host_count is 0, dst_host_srv_count is 1)	> 0.634
rule113 (duration is 2, count is 1, dst_host_count is 1, dst_host_srv_count is 1)	> 7.877 (Probing)
rule113 (duration is 2, count is 1, dst_host_count is 1, dst_host_srv_count is 1)	≤ 7.877 (DOS)
0-3 represent which fuzzy interval the attribute will be calculated in	

In the analysis of the used rules, the firing strength $\bar{w}_q$ plays a crucial role in determining the importance of different rules in the classification process. The value of $\bar{w}_q$ is influenced by the composition of attributes in the rules and the fuzzy degree of these attributes, which are obtained through the ANFIS-part of the proposed model.

Additionally, the p_{qj} of each rule, as shown in Equation 4.14 and Appendix C, determines the contribution of each attribute to the rule. The p_{qj} values represent the importance of the attributes in the rule, and a higher p_{qj} value indicates a higher contribution of that attribute to the rule.

In this way, the attributes and rules used in intrusion detection were first analyzed using the CART part, and then the composition of the rules and their contributions to the results was further analyzed using the ANFIS part. The specific patterns of the intrusions identified by the proposed method, as indicated by the red box in Figure 4.4 were analyzed and presented in Table 4.6.

By leveraging the combination and co-training mechanism of ANFIS and CART, the proposed method achieves a more advanced pattern extraction structure compared to using a single ANFIS or CART. This integration allows for the identification of deeper patterns in intrusion detection.

Furthermore, the proposed method offers the crucial advantage of greater interpretability compared to traditional artificial neural networks (ANNs). The interpretability of the algorithm allows for easier understanding and analysis of the complex patterns associated with intrusions. This interpretability is particularly beneficial in engineering applications, where practical intrusion detection systems (IDSs) are developed.

The ability to interpret the underlying patterns and decision-making process enhances the effectiveness and applicability of the proposed method in real-world scenarios, aiding in the protection against intrusions and the construction of robust engineering IDSs.

4.6 Discussion

The results on the KDDTrain+ dataset demonstrate that the proposed method achieves a high detection rate (DR) of 99.86% and a low false alarm rate (FAR) of 0.14%. These results indicate the effectiveness of the proposed method in accurately identifying intrusions while minimizing false alarms, as shown in Table 4.7.

Furthermore, the proposed method exhibits good generalization ability on the KDDTest+ dataset. The overall DR on KDDTest+ is 77.46%, with a precision of 80.58% and an accuracy of 77.46%. However, the FAR on KDDTest+ is relatively high at 22.54%, indicating room for improvement in reducing false alarms.

Comparisons with other studies, such as BPNN, GA-ANFIS, PSO-ANFIS, and CSO-ANFIS (Manimurugan et al., 2020), were conducted to evaluate the performance of the proposed method, as designed in Section 3.5 and Section 3.6. Table 4.7 and Table 4.8 present the classification performance of the five main categories, while Table 4.9 compares the performance with single CART and ANFIS. Additionally, Table 4.10 provides a comparison of accuracy with other existing approaches.

These comparative results highlight the competitive performance of the proposed method and its potential as an effective intrusion detection solution.

Table 4.7: Comparison of Detection Rate and FAR on KDDTrain+ Dataset

Classifier	DR	FAR
BPNN (Manimurugan et al., 2020)	85.05	6.10
FC-ANN (Manimurugan et al., 2020)	89.23	4.37
GA-ANFIS (Manimurugan et al., 2020)	93.15	3.75
PSO-ANFIS (Manimurugan et al., 2020)	94.46	3.38
CSO-ANFIS (Manimurugan et al., 2020)	95.80	3.45
CART	99.74	0.06
Proposed Model	99.86	0.14

Table 4.8: Comparison of Accuracy on KDDTrain+ Dataset

Classifier	Normal	DOS	Probing	R2L	U2R
BPNN (Manimurugan et al., 2020)	91.50	90.94	80.53	48.13	60.00
FC-ANN (Manimurugan et al., 2020)	93.87	80.32	89.12	89.57	75.58
GA-ANFIS (Manimurugan et al., 2020)	96.22	96.70	93.18	94.89	83.33
PSO-ANFIS (Manimurugan et al., 2020)	96.46	95.90	91.35	94.72	85.62
CSO-ANFIS (Manimurugan et al., 2020)	97.41	96.25	92.51	94.15	90.26
Proposed Model	99.87	99.92	99.79	97.79	80.77

Table 4.9: Comparison of CART, ANFIS, and Proposed Method on KDDTest+ Dataset

Category	Precision (%)	Recall (%)	F1-score(%)	DR(%)	FAR(%)	ACC(%)
CART	81.06	76.23	72.52	76.22	23.77	76.23
ANFIS	60.42	65.93	60.80	65.93	34.07	65.93
Proposed Model	80.58	77.46	74.46	77.46	22.54	77.46

Table 4.10: Comparison of Accuracy with Other Existing Approaches on KDDTrain+ Dataset

Category	ACC(%)
SVM (Toosi & Kahani, 2007)	88.33
DBN (Alom et al., 2015)	89.54
DBN-SVM (Toosi & Kahani, 2007)	92.84
ANN (MLP) (Benmessahel et al., 2019b)	97.04
Logistic Regression (Benmessahel et al., 2019b)	92.75
Naïve Bayes (Benmessahel et al., 2019b)	95
MVO-ANN (Benmessahel et al., 2019b)	98.21
Proposed Model	99.87

Based on the comparison with CSO-ANFIS (Manimurugan et al., 2020), the proposed method demonstrates better performance with a higher detection rate and a lower false alarm rate. Specifically, the proposed method achieves a detection rate that is 4.06% higher and a false alarm rate that is 2.31% lower than CSO-ANFIS. Furthermore, the proposed method outperforms CSO-ANFIS in terms of accuracy in all main categories except for U2R.

When comparing the proposed method with single CART and ANFIS on the KDDTest+ dataset, which features different intrusion distributions and some intrusions that are only present in this test dataset, it is generally difficult to improve performance. However, the proposed method outperforms the single CART and ANFIS in most metrics except for precision. This indicates that the proposed method enhances the generalization ability of intrusion detection compared to single CART and ANFIS.

The study also compared the proposed method with the work of Chen et al. (2021) and Chen et al. (2022) in the medical field, which also combines decision trees (DT) and ANFIS. However, there are differences in the approaches. Chen et al. used DT to generate crisp rule bases and then converted them into fuzzy rule bases, which were further optimized using ANFIS. In contrast, the proposed method connects ANFIS to the front of CART and co-trains the system. The fuzzy rules generated by modified ANFIS are used as input for CART, enhancing the depth of CART and enabling deeper pattern extraction. The proposed method also exhibits interpretability, similar to the works of Chen et al., allowing for the interpretation of fuzzy characteristics of intrusion behaviours.

Overall, the proposed method demonstrates improved performance compared to existing approaches, providing higher detection rates, lower false alarm rates, and enhanced accuracy in intrusion detection.

4.7 Chapter Summary

In summary, this chapter introduces a novel method that combines ANFIS and CART to create a deep and interpretable model for intrusion detection. The proposed approach leverages the strengths of both ANFIS and CART, resulting in a model that can capture and analyze deeper and fuzzier intrusion patterns. The selection of input attributes is based on Pearson correlation analysis, which ensures that only highly relevant attributes are considered. Additionally, the method employs standard deviation and adaptive K-means to determine the optimal interval divisions for continuous attributes, enhancing the accuracy and interpretability of the ANFIS part.

The performance of the proposed method is evaluated and compared with several existing algorithms, including BPNN, FC-ANN, GA-ANFIS, PSO-ANFIS, and CSO-

ANFIS. On the KDDTrain+ dataset, the proposed method achieves a high detection rate of 99.86% and a low false alarm rate of 0.14%, outperforming the other algorithms. Furthermore, the proposed method is compared with single CART and ANFIS on the KDDTest+ dataset, which contains intrusions not present in the training dataset. The results show that the proposed method outperforms single CART and ANFIS in terms of various performance metrics.

Overall, the proposed method offers a promising approach to intrusion detection, combining the interpretability of CART with the pattern-capturing capabilities of ANFIS. The experimental results demonstrate the effectiveness and superiority of the proposed method, as discussed in Section 4.6, highlighting its potential for practical applications in real-world intrusion detection systems.

CHAPTER 5

HYBRID DEEP FUZZY NEURAL NETWORK

5.1 Overview

Although the combination of ANFIS and CART in intrusion detection can achieve a deeper level of learning compared to ANFIS's five-layer structure, the depth of the model is limited by the constraints of ANFIS and CART, making it difficult to learn more complex patterns.

Learning highly deeper patterns can help us better understand and describe the complexity of data. In machine learning, deep neural networks have been shown to outperform traditional shallow models in many tasks, such as image and speech recognition. By learning deeper patterns through deep learning, models are better able to handle complex nonlinear relationships and improve their ability to represent data. In intrusion detection tasks, learning deeper patterns can help us better understand and describe the complexity of intrusion behaviors and improve the performance of intrusion detection models.

In this study, ResNet is employed to improve the performance and depth of ANFIS. ResNet, or Residual Network, is an influential neural network architecture that has demonstrated exceptional performance on the ImageNet dataset (He et al., 2016b). Its main contribution is to solve the problem of degradation in deep neural networks (i.e., as the number of layers increases, the performance of the network begins to saturate or even decline). When using ResNet for training, it is very easy to add layers, allowing the model to be very deep. ResNet can enable ANFIS to have extremely deep layers, and then use the

deep features of ResNet to identify deep patterns to identify complex intrusion behaviors in the current complex network environment.

The proposed architecture takes advantage of both fuzzy rules generated by ANFIS and concrete attributes using a ResNet to capture and identify deeper patterns in the training data. By combining these two components and incorporating a connection mechanism, the model is co-trained with losses to optimize its performance. This approach not only preserves the uncertainty property of continuous attributes but also enhances the model's ability to capture and understand the complexity of intrusion behavior.

5.2 Architecture of Model

In this approach, ANFIS is utilized to generate fuzzy rules that represent overlapped intervals of continuous attributes and their fuzzy combinations. These rules also incorporate discrete attributes. The modified ANFIS algorithm is employed to generate these fuzzy rules, as depicted in .

In , the input data X divided two parts, $\{X_1, X_2\}$. The first one is the selected continuous attributes X_1 . The attributes in X_1 are the continuous attributes selected by Pearson correlation analysis, which are more linear related to the target. The x_{1i} is the i -th attribute of X_1 , $i = 1, 2, \dots, n$. The second one is the rest continuous attributes and discrete attributes, namely X_2 . The x_{2j} is the j -th attribute of X_2 , $j = 1, 2, \dots, m$.

The proposed method consists of the following layers:

Layer 1 to Layer 3: Same as the proposed ANFIS-CART model described in Section 4.3.

Layer 4: Weighted Rules Calculation. $WF = \{WF_1, WF_2, \dots, WF_a\}$, $WF_q = \sum_{i=1}^n \bar{w}_q f_{qi}$, $\bar{W} = \{\bar{w}_1, \bar{w}_2, \dots, \bar{w}_a\}$, $P_{1q} \in P_1 = \{P_{11}, P_{12}, \dots, P_{1a}\}$, $P_{1q} = \{p_{q11}, p_{q12}, \dots, p_{q1n}\}$. Weighted rules set.

$$\bar{w}_q f_{qi} = \bar{w}_q p_{q1i} x_{1i} \quad \text{Equation 5.1}$$

$$WF = \bar{W} P_1 X_1 \quad \text{Equation 5.2}$$

Where p_{q1i} is the weight of i -th attribute of X_1 for q -th rule. p_{q1i} is the learning parameter. WF_q is the q -th weight set of WF . x_{1i} is the i -th attribute of X_1 . X_1 represents the set of continuous attributes selected from the input data X using Pearson correlation analysis.

Layer 5: Input Preparation for ResNet18, as shown in Figure 2.6. Concatenate the weighted rules WF and weighted X_2 as inputs of ResNet18. X_2 is the combination of rest continuous attributes (apart from the attributes in X_1) and discrete attributes.

$$X' = WF + P_2 X_2 + R \quad \text{Equation 5.3}$$

Where $P_{2q} \in P_2 = \{P_{21}, P_{22}, \dots, P_{2a}\}$, $P_{2q} = \{p_{q21}, p_{q22}, \dots, p_{q2m}\}$, p_{q2j} is the weight of the j -th attribute of X_2 for q -th rule. $R = \{r_q\}$, is the node bias set of $\{r_1, r_2, \dots, r_a\}$.

Layer 6: Feedback of ANFIS-ResNet. Calculate the predicted output Y' using the ResNet18 model. The structure of the ResNet18 model is depicted in Figure 2.6 (He et al., 2016). The algorithm then calculates the gradients and performs gradient-based optimization to co-train the learning parameters. This includes updating the normal learning parameters of ResNet and the *mean*, *sigma* and p_{qi} parameters of ANFIS, as shown in Equation 4.3 and Equation 5.1. The detailed algorithm for co-training is presented in Algorithm 2.

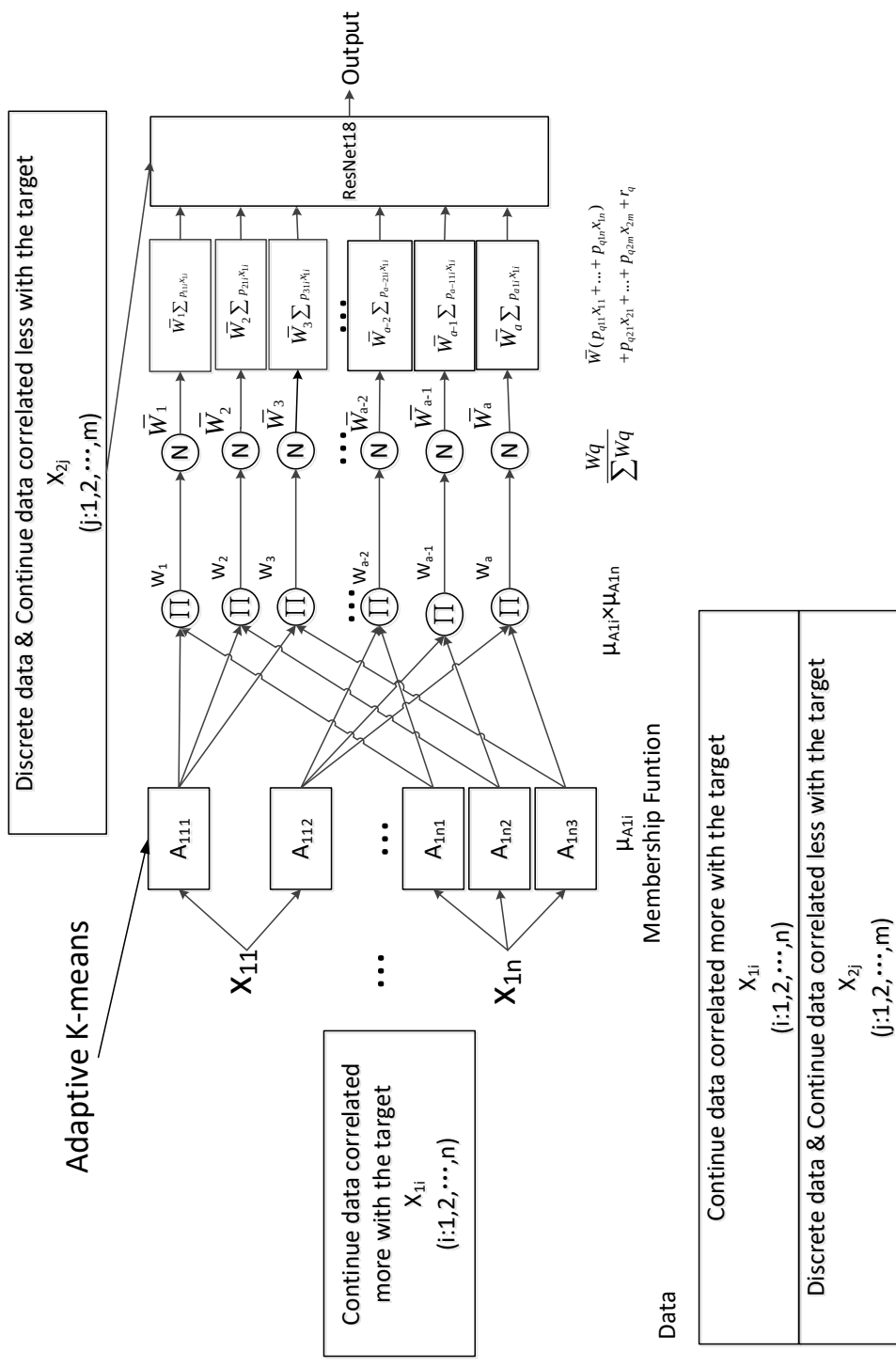


Figure 5.1: The Architecture of The Proposed Method

Adam optimizer (Javaid et al., 2016b) is used to train the parameters in ResNet. In the ANFIS-part, the Gradient Descent Only (SGD) is used to update all the parameters, as demonstrated in Section 2.5.2.

Adam optimizer is an optimization algorithm used in deep learning and machine learning for gradient-based optimization of stochastic objective functions. Adam stands for "Adaptive Moment Estimation", and it is an extension of the SGD optimizer that uses the first and second moments of the gradients to adapt the learning rate of each weight parameter.

Adam optimizes the parameters of a model by iteratively updating them based on gradients computed from the loss function. It maintains separate exponential moving averages of the squared gradients for each parameter, and it uses these averages to compute an adaptive learning rate for each parameter.

Adam also includes a momentum term, similar to the momentum used in SGD (Stochastic Gradient Descent), which helps the algorithm to traverse through local minima more quickly and effectively. The momentum term is computed as the product of the first moment estimate (the gradient) and the previous update.

Adam is considered an adaptive learning rate optimization algorithm, which means that it automatically adjusts the learning rate based on the gradient magnitudes observed during training. The algorithm maintains an exponentially decaying average of past gradients, called the first moment or the mean, and a second exponentially decaying average of past squared gradients, called the second moment or the variance. These statistics are used to calculate the update of the parameters during each iteration of training.

The Adam optimizer is employed to optimize the parameters in ResNet18. And, the loss is backpropagated to Layer 4, the $\bar{W}P_1$ and P_2 are calculated by the Adam optimizer. Then, the $\bar{W}$ is passed to Layer 1, where the *sigmas* and *means* are optimized using SGD (learning rate=1e-4, momentum=0.99).

Algorithm 2 The proposed algorithm

Input: $X = (X_1, X_2) \in \mathbb{R}^{tm \times td}$. $X_1 \in \mathbb{R}^{tm \times n}$ is the selected continuous data by Pearson correlation analysis, $X_2 = (X_{dis}, X_{con_other})$, $X_{dis} \in \mathbb{R}^{tm \times d_2}$ is discrete data, $X_{con_other} \in \mathbb{R}^{tm \times d_3}$ is the rest continuous data apart from the attributes in X_1 . The total input data's dimension is $tm \times td$. tm is the total item number of input data. td is the attribute number of input data, including the n , d_2 , and d_3 , $td = n + d_2 + d_3$. The number of attributes in X_2 is m , $m = d_2 + d_3$, $td = n + m$.

Output: Classification result $Y \in \mathbb{R}^{tm}$.

Step 1. $ANFIS(X_1) \rightarrow (WF)$:

- The original Gaussian membership functions (formed by *means* and *sigmas* after Adaptive K-means).

$$\mu(x) = e^{-\frac{1}{2}(\frac{x-mean}{sigma})^2} \quad \text{Equation 5.4}$$

- w_q , firing strengths of q -th rule, as follows:

$$w_q = \mu_{A_{11k}}(x_{11}) \times \dots \times \mu_{A_{1nk}}(x_{1n}) \quad \text{Equation 5.5}$$

- $\bar{w}_q$, firing strengths of q -th rule, as follows:
-

$$\bar{w}_q = \frac{w_q}{\sum_{q=1}^a w_q} \quad \text{Equation 5.6}$$

Where a is the number of rules.

● $WF = \{WF_1, WF_2, \dots, WF_a\}$, $WF_q = \sum_{i=1}^n \bar{w}_q f_{qi}$, $\bar{W} = \{\bar{w}_1, \bar{w}_2, \dots, \bar{w}_a\}$,
 $P_{1q} \in P_1 = \{P_{11}, P_{12}, \dots, P_{1a}\}$, $P_{1q} = \{p_{q11}, p_{q12}, \dots, p_{q1n}\}$. Weighted rules set, each weighted rule can be calculated as follows:

$$\bar{w}_q f_{qi} = \bar{w}_q p_{q1i} x_{1i} \quad \text{Equation 5.7}$$

$$WF = \bar{W} P_1 X_1 \quad \text{Equation 5.8}$$

Where $i = 1, \dots, n$.

Step 2. *ResNet*(WF, X_2) $\rightarrow (Y')$:

● Concatenate WF, X_2 (X_{dis} and X_{con_other}) as X' :

$$X' = \text{Concatenate}(WF, X_2)$$

Equation 5.9

$$\text{or } X' = \text{Concatenate}(WF, X_{dis}, X_{con_other})$$

● Calculate ***ResNet18***(X') to get Y' . The structure of ResNet18 is shown in Figure 2.6. Each node of the first layer to handle the inputs for ResNet18 is calculated as follows:

$$X_1 = WF + P_{21}X_{dis} + P_{22}X_{con_other} + R$$

Equation 5.10

$$\text{or } X_1 = WF + P_2 X_2 + R$$

$$\text{or } X_1 = \overline{W}P_1X_1 + P_2X_2 + R$$

Where $P_2 = \{P_{21}, P_{22}\}$. X_1 is the output of ResNet18's first layer. $P_{1q} \in P_1 = \{P_{11}, P_{12}, \dots, P_{1a}\} \in \mathbb{R}^{a \times n}$, $P_{1q} = \{p_{q11}, p_{q12}, \dots, p_{q1n}\}$. $P_{2q} \in P_2 = \{P_{21}, P_{22}, \dots, P_{2a}\} \in \mathbb{R}^{a \times m}$, $P_{2q} = \{p_{q21}, p_{q22}, \dots, p_{q2m}\}$. $P_1 \in \mathbb{R}^{a \times d_2}$, $P_2 \in \mathbb{R}^{a \times d_3}$. $R = \{r_q\}$ is the node bias set of $\{r_1, r_2, \dots, r_a\}$.

Step 3. *Backpropagation*(Y', Y):

- Adam optimizer to transfer the gradient to Layer 5. Then the gradient divides into two parts:

The $P = \{P_1, P_2\}$, R and other parameters in ResNet18 are trained by Adam (learning rate=0.001).

The $\overline{W}P_1$ and P_2 are calculated by the Adam optimizer.

- Pass the $\overline{W}$ to Layer 1. Use the SGD (learning rate=1e-4, momentum=0.99) to train the *sigmas* and *means*. The learning process of ANFIS is demonstrated in Section 2.5.2. Gradient Descent Only is utilized to update the parameters.

Step 4. *Verify_Loss*(Y', Y):

- Calculate the loss of (Y', Y) on each epoch.
 - Early Stopping mechanism (patience=5). If the loss keeps no improvement after 5 epochs, stop training. Specifically, the training of the model is stopped if the validation loss does not improve for 5 consecutive epochs. The reason for choosing a
-

patience of 5 is the same as described in Algorithm 1.

During the training process, the continuous attribute x_{1i} is divided into overlapped intervals using standard deviation and adaptive K-means, resulting in fuzzy membership degrees for each interval. These membership degrees represent the degree of belonging to the corresponding interval, introducing a level of fuzziness to the algorithm. Fuzzy rules are then generated by taking the product of the membership degrees for the relevant intervals, representing different combinations of selected continuous attributes.

The firing strengths (w_q) of these fuzzy rules are normalized, and weighted fuzzy rules are constructed to indicate their contribution to the target. These weighted fuzzy rules, along with the discrete data and remaining continuous attributes, are concatenated and used as input for ResNet. By incorporating fuzzy intervals and fuzzy rules into the ResNet architecture, the proposed method introduces more fuzziness and allows for deep pattern extraction.

To evaluate the performance of the proposed method, it will be tested on the validation dataset (KDDTrain+ subset) and the test dataset (KDDTest+ subset). The trained model will be applied to these datasets to obtain the predicted results ($valid_Y'$ and $test_Y'$). The performance of the model will be evaluated by comparing the predicted results with the actual values using the metrics discussed in Section 3.5.

In summary, the proposed method combines fuzzy rules generated by ANFIS with ResNet to perform deep pattern extraction and enhance the fuzziness of the algorithm. The

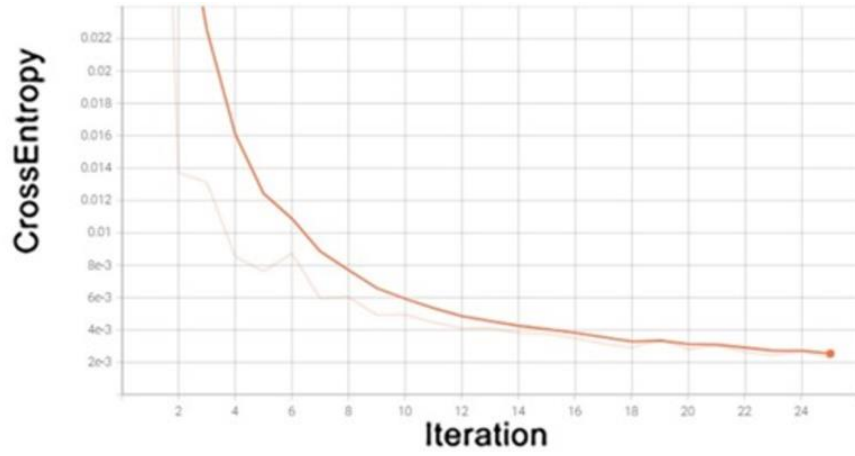
performance of the method will be evaluated on the validation and test datasets using appropriate metrics.

5.3 Experimental Results

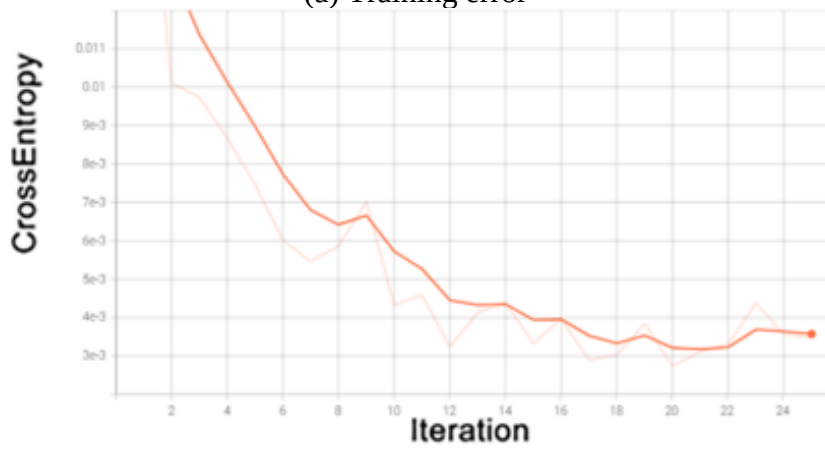
In this study, the proposed method applied the adaptive K-means and standard deviation techniques for attribute selection and overlapped interval initialization of continuous attributes, as discussed in Section 4.2. The method was then trained on 80% of the KDDTrain+ dataset, with the remaining portion used as the validation set. The KDDTest+ dataset was used as the test set to evaluate the performance of the proposed method. The implementation of the method was done using Python 3, utilizing PyTorch and sklearn libraries.

The proposed method utilizes Algorithm 2 to generate fuzzy rules by combining selected continuous attributes through the ANFIS-part of the architecture. The ResNet-part of the method is responsible for identifying deep patterns within the generated fuzzy rules and the discrete data. The entire architecture is co-trained using the loss function, which represents the discrepancy between the actual target variable Y and the predicted target variable Y' . Figure 5.2 illustrates the changes in training, validation, and testing errors during the training process. Initially, there is some overfitting observed, but after 8 epochs, the overfitting is reduced, and the model maintains good performance. The after-trained p_{qli} values for the four attributes and the r values for each rule are provided in Appendix D.

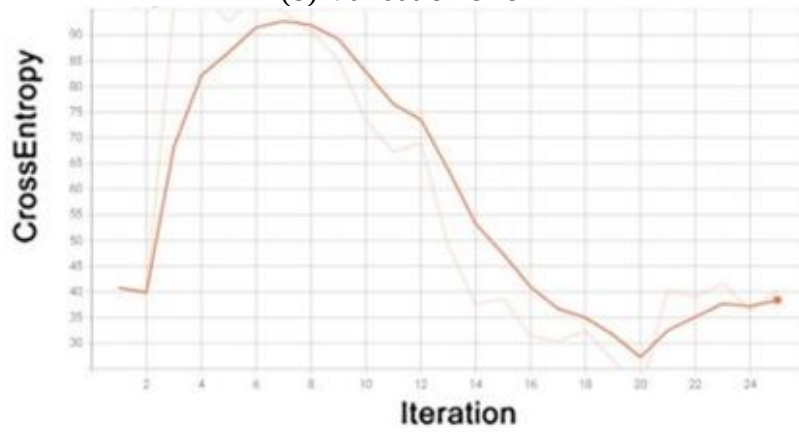
In summary, the proposed method combines the ANFIS-part and the ResNet-part, with co-training using the loss function, to generate and analyze fuzzy rules for intrusion detection. The method's performance is evaluated on the validation and test sets, and the implementation is done using Python 3, PyTorch, and sklearn libraries.



(a) Training error



(b) Validation error



(c) Testing error

Figure 5.2: CrossEntropy Convergence Curve of the Proposed Method

The performance of the proposed method on various metrics is presented in Table 5.1 and Table 5.2.

Table 5.1: Experimental Results for Different Performance Metrics in Five Categories of The KDDTrain+ Dataset

Category	Precision (%)	Recall (%)	F1-score(%)	DR(%)	FAR(%)	ACC(%)
Normal	99.11	98.91	99.01	98.91	0.89	99.01
DOS	99.61	99.88	99.74	99.88	0.39	99.74
PROBE	95.32	96.89	96.10	96.89	4.76	96.07
R2L	91.78	77.39	83.97	77.39	6.93	85.23
U2R	100	46.15	63.16	46.15	0.00	73.08
weighted avg	98.88	98.89	98.87	98.88	1.11	98.88

Table 5.2: Experimental Results for Different Performance Metrics in Five Categories of The KDDTest+ Dataset

Category	Precision (%)	Recall (%)	F1-score(%)	DR(%)	FAR(%)	ACC(%)
Normal	68.88	96.67	80.44	96.67	43.68	76.50
DOS	93.14	79.92	86.03	79.92	5.89	87.02
PROBE	68.69	65.96	67.30	65.96	30.07	67.95
R2L	88.83	5.77	10.84	5.77	0.73	52.52
U2R	80.00	4.00	7.62	4.00	7.62	51.50
weighted avg	79.42	75.91	71.73	75.90	24.15	75.91

5.4 Comparative Analysis

This study conducted a comparative analysis of the proposed method with other state-of-the-art approaches, including MVO-ANN (Benmessahel et al., 2019a), FC-ANN (Manimurugan et al., 2020), DNN(Tang et al., 2016), CSO-ANFIS (Manimurugan et al.,

2020), and original ResNet, on the same dataset. The classification performance of the five main categories was evaluated, and the results are presented in Table 5.3 and Table 5.4.

Table 5.3: Detection Rate, ACC, and FAR Comparison of Different Classifiers on KDDTrain+

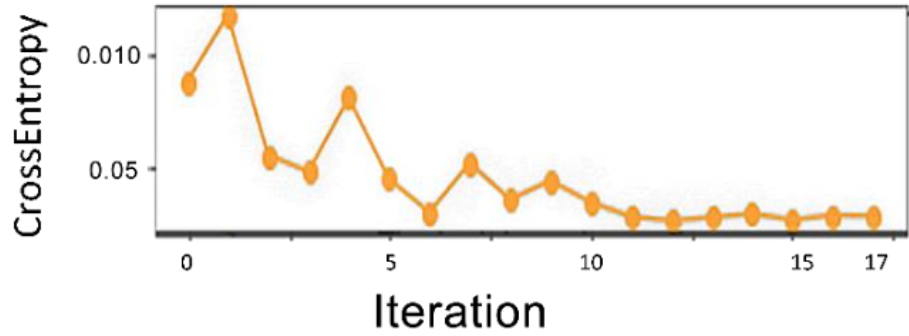
Classifier	DR	FAR	ACC	F1-score	Recall	Precision
MVO-ANN (Benmessahel et al., 2019a)	96.25	0.032	98.21			
FC-ANN (Manimurugan et al., 2020)	89.23	4.37				
DNN(Tang et al., 2016)			91.7			
CSO-ANFIS (Manimurugan et al., 2020)	95.80	3.45				
DBN+ResNet101 (G. N. Nguyen et al., 2021)			98.95	97.62	98.76	98.94
ResNet	43.42	56.58	43.42			
ANFIS	88.20	11.79	88.20	87.22	88.20	88.28
Proposed Model	98.88	1.11	98.88	98.87	98.89	98.88

Table 5.4: ACC Comparison of Different Classifiers on KDDTrain+

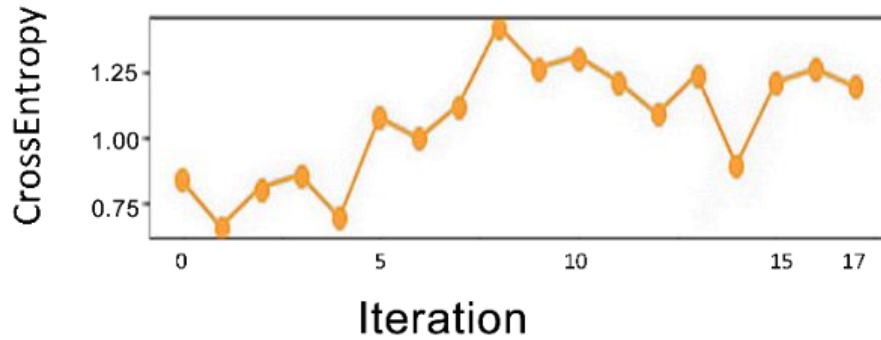
Classifier	Normal	DOS	PROBE	R2L	U2R
BPNN (Manimurugan et al., 2020)	91.50	90.94	80.53	48.13	60.00
GA-DBN(Y. Zhang et al., 2019)		99.45	97.78	99.37	98.68
FC-ANN (Manimurugan et al., 2020)	93.87	80.32	89.12	89.57	75.58
GA-ANFIS (Manimurugan et al., 2020)	96.22	96.70	93.18	94.89	83.33
PSO-ANFIS (Manimurugan et al., 2020)	96.46	95.90	91.35	94.72	85.62
CSO-ANFIS (Manimurugan et al., 2020)	97.41	96.25	92.51	94.15	90.26
ResNet	53.66	38.69	3.53	35.48	50.00
ANFIS	93.19	85.94	71.79	49.95	50.00
Proposed Model	99.01	99.74	96.07	73.08	98.88

In this study, the performance of the original ResNet was compared to the proposed method on the NSL-KDD dataset. However, the results showed a significant overfitting problem with the original ResNet on this dataset, as shown in Figure 5.3. This finding is consistent with the observation by Xiao and Xiao (2019), who highlighted the tendency of deep learning models to overfit on low-dimensional and small-scale datasets, resulting in a reduced ability to generalize to new data. In this study, it is suggested that one of the reasons for the overfitting of the original ResNet on the NSL-KDD dataset could be its tendency to overfit on low-dimensional and small-scale datasets, which can lead to a reduction in the model's generalization ability. Additionally, the highly detailed classification of deep learning can exacerbate overfitting on problems with uncertainty. To address this issue, the ANFIS is used in this study to transform the excessively precise values into fuzzy values,

which helps to overcome the over-detailed data classification and improve the model's generalization ability.



(a) CrossEntropy training error of ResNet



(b) CrossEntropy testing error of ResNet

Figure 5.3: Evaluation Results of ResNet on NSL-KDD Dataset

5.5 Discussion

The experimental results on the KDDTrain+ dataset demonstrate the high performance of the proposed method, with a detection rate of 98.88% and a false alarm rate of 1.11%, as shown in Table 5.1. On the KDDTest+ dataset, the proposed method shows a certain level of generalization ability, achieving a detection rate of 75.9%, precision of 79.42%, and accuracy of 75.91%, as indicated in Table 5.2. However, the false alarm rate on the KDDTest+ dataset is relatively high at 24.15%, indicating the need for further improvements.

In terms of comparison with other methods, the proposed method outperforms single ANFIS and ResNet models, achieving higher detection rates and lower false alarm rates, as shown in Table 5.3 and Table 5.4. This indicates that the proposed method improves upon the individual strengths of ANFIS and ResNet, addressing the limitations of each model. Compared to other methods, such as MVO-ANN (Benmessahel et al., 2019a), DBN+ResNet101 (G. N. Nguyen et al., 2021), CSO-ANFIS, BPNN, and GA-DBN, the proposed method demonstrates superior performance in terms of accuracy and detection rates in various categories.

The results highlight the effectiveness of the proposed method in enhancing the performance of intrusion detection compared to individual models and other existing approaches. The combination of ANFIS and ResNet, along with the co-training mechanism, contributes to improved detection rates and reduced false alarm rates, demonstrating the potential of the proposed method for intrusion detection tasks. However, further refinement is necessary to address the false alarm rate on the KDDTest+ dataset.

The proposed method in this study improves the performance of ANFIS and alleviates the overfitting problem of ResNet to some extent. However, what is more promising for the future is its deeper feature recognition ability. With the increasing complexity of network environments and intrusion scenarios, more factors and complex intrusion patterns will come into play, making it necessary to have a deeper network that can recognize features effectively. Therefore, compared to the previous combination of ANFIS and CART, the proposed method will have more usability in the future.

However, a significant issue with this model is that ResNet is a black box model, with limited interpretability compared to the previous proposed model. Future research can

focus on developing explainable deep learning models that retain the high performance of black box models while providing insight into the decision-making process, such as using attention mechanisms, integrating interpretable modules, or developing post-hoc explanation methods.

Additionally, the architecture of the ResNet component needs to be further refined. The current implementation of ResNet18 might not be optimal for intrusion detection tasks, so more research is required to investigate different architectures or variations of ResNet that can enhance performance.

Moreover, the proposed method can be further improved by incorporating more advanced techniques such as transfer learning or domain adaptation. Transfer learning allows the model to leverage knowledge from pre-trained models on related tasks, while domain adaptation addresses the problem of domain shift by adapting the model to the target domain.

Currently, the experimental research is only tested on NSL-KDD dataset. If the system is tested in a realistic environment, the dimensionality of the influencing factors will be very high, and the intrusion patterns will be even more complex, with intricate relationships among various factors. Therefore, deeper layers will be necessary to capture the patterns. Therefore, this model is expected to have greater potential for future applications compared to the previous combination of ANFIS and CART.

5.6 Chapter Summary

The chapter describes a new intrusion detection method based on a hybrid deep learning model that combines ANFIS and ResNet. The proposed method is tested on the NSL-KDD dataset and compared with other existing intrusion detection methods.

The experimental results demonstrate that the proposed method surpasses other methods in various performance metrics, including detection rate, false alarm rate, F1-score, recall, accuracy, and precision. Specifically, the proposed method achieves a high detection rate of 98.88% and a low false alarm rate of 1.11% on the KDDTrain+ dataset, and a total detection rate of 75.9%, precision of 79.42%, and accuracy of 75.91% on the KDDTest+ dataset.

The proposed method effectively enhances the performance of ANFIS and mitigates the overfitting issue associated with ResNet. However, it is worth noting that the proposed method lacks interpretability, as ResNet is considered a black-box model. This limitation suggests that the proposed method may have challenges in explaining the underlying decision-making process or providing clear insights into the reasons behind its predictions.

Compared to the method proposed in Chapter 4, the proposed method in this chapter is better suited for complex network environments with increasingly high dimensional and evolving intrusion patterns. However, further research is necessary to improve its interpretability and applicability in real-world scenarios. The proposed method shows promising potential for enhancing intrusion detection performance in the future, as it can identify deeper-level features of network traffic.

CHAPTER 6

CONCLUSIONS AND RECOMMENDATIONS

6.1 Conclusion

The objective of this research was to extend the capabilities of ANFIS to perform deeper pattern extraction tasks. The study has two main contributions. Firstly, the proposed method combines ANFIS with CART, resulting in a deeper, fuzzy, and interpretable structure for intrusion detection. Secondly, the study introduces a ResNet-enhanced ANFIS structure, which deepens the structure of ANFIS and improves ResNet's ability to handle fuzzy characteristics.

Chapter 4 of this study introduces a novel method that combines ANFIS and CART for deeper and interpretable pattern extraction in intrusion detection. The proposed method offers an interpretable and deeper model compared to using ANFIS or CART individually, enabling the detection of deeper and fuzzier intrusion patterns. Section 4.5 of the chapter focuses on interpreting the proposed model, providing insights into the reasoning and decision-making process.

The performance of the proposed method is evaluated by comparing it to other algorithms such as BPNN, FC-ANN, GA-ANFIS, PSO-ANFIS, and CSO-ANFIS. The results demonstrate the superiority of the proposed method, achieving a detection rate of 99.86% and a false alarm rate of 0.14% on the KDDTrain+ dataset, outperforming the other methods. The performance of the proposed method is further evaluated on the KDDTest+ dataset, which contains unseen intrusion patterns. The comparison with single CART,

ANFIS, and other methods highlights the improved performance of the proposed method in detecting these intrusion patterns.

In Chapter 5, the primary focus of this study was on proposing an enhanced architecture of ANFIS, known as ANFIS-ResNet, for deeper pattern extraction tasks. The proposed architecture integrates the modified ANFIS and ResNet to generate a deeper and more effective structure. Through a comparison with other existing algorithms, including BPNN, GA-DBN, DNN, FC-ANN, GA-ANFIS, PSO-ANFIS, CSO-ANFIS, original ANFIS, and original ResNet, the proposed method was found to achieve a higher detection rate and lower false alarm rate on the KDDTrain+ dataset. Furthermore, the improved performance on various metrics demonstrated the enhanced generalization ability of the proposed ANFIS-ResNet architecture. The study suggests that the proposed method has the potential to improve intrusion detection performance in more complex and high-dimensional network environments.

In this study, the CART-enhanced ANFIS model provided a deeper, fuzzy, and interpretable structure that was useful for reading deep-level features of network traffic and analyzing intrusions. The ANFIS-CART model achieved high detection and low false alarm rates on the KDDTrain+ dataset, demonstrating its outstanding performance. However, the CART-enhanced ANFIS model also had some drawbacks, such as its inability to identify deeper-level information.

The ResNet-enhanced ANFIS model generated a deep architecture by connecting the modified ANFIS and ResNet to improve the deep pattern extraction ability of ANFIS. Compared with other algorithms, the ANFIS-ResNet model performed well on various metrics and exhibited stronger generalization ability. On the KDDTrain+ dataset, the

ANFIS-ResNet model achieved high detection and low false alarm rates. However, compared with the CART-enhanced ANFIS model, the ResNet-enhanced ANFIS model had less interpretability, and constructing and adjusting such a deep network structure may require more computational resources and time. Nevertheless, it has great potential for handling increasingly complex network environments and intrusions in the future.

6.2 Future Works

Currently, this study has proposed several improvements for fuzzy classification problems, especially for intrusion detection. However, the work presented in this study has several limitations that need to be addressed in future research.

- i. **Data Limitations:** The success of the proposed approach heavily depends on the quality and diversity of the training data. Limited availability of high-quality labeled data may limit the performance of the model, especially when dealing with novel or rare classes.
- ii. **Algorithm Limitations:** The computational complexity of the proposed algorithm may be high, making it unsuitable for real-time or large-scale applications. Improving the efficiency of the algorithm is a key area for future research.
- iii. **Experimental Limitations:** The evaluation of the proposed approach was conducted using a limited number of datasets and experimental settings. Future research should include additional datasets and experimental conditions to validate the generalizability of the approach.

- iv. Generalizability: The proposed approach was trained and evaluated using a specific dataset or problem domain. Its generalizability to other datasets or problem domains is unknown and needs to be further investigated.

Deep fuzzy classification methods have demonstrated their effectiveness in various applications, such as intrusion detection and image classification. However, to fully unleash their potential, future research needs to focus on further improving their accuracy, efficiency, privacy, and interpretability. This may involve exploring new network architectures, incorporating more advanced fuzzy logic techniques, and leveraging federated learning and other emerging technologies. With continued research and development, deep fuzzy classification methods have the potential to significantly enhance the performance and intelligence of various fuzzy classification systems, and enable more sophisticated decision-making in diverse fields.

Therefore, further investigation is necessary. In future research, the emerging problems in the process of this study will continue to be studied. The improvement mainly lies in the following aspects:

- i. Data Augmentation and Curation: To address the data limitations, future work could focus on data augmentation and curation techniques. This includes exploring methods to generate synthetic data or applying transfer learning techniques to leverage knowledge from related domains where labeled data is more abundant. Additionally, active learning strategies can be employed to efficiently select and label the most informative data samples.
- ii. Algorithm Optimization: To overcome the computational complexity of the proposed algorithm, future research could investigate algorithm optimization

techniques. This may involve exploring parallel processing methods, implementing more efficient data structures, or utilizing hardware acceleration (e.g., GPUs or TPUs) to speed up computations. Additionally, algorithm hyperparameters can be fine-tuned to balance performance and computational efficiency.

- iii. **Extensive Experimental Evaluation:** To address the experimental limitations, future work should aim to conduct a more comprehensive experimental evaluation. This includes evaluating the proposed approach on a wider range of datasets, exploring different experimental settings, and comparing it with a broader set of baseline methods. Such an evaluation will provide a more accurate assessment of the approach's performance and generalizability.
- iv. **Generalizability and Adaptability:** To investigate the generalizability of the proposed approach, future research should explore its application to diverse datasets and problem domains. This may involve adapting the model to handle different types of input data, incorporating domain-specific knowledge, or developing transfer learning strategies to reuse learned representations across domains.
- v. **Interpretability and Explainability:** As the adoption of machine learning models in critical applications increases, interpretability and explainability become crucial. Future work could focus on enhancing the interpretability of the proposed model by incorporating explainable AI techniques. This could include developing visualization tools, generating human-understandable

explanations for model predictions, or exploring model-agnostic interpretability methods.

- vi. **Real-world Applications:** Finally, future research should aim to deploy and evaluate the proposed approach in real-world scenarios. This involves collaborating with domain experts to understand practical challenges and requirements, developing deployment pipelines, and conducting field tests to assess the model's performance in real-world conditions.

By addressing these future work directions, we can further improve the performance, efficiency, generalizability, and interpretability of the proposed approach, bringing it closer to real-world applicability and impact.

REFERENCES

- Abu Alghanam, O., Almobaideen, W., Saadeh, M., & Adwan, O. (2023). An improved PIO feature selection algorithm for IoT network intrusion detection system based on ensemble learning. *Expert Systems with Applications*, 213, 118745.
- Al-Qatf, M., Lasheng, Y., Al-Habib, M., & Al-Sabahi, K. (2018). Deep Learning Approach Combining Sparse Autoencoder with SVM for Network Intrusion Detection. *IEEE Access*, 6, 52843–52856.
- Ali Alheeti, K.M., Gruebler, A., & McDonald-Maier, K.D. (2015). An intrusion detection system against malicious attacks on the communication network of driverless cars. In: *CCNC, 2015 12th Annual IEEE Consumer Communications and Networking Conference*, Las Vegas, NV, USA, December 2015. IEEE.
- Aljawarneh, S., Aldwairi, M., & Yassein, M. B. (2018). Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model. *Journal of Computational Science*, 25, 152–160.
- Alom, M.Z., Bontupalli, V., & Taha, T.M. (2015). Intrusion detection using deep belief networks. In: *NAECON, 2015 National Aerospace and Electronics Conference*, Dayton, OH, USA, June 2015. IEEE.
- Alrayes, F. S., Alshuqayran, N., Nour, M. K., Al Duhayyim, M., Mohamed, A., Mohammed, A. A. A., ... Yaseen, I. (2023). Optimal Fuzzy Logic Enabled Intrusion Detection for Secure IoT-Cloud Environment. *Computers, Materials and Continua*, 74, 6737–6753.
- Alsaadi, H. I. H., ALmuttari, R. M., Ucan, O. N., & Bayat, O. (2022). An adapting soft computing model for intrusion detection system. *Computational Intelligence*, 38, 855–875.

- Althubiti, S.A., Jones, E.M., & Roy, K. (2019). LSTM for anomaly-based network intrusion detection. In: ITNAC, *2018 28th International Telecommunication Networks and Applications Conference*, Sydney, NSW, Australia, November 2019. IEEE.
- Amiri, F., Rezaei Yousefi, M., Lucas, C., Shakery, A., & Yazdani, N. (2011). Mutual information-based feature selection for intrusion detection systems. *Journal of Network and Computer Applications*, 34, 1184–1199.
- Amor, N. Ben, Benferhat, S., & Elouedi, Z. (2004). Naive Bayes vs decision trees in intrusion detection systems. In: ACM, *Proceedings of the 2004 ACM symposium on Applied computing*, New York, NY, USA, March 2004. ACM.
- Anderson, J. P. (1980). Computer security threat monitoring and surveillance. *Technical Report James P Anderson Co Fort Washington Pa, 19034*, 215 646-4706.
- Ashfaq, R. A. R., Wang, X. Z., Huang, J. Z., Abbas, H., & He, Y. L. (2017). Fuzziness based semi-supervised learning approach for intrusion detection system. *Information Sciences*, 378, 484–497.
- Badrinarayanan, V., Kendall, A., & Cipolla, R. (2017). SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39, 2481–2495.
- Balyan, A. K., Ahuja, S., Lilhore, U. K., Sharma, S. K., Manoharan, P., Algarni, A. D., ... Raahemifar, K. (2022). A Hybrid Intrusion Detection Model Using EGA-PSO and Improved Random Forest Method. *Sensors*, 22, 1–20.
- Benmessahel, I., Xie, K., Chellal, M., & Semong, T. (2019a). A new evolutionary neural networks based on intrusion detection systems using locust swarm optimization. *Evolutionary Intelligence*, 12, 131–146.
- Benmessahel, I., Xie, K., Chellal, M., & Semong, T. (2019b). A new evolutionary neural

- networks based on intrusion detection systems using locust swarm optimization. *Evolutionary Intelligence*, 12, 131–146.
- Bhattacharya, S., S, S. R. K., Maddikunta, P. K. R., Kaluri, R., Singh, S., Gadekallu, T. R., ... Tariq, U. (2020). A Novel PCA-Firefly Based XGBoost Classification Model for Intrusion Detection in Networks Using GPU. *Electronics*, 9, 219.
- Blockeel, H., Devos, L., Frénay, B., Nanfack, G., & Nijssen, S. (2023). Decision trees: from efficient prediction to responsible AI. *Frontiers in Artificial Intelligence*, 6, 1124553.
- Boughaci, D., Kadi, M. D. E., & Kada, M. (2012). *Lecture Notes in Computer Science*, Doha: ICONIP.
- Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (2017). *Classification And Regression Trees*, New York: Routledge.
- Camastra, F., Ciaramella, A., & Staiano, A. (2013). Machine learning and soft computing for ICT security: An overview of current trends. *Journal of Ambient Intelligence and Humanized Computing*, 4, 235–247.
- Casas, P., Mazel, J., & Owezarski, P. (2012). Unsupervised Network Intrusion Detection Systems: Detecting the Unknown without Knowledge. *Computer Communications*, 35, 772–783.
- Chandrasekhar, A.M., & Raghuveer, K. (2013). Intrusion detection technique by using k-means, fuzzy neural network and SVM classifiers. In: IEEE, *2013 International Conference on Computer Communication and Informatics*, Coimbatore, India, January 2013. IEEE.
- Chen, T., Shang, C., Su, P., Keravnou-Papailiou, E., Zhao, Y., Antoniou, G., & Shen, Q. (2021). A Decision Tree-Initialised Neuro-fuzzy Approach for Clinical Decision Support. *Artificial Intelligence in Medicine*, 111, 101986.

- Chen, T., Su, P., Shen, Y., Chen, L., Mahmud, M., Zhao, Y., & Antoniou, G. (2022). A dominant set-informed interpretable fuzzy system for automated diagnosis of dementia. *Frontiers in Neuroscience*, 16, e867664.
- Chen, Y., Abraham, A., & Yang, B. (2007). Hybrid flexible neural-tree-based intrusion detection systems. *International Journal of Intelligent Systems*, 22, 337–352.
- Chidananda Murthy, P., Manjunatha, A. S., Jaiswal, A., & Madhu, B. R. (2016). Building efficient classifiers for intrusion detection with reduction of features. *International Journal of Applied Engineering Research*, 11(6), 4590–4596.
- CINIC. (2022). *The 50th Statistical Report on the Development Status of the Internet in China* [online] Available at <http://www.cnnic.net.cn/n4/2022/0914/c88-10226.html> [Assessed on November 14, 2022].
- Cisco. (2020). *Cisco Annual Internet Report—Cisco Annual InternetReport (2018–2023) White Paper* [online] Available at <https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html> [Assessed on April 6, 2022].
- De la Hoz, E., De La Hoz, E., Ortiz, A., Ortega, J., & Prieto, B. (2015). PCA filtering and probabilistic SOM for network intrusion detection. *Neurocomputing*, 164, 71–81.
- Denning, D. E. (1987). An Intrusion-Detection Model. *IEEE Transactions on Software Engineering*, SE-13, 222–232.
- Drewek-Ossowicka, A., Pietrolaj, M., & Rumiński, J. (2021). A survey of neural networks usage for intrusion detection systems. *Journal of Ambient Intelligence and Humanized Computing*, 12, 497–514.
- Du, M., Li, F., Zheng, G., & Srikumar, V. (2017). DeepLog: Anomaly detection and diagnosis from system logs through deep learning. *Proceedings of the ACM Conference*

- on *Computer and Communications Security*, 1285–1298.
- Elhag, S., Fernández, A., Bawakid, A., Alshomrani, S., & Herrera, F. (2015). On the combination of genetic fuzzy systems and pairwise learning for improving detection rates on Intrusion Detection Systems. *Expert Systems with Applications*, 42, 193–202.
- Erfani, S. M., Rajasegarar, S., Karunasekera, S., & Leckie, C. (2016). High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning. *Pattern Recognition*, 58, 121–134.
- Fatani, A., Dahou, A., Al-Qaness, M. A. A., Lu, S., & Elaziz, M. A. (2022). Advanced feature extraction and selection approach using deep learning and aquila optimizer for iot intrusion detection system. *Sensors*, 22, 140.
- Feng, W., Zhang, Q., Hu, G., & Huang, J. X. (2014). Mining network data for intrusion detection through combining SVMs with ant colony networks. *Future Generation Computer Systems*, 37, 127–140.
- Fernandez Maimo, L., Perales Gomez, A. L., Garcia Clemente, F. J., Gil Perez, M., & Martinez Perez, G. (2018). A Self-Adaptive Deep Learning-Based System for Anomaly Detection in 5G Networks. *IEEE Access*, 6, 7700–7712.
- Ferrag, M. A., Maglaras, L., Moschoyiannis, S., & Janicke, H. (2020). Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study. *Journal of Information Security and Applications*, 50, 102419.
- Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... Bengio, Y. (2020). Generative adversarial networks. *Communications of the ACM*, 63, 139–144.
- Gorzalczany, M. B., & Rudzinski, F. (2022). Intrusion Detection in Internet of Things With MQTT Protocol - An Accurate and Interpretable Genetic-Fuzzy Rule-Based Solution.

- IEEE Internet of Things Journal*, 9, 24843–24855.
- Guo, C., Zhou, Y., Ping, Y., Zhang, Z., Liu, G., & Yang, Y. (2014). A distance sum-based hybrid method for intrusion detection. *Applied Intelligence*, 40, 178–188.
- Gupta, A. R., & Agrawal, J. (2021). The multi-demeanor fusion based robust intrusion detection system for anomaly and misuse detection in computer networks. *Journal of Ambient Intelligence and Humanized Computing*, 12, 303–319.
- Hassan, M. M., Gumaei, A., Alsanad, A., Alrubaian, M., & Fortino, G. (2020). A hybrid deep learning model for efficient intrusion detection in big data environment. *Information Sciences*, 513, 386–396.
- He, K., Zhang, X., Ren, S., & Sun, J. (2016a). Deep residual learning for image recognition. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2016-Decem*, 770–778.
- He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2016-Decem*, 770–778.
- Hinton, G. E., Osindero, S., & Teh, Y. W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, 18, 1527–1554.
- Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 9, 1735–1780.
- Hodo, E., Bellekens, X., Hamilton, A., Dubouilh, P.-L., Iorkyase, E., Tachtatzis, C., & Atkinson, R. (2016). Threat analysis of IoT networks using artificial neural network intrusion detection system. In: *ISNCC, 2016 International Symposium on Networks, Computers and Communications*, Yasmine Hammamet, Tunisia, May 2016. IEEE.
- Hodo, E., Bellekens, X., Hamilton, A., Dubouilh, P. L., Iorkyase, E., Tachtatzis, C., &

- Atkinson, R. (2016b). Threat analysis of IoT networks using artificial neural network intrusion detection system. *2016 International Symposium on Networks, Computers and Communications, ISNCC 2016*, 4–9.
- Horng, S.-J., Su, M.-Y., Chen, Y.-H., Kao, T.-W., Chen, R.-J., Lai, J.-L., & Perkasa, C. D. (2011). A novel intrusion detection system based on hierarchical clustering and support vector machines. *Expert Systems with Applications*, 38, 306–313.
- Huang, S. (2018). *KDD Cup 1999 Data | Kaggle* [online] Available at <https://www.kaggle.com/datasets/galaxyh/kdd-cup-1999-data> [Assessed on May 11, 2021].
- Iandola, F.N., Han, S., Moskewicz, M.W., Ashraf, K., Dally, W.J., & Keutzer, K. (2016). SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size. In: *ICLR, 2017 International Conference on Learning Representations*, Toulon, France, November 2017. ICLR.
- Ibrahim, L. M., Taha, D. B., & Mahmod, M. S. (2013). A comparison study for intrusion database (KDD99, NSL-KDD) based on self organization map (SOM) artificial neural network. *Journal of Engineering Science and Technology*, 8, 107–119.
- Ieracitano, C., Adeel, A., Morabito, F. C., & Hussain, A. (2020). A novel statistical analysis and autoencoder driven intelligent intrusion detection approach. *Neurocomputing*, 387, 51–62.
- imarc (2022). *Cybersecurity Market Report by Component (Solutions, Services), Deployment Type (Cloud-based, On-premises), User Type (Large Enterprises, Small and Medium Enterprises), Industry Vertical (IT and Telecom, Retail, BFSI, Healthcare, Defense/Government, Manuf* [online] Available at

- <https://www.imarcgroup.com/cybersecurity-market> [Assessed on December 13, 2023].
- Jang, J. S. R. (1993). ANFIS: Adaptive-Network-Based Fuzzy Inference System. *IEEE Transactions on Systems, Man and Cybernetics*, 23, 665–685.
- Javaid, A., Niyaz, Q., Sun, W., & Alam, M. (2016). A Deep Learning Approach for Network Intrusion Detection System. In: BICT, *EAI International Conference on Bio-inspired Information and Communications Technologies*, Toledo, USA, May 2016. ACM.
- Jazara (2009). *Internet World Statistics* [online] Available at <https://www.internetworldstats.com/stats.htm> [Assessed on January 9, 2023].
- Kalita, D. J., Singh, V. P., & Kumar, V. (2023). A novel adaptive optimization framework for SVM hyper-parameters tuning in non-stationary environment: A case study on intrusion detection system. *Expert Systems with Applications*, 213, 119189.
- Kang, M.-J., & Kang, J.-W. (2016). Intrusion Detection System Using Deep Neural Network for In-Vehicle Network Security. *PLOS ONE*, 11, e0155781.
- Khan, L., Awad, M., & Thuraisingham, B. (2007). A new intrusion detection system using support vector machines and hierarchical clustering. *The VLDB Journal*, 16, 507–521.
- Kim, D. S., & Park, J. S. (2003). Network-Based Intrusion Detection with Support Vector Machines. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2662, 747–756.
- Kim, G., Lee, S., & Kim, S. (2014). A novel hybrid intrusion detection method integrating anomaly detection with misuse detection. *Expert Systems with Applications*, 41, 1690–1700.
- Kim, J., Kim, J., Thi Thu, H. Le, & Kim, H. (2016). Long Short Term Memory Recurrent Neural Network Classifier for Intrusion Detection. In: *IEEE, 2016 International Conference on Platform Technology and Service (PlatCon)*, Jeju, Korea (South), 15-17

February 2016. IEEE.

- Koc, L., Mazzuchi, T. A., & Sarkani, S. (2012). A network intrusion detection system based on a Hidden Naïve Bayes multiclass classifier. *Expert Systems with Applications*, 39, 13492–13500.
- Kocher, G., & Kumar, G. (2021). Machine learning and deep learning methods for intrusion detection systems: recent developments and challenges. *Soft Computing*, 25(15), 9731–9763.
- Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, 60, 84–90.
- Kuang, F., Xu, W., & Zhang, S. (2014). A novel hybrid KPCA and SVM with GA model for intrusion detection. *Applied Soft Computing Journal*, 18, 178–184.
- Kumar, G., & Kumar, K. (2015). A multi-objective genetic algorithm based approach for effective intrusion detection using neural networks. In *Studies in Computational Intelligence*, 563, 173–200.
- Kumar, S., & Yadav, A. (2014). Increasing performance Of intrusion detection system using neural network. In: *IEEE, 2014 IEEE International Conference on Advanced Communications, Control and Computing Technologies*, Ramanathapuram, India, 08-10 May 2014. IEEE.
- Lansky, J., Ali, S., Mohammadi, M., Majeed, M. K., Karim, S. H. T., Rashidi, S., ... Rahmani, A. M. (2021). Deep Learning-Based Intrusion Detection Systems: A Systematic Review. *IEEE Access*, 9, 101574–101599.
- LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, 521, 436–444.
- LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86, 2278–2323.

- Li, F. Q., Zhao, R. J., Wang, S. L., Chen, L. B., Liew, A. W. C., & Ding, W. (2022). Online Intrusion Detection for Internet of Things Systems With Full Bayesian Possibilistic Clustering and Ensembled Fuzzy Classifiers. *IEEE Transactions on Fuzzy Systems*, 30, 4605–4617.
- Li, X. K., Chen, W., Zhang, Q., & Wu, L. (2020). Building Auto-Encoder Intrusion Detection System based on random forest feature selection. *Computers and Security*, 95, 101851.
- Li, Yang, Fang, B.-X., Guo, L., & Chen, Y. (2007). LNCS (Ed.), Intelligence and Security Informatics: Vol. 4430 LNCS, Berlin, Heidelberg: Springer Berlin Heidelberg.
- Li, Yinhui, Xia, J., Zhang, S., Yan, J., Ai, X., & Dai, K. (2012). An efficient intrusion detection system based on support vector machines and gradually feature removal method. *Expert Systems with Applications*, 39, 424–430.
- Lin, H., Wu, C., & Masdari, M. (2022). A comprehensive survey of network traffic anomalies and DDoS attacks detection schemes using fuzzy techniques. *Computers and Electrical Engineering*, 104, 108466.
- Lin, S.-W., Ying, K.-C., Lee, C.-Y., & Lee, Z.-J. (2012). An intelligent algorithm with feature selection and decision rules applied to anomaly intrusion detection. *Applied Soft Computing*, 12, 3285–3290.
- Lin, W.-C., Ke, S.-W., & Tsai, C.-F. (2015). CANN: An intrusion detection system based on combining cluster centers and nearest neighbors. *Knowledge-Based Systems*, 78, 13–21.
- Louk, M. H. L., & Tama, B. A. (2023). Dual-IDS: A bagging-based gradient boosting decision tree model for network anomaly intrusion detection system. *Expert Systems with Applications*, 213, 119030.

- Manimurugan, S., Majdi, A. qdah, Mohmmmed, M., Narmatha, C., & Varatharajan, R. (2020). Intrusion detection in networks using crow search optimization algorithm with adaptive neuro-fuzzy inference system. *Microprocessors and Microsystems*, 79, 103261.
- Masdari, M., & Khezri, H. (2020). A survey and taxonomy of the fuzzy signature-based Intrusion Detection Systems. *Applied Soft Computing Journal*, 92, 106301.
- Mazini, M., Shirazi, B., & Mahdavi, I. (2019). Anomaly network-based intrusion detection system using a reliable hybrid artificial bee colony and AdaBoost algorithms. *Journal of King Saud University - Computer and Information Sciences*, 31, 541–553.
- McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5, 115–133.
- Moustafa, N., & Slay, J. (2015). UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In: *MilCIS, 2015 Military Communications and Information Systems Conference*, Canberra, Australia, 10-12 Nov 2015. Institute of Electrical and Electronics Engineers Inc.
- Mukkamala, S., Sung, A. H., & Abraham, A. (2005). Intrusion detection using an ensemble of intelligent paradigms. *Journal of Network and Computer Applications*, 28, 167–182.
- Neu, D. A., Lahann, J., & Fettke, P. (2022). A systematic literature review on state-of-the-art deep learning methods for process prediction. *Artificial Intelligence Review*, 55(2), 801–827.
- Nguyen, G. N., Viet, N. H. Le, Elhoseny, M., Shankar, K., Gupta, B. B., & El-Latif, A. A. (2021). Secure blockchain enabled Cyber–physical systems in healthcare using deep belief network with ResNet model. *Journal of Parallel and Distributed Computing*, 153, 150–160.
- Nguyen, T.-L., 2023. Deep ANFIS for Understanding Emotions in Movie Clips. In: ICERA

- , 2022 *International Conference on Engineering Research and Applications*, Thai Nguyen, Vietnam, 1-2 December 2022. Springer Science and Business Media Deutschland GmbH.
- Ozay, M., Esnaola, I., Yarman Vural, F. T., Kulkarni, S. R., & Poor, H. V. (2016). Machine Learning Methods for Attack Detection in the Smart Grid. *IEEE Transactions on Neural Networks and Learning Systems*, 27, 1773–1786.
- Pajouh, H. H., Javidan, R., Khayami, R., Dehghantanha, A., & Choo, K. K. R. (2019). A Two-Layer Dimension Reduction and Two-Tier Classification Model for Anomaly-Based Intrusion Detection in IoT Backbone Networks. *IEEE Transactions on Emerging Topics in Computing*, 7, 314–323.
- Pandeewari, N., & Kumar, G. (2016). Anomaly Detection System in Cloud Environment Using Fuzzy Clustering Based ANN. *Mobile Networks and Applications*, 21, 494–505.
- Piotrowski, A. P., Napiorkowski, J. J., & Piotrowska, A. E. (2020). Impact of deep learning-based dropout on shallow neural networks applied to stream temperature modelling. *Earth-Science Reviews*, 201, 103076.
- Pu, G., Wang, L., Shen, J., & Dong, F. (2021). A hybrid unsupervised clustering-based anomaly detection method. *Tsinghua Science and Technology*, 26, 146–153.
- Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, 1, 81–106.
- Quinlan, R. (1993). *San Diego: Morgan Kaufmann*, Dutch: Elsevier.
- Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. In: IEEE, 2016 *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, 27-30 June 2016. IEEE.
- Ren, S., He, K., Girshick, R., & Sun, J. (2017). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and*

- Machine Intelligence*, 39, 1137–1149.
- Saied, A., Overill, R. E., & Radzik, T. (2016). Detection of known and unknown DDoS attacks using Artificial Neural Networks. *Neurocomputing*, 172, 385–393.
- Sangkatsanee, P., Wattanapongsakorn, N., & Charnsripinyo, C. (2011). Practical real-time intrusion detection using machine learning approaches. *Computer Communications*, 34, 2227–2235.
- Saridou, B., Rose, J. R., Shiaeles, S., & Papadopoulos, B. (2022). SAGMAD—A Signature Agnostic Malware Detection System based on Binary Visualisation and Fuzzy Sets. *Electronics (Switzerland)*, 11, 1044.
- Savva, M., Ioannou, I., & Vassiliou, V. (2022). Fuzzy-Logic Based IDS for Detecting Jamming Attacks in Wireless Mesh IoT Networks. In: IEEE, 2022 20th Mediterranean Communication and Computer Networking Conference, Pafos, Cyprus, 01-03 June 2022. IEEE.
- Shiomoto, K. (2023). Network Intrusion Detection System Based on an Adversarial Auto-Encoder with Few Labeled Training Samples. *Journal of Network and Systems Management*, 31, 1–22.
- Shone, N., Ngoc, T. N., Phai, V. D., & Shi, Q. (2018). A Deep Learning Approach to Network Intrusion Detection. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2, 41–50.
- Siddique, N., & Adeli, H. (2013). Computational Intelligence, In *Computational Intelligence: Synergies of Fuzzy Logic, Neural Networks and Evolutionary Computing* Oxford, UK: John Wiley & Sons Ltd.
- Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In ICLR, 3rd International Conference on Learning

- Representations*, San Diego, California, US, 7-9 May 2015. ICLR.
- Singh, R., Kumar, H., & Singla, R. K. (2015). An intrusion detection system using network traffic profiling and online sequential extreme learning machine. *Expert Systems with Applications*, 42, 8609–8624.
- Sivatha Sindhu, S. S., Geetha, S., & Kannan, A. (2012). Decision tree based light weight intrusion detection using a wrapper approach. *Expert Systems with Applications*, 39, 129–141.
- Stallings, W., & Lawrie, B. (2017). *Computer Security: Principles and Practice*, 4th ed., London: Pearson Education.
- Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the Inception Architecture for Computer Vision. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2016-Decem*, 2818–2826.
- Tajbakhsh, A., Rahmati, M., & Mirzaei, A. (2009). Intrusion detection using fuzzy association rules. *Applied Soft Computing Journal*, 9, 462–469.
- Tang, T.A., Mhamdi, L., McLernon, D., Zaidi, S.A.R., & Ghogho, M. (2016). Deep learning approach for Network Intrusion Detection in Software Defined Networking. In: WINCOM, *2016 International Conference on Wireless Networks and Mobile Communications*, Fez, Morocco, October 2016. IEEE.
- Tang, T.A., Mhamdi, L., McLernon, D., Zaidi, S.A.R., & Ghogho, M. (2018). Deep Recurrent Neural Network for Intrusion Detection in SDN-based Networks. In: IEEE, *2018 4th IEEE Conference on Network Softwarization and Workshops*, Montreal, QC, Canada, 25-29 June 2018. IEEE.
- Tavallae, M., Bagheri, E., Lu, W., & Ghorbani, A.A. (2009). A detailed analysis of the

- KDD CUP 99 data set. In: CISDA, *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, Ottawa, ON, Canada, July 2009. IEEE.
- Thakkar, A., & Lohiya, R. (2023). Fusion of statistical importance for feature selection in Deep Neural Network-based Intrusion Detection System. *Information Fusion*, 90, 353–363.
- Tong, X., Wang, Z., & Yu, H. (2009). A research using hybrid RBF/Elman neural networks for intrusion detection system secure model. *Computer Physics Communications*, 180, 1795–1801.
- Toosi, A. N., & Kahani, M. (2007). A new approach to intrusion detection based on an evolutionary soft computing model using neuro-fuzzy classifiers. *Computer Communications*, 30, 2201–2212.
- Tsang, C. H., Kwong, S., & Wang, H. (2007). Genetic-fuzzy rule mining approach and evaluation of feature selection techniques for anomaly intrusion detection. *Pattern Recognition*, 40, 2373–2391.
- Wang, G., Hao, J., Mab, J., & Huang, L. (2010). A new approach to intrusion detection using Artificial Neural Networks and fuzzy clustering. *Expert Systems with Applications*, 37, 6225–6232.
- Wang, H., Wei, Q., & Xie, Y. (2022). A Novel Method for Network Intrusion Detection. *Scientific Programming*, 2022, 1–13.
- Wang, W., & Battiti, R. (2006). Identifying intrusions in computer networks with principal component analysis. In: ARES'06, *First International Conference on Availability, Reliability and Security*, Vienna, Austria, 20-22 April 2006. IEEE.
- Wei Wang, Ming Zhu, Xuewen Zeng, Xiaozhou Ye, & Yiqiang Sheng (2017). Malware

- traffic classification using convolutional neural network for representation learning. In: IEEE, *2017 International Conference on Information Networking (ICOIN)*, Da Nang, Vietnam, 11-13 January 2017. IEEE.
- Weng, W., & Zhu, X. (2021). INet: Convolutional Networks for Biomedical Image Segmentation. *IEEE Access*, 9, 16591–16603.
- Xiang, C., Yong, P. C., & Meng, L. S. (2008). Design of multiple-level hybrid classifier for intrusion detection system using Bayesian clustering and decision trees. *Pattern Recognition Letters*, 29, 918–924.
- Xiao, Yihan, Xing, C., Zhang, T., & Zhao, Z. (2019). An Intrusion Detection Model Based on Feature Reduction and Convolutional Neural Networks. *IEEE Access*, 7, 42210–42219.
- Xiao, Yuelei, & Xiao, X. (2019). An Intrusion Detection System Based on a Simplified Residual Network. *Information*, 10, 356.
- Xie, S., Girshick, R., Dollár, P., Tu, Z., & He, K. (2017). Aggregated residual transformations for deep neural networks. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, 2017-Janua*, 5987–5995.
- Xu, Y., de Souza, R. W. R., Medeiros, E. P., Jain, N., Zhang, L., Passos, L. A., & de Albuquerque, V. H. C. (2023). Intelligent IoT security monitoring based on fuzzy optimum-path forest classifier. *Soft Computing*, 27, 4279–4288.
- Xuren, W., Famei, H., & Rongsheng, X. (2006). Modeling Intrusion Detection System by Discovering Association Rule in Rough Set Theory Framework. In: *CIMCA'06 2006 International Conference on Computational Intelligence for Modelling Control and Automation and International Conference on Intelligent Agents Web Technologies and International Commerce*, Sydney, NSW, Australia, 28 November 2006 - 01 December

2006. IEEE.

- Yin, C., Zhu, Y., Fei, J., & He, X. (2017). A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks. *IEEE Access*, 5, 21954–21961.
- Zadeh, L. A. (1965). Fuzzy sets. *Information and Control*, 8, 338–353.
- Zaremba, W., Sutskever, I., & Vinyals, O. (2014). Recurrent Neural Network Regularization. *ICLR*, 2013, 1–8.
- Zhang, C., Jiang, J., & Kamel, M. (2005). Intrusion detection using hierarchical neural networks. *Pattern Recognition Letters*, 26, 779–791.
- Zhang, Y., Li, P., & Wang, X. (2019). Intrusion Detection for IoT Based on Improved Genetic Algorithm and Deep Belief Network. *IEEE Access*, 7, 31711–31722.
- Zhou, X., Liang, W., Li, W., Yan, K., Shimizu, S., & Wang, K. I. K. (2022). Hierarchical Adversarial Attacks Against Graph-Neural-Network-Based IoT Network Intrusion Detection System. *IEEE Internet of Things Journal*, 9, 9310–9319.

APPENDICES

Appendix A: Journal Publications

1. **Liu, J.**, Yinchai, W., Siong, T.C., Li, X., Zhao, L., Wei, F. (2022). On the combination of adaptive neuro-fuzzy inference system and deep residual network for improving detection rates on intrusion detection. *PLoS ONE*, 17(12): e0278819. <https://doi.org/10.1371/journal.pone.0278819>.
2. **Liu, J.**, Yinchai, W., Siong, T.C. et al (2022). A hybrid interpretable deep structure based on adaptive neuro-fuzzy inference system, decision tree, and K-means for intrusion detection. *Sci Rep*, 12, 20770. <https://doi.org/10.1038/s41598-022-23765-x>.
3. **Liu, J.**, Yinchai, W., Wei, F. et al (2023). Secure Cloud-Aided Approximate Nearest Neighbor Search on High-Dimensional Data. *IEEE Access*, 11, 109027–109037. <https://doi.org/10.1109/ACCESS.2023.3321457>

Appendix B: The Generated Rules with Four Attributes' Intervals of Modified ANFIS in The First Proposed Method

rule_num	duration	count	dst_host_count	dst_host_srv_count
rule0	0	0	0	0
rule1	0	0	0	1
rule2	0	0	0	2
rule3	0	0	0	3
rule4	0	0	1	0
rule5	0	0	1	1
rule6	0	0	1	2
rule7	0	0	1	3
rule8	0	0	2	0
rule9	0	0	2	1
rule10	0	0	2	2
rule11	0	0	2	3
rule12	0	1	0	0
rule13	0	1	0	1
rule14	0	1	0	2
rule15	0	1	0	3
rule16	0	1	1	0
rule17	0	1	1	1
rule18	0	1	1	2
rule19	0	1	1	3
rule20	0	1	2	0
rule21	0	1	2	1

rule22	0	1	2	2
rule23	0	1	2	3
rule24	0	2	0	0
rule25	0	2	0	1
rule26	0	2	0	2
rule27	0	2	0	3
rule28	0	2	1	0
rule29	0	2	1	1
rule30	0	2	1	2
rule31	0	2	1	3
rule32	0	2	2	0
rule33	0	2	2	1
rule34	0	2	2	2
rule35	0	2	2	3
rule36	0	3	0	0
rule37	0	3	0	1
rule38	0	3	0	2
rule39	0	3	0	3
rule40	0	3	1	0
rule41	0	3	1	1
rule42	0	3	1	2
rule43	0	3	1	3
rule44	0	3	2	0
rule45	0	3	2	1

rule46	0	3	2	2
rule47	0	3	2	3
rule48	1	0	0	0
rule49	1	0	0	1
rule50	1	0	0	2
rule51	1	0	0	3
rule52	1	0	1	0
rule53	1	0	1	1
rule54	1	0	1	2
rule55	1	0	1	3
rule56	1	0	2	0
rule57	1	0	2	1
rule58	1	0	2	2
rule59	1	0	2	3
rule60	1	1	0	0
rule61	1	1	0	1
rule62	1	1	0	2
rule63	1	1	0	3
rule64	1	1	1	0
rule65	1	1	1	1
rule66	1	1	1	2
rule67	1	1	1	3
rule68	1	1	2	0
rule69	1	1	2	1

rule70	1	1	2	2
rule71	1	1	2	3
rule72	1	2	0	0
rule73	1	2	0	1
rule74	1	2	0	2
rule75	1	2	0	3
rule76	1	2	1	0
rule77	1	2	1	1
rule78	1	2	1	2
rule79	1	2	1	3
rule80	1	2	2	0
rule81	1	2	2	1
rule82	1	2	2	2
rule83	1	2	2	3
rule84	1	3	0	0
rule85	1	3	0	1
rule86	1	3	0	2
rule87	1	3	0	3
rule88	1	3	1	0
rule89	1	3	1	1
rule90	1	3	1	2
rule91	1	3	1	3
rule92	1	3	2	0
rule93	1	3	2	1

rule94	1	3	2	2
rule95	1	3	2	3
rule96	2	0	0	0
rule97	2	0	0	1
rule98	2	0	0	2
rule99	2	0	0	3
rule100	2	0	1	0
rule101	2	0	1	1
rule102	2	0	1	2
rule103	2	0	1	3
rule104	2	0	2	0
rule105	2	0	2	1
rule106	2	0	2	2
rule107	2	0	2	3
rule108	2	1	0	0
rule109	2	1	0	1
rule110	2	1	0	2
rule111	2	1	0	3
rule112	2	1	1	0
rule113	2	1	1	1
rule114	2	1	1	2
rule115	2	1	1	3
rule116	2	1	2	0
rule117	2	1	2	1

rule118	2	1	2	2
rule119	2	1	2	3
rule120	2	2	0	0
rule121	2	2	0	1
rule122	2	2	0	2
rule123	2	2	0	3
rule124	2	2	1	0
rule125	2	2	1	1
rule126	2	2	1	2
rule127	2	2	1	3
rule128	2	2	2	0
rule129	2	2	2	1
rule130	2	2	2	2
rule131	2	2	2	3
rule132	2	3	0	0
rule133	2	3	0	1
rule134	2	3	0	2
rule135	2	3	0	3
rule136	2	3	1	0
rule137	2	3	1	1
rule138	2	3	1	2
rule139	2	3	1	3
rule140	2	3	2	0
rule141	2	3	2	1

rule142	2	3	2	2
rule143	2	3	2	3
rule144	3	0	0	0
rule145	3	0	0	1
rule146	3	0	0	2
rule147	3	0	0	3
rule148	3	0	1	0
rule149	3	0	1	1
rule150	3	0	1	2
rule151	3	0	1	3
rule152	3	0	2	0
rule153	3	0	2	1
rule154	3	0	2	2
rule155	3	0	2	3
rule156	3	1	0	0
rule157	3	1	0	1
rule158	3	1	0	2
rule159	3	1	0	3
rule160	3	1	1	0
rule161	3	1	1	1
rule162	3	1	1	2
rule163	3	1	1	3
rule164	3	1	2	0
rule165	3	1	2	1

rule166	3	1	2	2
rule167	3	1	2	3
rule168	3	2	0	0
rule169	3	2	0	1
rule170	3	2	0	2
rule171	3	2	0	3
rule172	3	2	1	0
rule173	3	2	1	1
rule174	3	2	1	2
rule175	3	2	1	3
rule176	3	2	2	0
rule177	3	2	2	1
rule178	3	2	2	2
rule179	3	2	2	3
rule180	3	3	0	0
rule181	3	3	0	1
rule182	3	3	0	2
rule183	3	3	0	3
rule184	3	3	1	0
rule185	3	3	1	1
rule186	3	3	1	2
rule187	3	3	1	3
rule188	3	3	2	0
rule189	3	3	2	1

rule190	3	3	2	2
rule191	3	3	2	3

0-3 represent the interval of attribute

Appendix C: The Generated Rules with Four Attributes' Weights of Modified ANFIS in The First Proposed Method

rule number	duration	count	dst_host_count	dst_host_srv_count
rule0	-0.001081373	0.001768952	-0.054062329	0.001218294
rule1	0.139038563	0.077832215	-0.200930342	0.799063981
rule2	0.132066861	-0.285668701	-1.924451113	0.634215117
rule3	0.160386488	-0.529788375	-2.695405245	-0.505216897
rule4	-0.033822022	0.799601138	0.955424905	-0.043199532
rule5	-1.860505342	4.401194572	-2.359276533	-0.784082592
rule6	-0.198397279	2.083471298	2.683088303	-1.102190018
rule7	0.214376464	-0.311915219	-2.610447645	-1.874673605
rule8	-0.216854125	-0.372936517	-12.3175993	-0.938369393
rule9	-0.590529144	1.327121377	-7.090093613	-0.196419567
rule10	-0.098307095	1.08185935	2.751200676	0.373473167
rule11	-0.00114681	0.528308868	0.015081241	-0.788546562
rule12	-0.001322923	0.047712218	0.03107987	-0.006127751
rule13	0.191806406	0.246356755	-0.448630273	0.012798227
rule14	0.181211203	-0.132366851	-2.217318773	-0.781094968
rule15	0.212443754	-0.390309364	-3.019026041	-1.079549193
rule16	-0.027433943	-2.640571356	-7.8497715	-0.484213769
rule17	-1.548238754	1.164277434	-11.49091148	-1.900151134
rule18	-0.12165805	2.259680986	2.23331809	-1.730063081
rule19	0.270934731	-0.12045256	-2.936773539	-2.546766281
rule20	-0.154171333	-13.35506916	-22.51743507	-1.991807699
rule21	-0.460084617	-18.02322006	-32.57452774	0.634644508

rule22	-0.067587785	1.483944416	3.617157936	0.663432539
rule23	0.025569208	0.921133816	1.24286294	0.0695903
rule24	0.001383552	0.001983907	0.000504626	-0.002595596
rule25	0.017937519	0.425003797	1.896654129	1.247033834
rule26	-0.008393357	0.082253821	0.375665635	0.423784912
rule27	0.004580019	0.229592174	0.184964687	0.550007284
rule28	-0.016498374	-2.355172396	-6.56633234	-0.358063519
rule29	-0.142536968	2.193591833	4.833705425	1.25861001
rule30	-0.011521723	1.220077634	3.852017641	0.185011312
rule31	0.05202603	0.425407559	1.301709056	0.359811991
rule32	-0.120529056	42.22142029	74.76231384	2.662210464
rule33	-0.236752048	-19.1728344	-54.53039551	-3.255004644
rule34	-0.074017659	0.818728805	2.621268749	0.377600431
rule35	-0.061304629	0.606279135	0.895581007	-0.582393706
rule36	-1.43E-12	5.29E-10	3.20E-10	3.54E-10
rule37	0.22649318	0.145881921	1.358865499	-0.239084437
rule38	0.19332014	-0.258645356	-0.410698742	-1.16710794
rule39	0.214045718	0.059079777	-0.383914173	-1.07366538
rule40	3.32E-09	0.010082212	0.007777808	0.002924394
rule41	0.284090012	0.27335152	3.002543688	-0.196016133
rule42	0.247772858	-0.166586131	1.006505728	-1.313096523
rule43	0.268794537	0.179216802	1.054638505	-1.182042122
rule44	3.35E-09	0.002126397	0.00232107	0.000195794
rule45	0.023408396	0.788968742	3.75501132	1.071406841

rule46	0.007694576	0.463404953	2.705916882	0.450632095
rule47	0.011284348	0.837607026	2.844821692	0.688829958
rule48	-0.004519064	0.001422714	-0.009442911	-0.003010672
rule49	0.250881672	0.043879054	-0.378223062	1.1298666
rule50	0.245057687	-0.479167342	-2.898801565	-0.233154982
rule51	0.28939873	-0.881568313	-4.05624485	-0.596299052
rule52	-0.039527472	1.102188468	1.394570112	-0.045735218
rule53	-2.523619413	5.876322269	-4.082207203	-1.113071918
rule54	-0.233344123	2.893269539	3.358667374	-1.705601811
rule55	0.365388602	-0.571603179	-3.91358304	-2.762370825
rule56	-0.096449003	-0.389258802	-11.86398697	-1.048708439
rule57	-0.648975015	1.84504509	-9.428413391	-0.262226313
rule58	-0.122723676	1.526170611	3.675691128	0.473174334
rule59	0.019227743	0.716616571	0.433105946	-0.700147212
rule60	-0.004726472	0.01691396	-0.002994359	-0.004693782
rule61	0.327764302	0.25034219	-0.855771124	0.087053426
rule62	0.316109389	-0.302399963	-3.430533409	-1.188510656
rule63	0.364781171	-0.722803831	-4.575351238	-1.583468676
rule64	-0.032212023	-3.443399429	-10.25354004	-0.63297838
rule65	-2.088859797	1.696448684	-16.20996094	-2.723174572
rule66	-0.122992493	3.100889921	2.821798563	-2.524126053
rule67	0.447178274	-0.339542449	-4.412407398	-3.696424007
rule68	-0.119420961	-30.40664101	-70.93028259	-3.970664024
rule69	-0.494559944	-31.49012566	-58.61354065	0.432494402

rule70	-0.078758582	1.9283216	4.559586525	0.620995343
rule71	0.057522438	1.396550536	2.190544605	0.559102178
rule72	0.001868544	0.002623487	0.000763689	-0.003362291
rule73	0.071997277	0.561711729	2.681076288	1.727264643
rule74	0.033724435	0.07018654	0.485408992	0.538267136
rule75	0.055048671	0.245709956	0.230158791	0.700734556
rule76	-0.020229546	-3.153582335	-8.750577927	-0.476223409
rule77	-0.148974016	3.224707127	7.351079941	1.864966154
rule78	0.02647179	1.698243618	5.471745491	0.290784985
rule79	0.12106391	0.516711295	1.843843341	0.517369628
rule80	-0.151729286	-27.01334381	-69.70384979	-3.539674282
rule81	-0.29708153	-35.90541458	-91.80428314	-5.370013237
rule82	-0.091568582	1.162162662	3.780166626	0.605331957
rule83	-0.071819089	0.854179382	1.447196364	-0.656881988
rule84	-1.95E-12	7.01E-10	4.25E-10	4.70E-10
rule85	0.387769043	0.166923299	1.790740252	-0.59646982
rule86	0.339234829	-0.41465345	-0.768332541	-1.938379526
rule87	0.372258693	0.00825736	-0.711018741	-1.829311728
rule88	4.57E-09	0.013418086	0.010364574	0.003880391
rule89	0.468460143	0.352972358	4.189837933	-0.413980484
rule90	0.415261477	-0.278165549	1.304824233	-2.026432991
rule91	0.448907703	0.178744435	1.396084189	-1.867810249
rule92	4.47E-09	0.03839048	0.038322512	0.00361887
rule93	0.055013657	1.54572022	5.772963524	1.755168796

rule94	0.032006182	0.857899666	4.040636063	0.764200628
rule95	0.038780957	1.343895316	4.23221302	1.10833478
rule96	-0.004151078	-2.14E-05	-9.54E-05	-8.71E-05
rule97	-0.002720724	-0.448085606	-1.988174677	0.298294693
rule98	-0.041786838	-0.963900745	-4.398100853	-1.034447908
rule99	-0.00346219	-1.126066446	-4.983433247	-0.963133991
rule100	0.002068437	1.48E-05	0.007390857	2.78E-05
rule101	-0.691599369	1.277299643	3.771636248	1.371480465
rule102	-0.323258072	0.753013194	2.017140865	-0.15969345
rule103	0.024319593	-0.802916229	-2.066513777	-0.104076743
rule104	0.12830852	0.000990826	0.261510909	0.002312399
rule105	-0.298324764	1.093824863	4.204276085	1.41082406
rule106	-0.280432194	0.828949153	3.078096151	0.704080045
rule107	-0.19813332	0.553813577	1.837641597	0.824058235
rule108	-0.003877367	-2.00E-05	-8.83E-05	-8.14E-05
rule109	0.064386941	-0.278280854	-2.443759918	-0.480077893
rule110	0.023635656	-0.805821121	-4.918300152	-1.847494125
rule111	0.066248633	-0.977693081	-5.484055996	-1.768696189
rule112	0.001592415	9.61E-06	0.005804855	2.08E-05
rule113	-0.545165539	1.557469606	3.666261911	0.555639088
rule114	-0.236640856	1.022432208	1.739607215	-1.01372695
rule115	0.099953808	-0.605848789	-2.536178827	-0.962924957
rule116	0.021737773	0.000154318	0.043381479	0.00032846
rule117	-0.257518888	1.346171737	4.284375668	1.04075253

rule118	-0.242931396	1.077871203	3.128784418	0.318505555
rule119	-0.16267693	0.785520494	1.842778683	0.439055264
rule120	3.59E-07	7.44E-09	5.09E-08	4.02E-07
rule121	-0.199718088	0.205600634	1.4254179	1.142837644
rule122	-0.238094524	-0.254577249	-0.620090723	0.047523368
rule123	-0.220653951	-0.025890736	-0.712036729	0.22312066
rule124	0.000409024	1.54E-06	0.001442077	5.40E-06
rule125	-0.257567465	0.790305436	4.124883175	0.921872616
rule126	-0.262752473	0.285182744	1.919414639	-0.357197881
rule127	-0.202736869	0.080827564	0.538424194	-0.199167341
rule128	4.88E-05	1.14E-05	0.000195872	7.54E-07
rule129	-0.281838	0.939811409	3.913063765	1.113680959
rule130	-0.294478238	0.689358354	2.847950459	0.507178366
rule131	-0.284246773	0.817768276	2.439880371	0.659427166
rule132	-2.34E-12	2.60E-11	-3.22E-11	1.55E-11
rule133	0.080208391	-0.112983108	0.640508294	-1.061910629
rule134	0.032423679	-0.659290731	-1.762480736	-2.323480129
rule135	0.059984226	-0.261609197	-1.721788287	-2.219854832
rule136	-2.55E-12	5.44E-09	4.14E-09	1.58E-09
rule137	0.109747104	0.0300116	2.788052559	-0.945368052
rule138	0.057223402	-0.590050817	0.054780204	-2.42775631
rule139	0.084819868	-0.132582933	0.142871946	-2.289439201
rule140	-1.66E-12	1.52E-08	1.53E-08	1.44E-09
rule141	-0.174044609	0.738465846	4.232902527	0.987543821

rule142	-0.197635144	0.43904826	2.956787348	0.28764087
rule143	-0.19387579	0.775994062	3.00496316	0.439241767
rule144	-0.002012988	-5.18E-06	-0.000498161	-0.000134919
rule145	0.332277179	1.528301477	10.3962822	5.348337173
rule146	0.291928738	0.977288127	7.894379139	3.957846642
rule147	0.32136488	1.448859096	8.438435555	4.553383827
rule148	-0.062932283	-0.00028249	-0.07204517	-0.000585873
rule149	-0.787251651	1.238612413	8.908841133	4.177410126
rule150	0.078895994	0.621285737	7.10497427	2.595172167
rule151	0.386761248	1.155265689	7.933054447	3.18535471
rule152	0.072855644	0.000243537	0.062223762	0.000488029
rule153	-0.519348264	1.116298914	5.376068592	2.247624874
rule154	-0.042074516	0.821151733	4.656406879	1.519568682
rule155	0.032181624	1.179224133	4.906630516	1.818805099
rule156	-0.001712054	-4.40E-06	-0.000423688	-0.000114749
rule157	0.406323433	1.86816299	10.72436523	4.865225315
rule158	0.36438635	1.305169344	8.157360077	3.43857193
rule159	0.396506995	1.786006451	8.742208481	4.058995247
rule160	-0.051362179	-0.0002305	-0.058781255	-0.000478731
rule161	-0.558796346	1.599603772	9.312674522	3.58443594
rule162	0.172738671	0.968842208	7.293314457	1.962142467
rule163	0.467675239	1.512906432	8.144284248	2.575136423
rule164	-0.234258235	-0.001076021	-0.274922997	-0.00215626
rule165	-0.396524787	1.404635549	5.699707985	1.937833428

rule166	-0.001937766	1.105009675	4.858265877	1.19314456
rule167	0.069842286	1.470051885	5.124083996	1.501168013
rule168	-1.86E-12	2.67E-11	-3.31E-11	1.53E-11
rule169	0.062985294	0.750344336	4.695841789	2.388947248
rule170	0.026967481	0.270852387	2.591834545	1.260462642
rule171	0.04048967	0.715738356	2.805474997	1.588949561
rule172	-0.02435345	-0.000109517	-0.027981641	-0.000219464
rule173	-0.019379646	0.621891141	4.910216808	1.360837579
rule174	0.050177414	0.076765157	2.640815258	0.045022875
rule175	0.102403983	0.58518213	2.890043497	0.383351952
rule176	-0.146197721	-0.000657474	-0.167984858	-0.001317527
rule177	-0.25597772	0.91456902	3.982939959	1.213005424
rule178	-0.078414552	0.648569882	3.077502251	0.58798033
rule179	-0.068998419	0.998229861	3.146682501	0.80227977
rule180	-1.91E-12	2.77E-11	-3.46E-11	1.65E-11
rule181	0.293499172	-0.821864784	-3.05713439	-3.008848667
rule182	0.248123869	-1.377656698	-5.477371693	-4.271422863
rule183	0.27527082	-0.953180373	-5.584542274	-4.221579075
rule184	-2.04E-12	3.19E-11	-3.92E-11	1.82E-11
rule185	0.373742163	-0.745830357	-1.031740308	-3.08911109
rule186	0.323927075	-1.379240513	-3.796514273	-4.580729961
rule187	0.351587474	-0.877773046	-3.846138239	-4.490730762
rule188	-1.29E-12	2.26E-11	-2.58E-11	1.29E-11
rule189	0.03230501	0.496430099	2.980915308	0.247499153

rule190	0.010782043	0.188266158	1.678574681	-0.464001805
rule191	0.015415906	0.548541963	1.671357512	-0.324017853

Appendix D: The Generated Rules with Four Attributes' Weights and r of Modified ANFIS in The Second Proposed Method

Rule Number	duration	count	dst_host_count	dst_host_srv_count	r
rule0	0.912633	0.038782	0.382223	0.188299	0.246868
rule1	0.793117	0.387096	0.576429	0.048162	0.698428
rule2	0.324191	0.604768	0.496349	0.227386	0.157725
rule3	0.2888	0.002117	0.045971	0.40616	0.429863
rule4	-0.17317	0.474115	0.173647	0.405452	0.196219
rule5	0.992505	0.924066	0.558316	0.100145	0.120558
rule6	-0.10636	0.439569	0.128769	0.715641	0.467034
rule7	0.544545	0.61816	0.716447	0.048606	0.92977
rule8	-0.46834	0.692778	0.723793	0.561677	0.796844
rule9	0.39182	0.44647	0.795715	0.733722	0.600022
rule10	0.023456	0.022299	0.118969	0.748522	0.753872
rule11	0.766467	0.596883	0.069285	0.094784	0.123723
rule12	0.844623	0.314823	0.709759	0.727116	0.61416
rule13	0.326573	0.558993	0.743943	0.866395	0.100346
rule14	0.940911	0.744211	0.446172	0.212663	0.180843
rule15	0.903323	0.483978	0.457087	0.174246	0.384277
rule16	0.344584	1.457618	0.720129	1.056152	0.971608
rule17	0.386902	0.608888	0.465949	0.720046	0.455176
rule18	-0.01638	0.180129	0.446932	0.711239	0.882801
rule19	0.286007	0.900344	0.809844	0.465056	0.335725
rule20	0.132107	0.310317	0.257752	0.945694	0.764025

rule21	0.990796	0.337944	0.953611	0.740688	0.192604
rule22	0.81035	0.931257	-0.00323	0.488747	0.276794
rule23	0.269117	0.249037	0.799538	0.094236	0.044439
rule24	0.25111	0.863521	0.877058	1.125156	1.137547
rule25	0.782634	0.474079	0.593925	0.607624	0.419111
rule26	0.857426	0.384196	0.335954	0.914791	0.184694
rule27	0.270526	0.719492	0.526085	0.125003	0.222262
rule28	0.900054	0.430908	0.656443	0.297443	1.1813
rule29	0.592964	0.393583	0.202679	0.275164	0.24013
rule30	0.320082	0.683497	0.496568	0.54302	0.680666
rule31	0.068587	0.104896	0.587402	0.160419	0.736977
rule32	0.958706	0.397265	0.144433	0.640488	0.616761
rule33	0.399153	0.611011	0.596781	0.490967	0.347192
rule34	0.139596	0.550007	0.831538	0.120628	0.742352
rule35	0.832145	0.661792	0.142331	0.627675	0.82275
rule36	0.82718	0.392866	0.267327	0.084517	0.507285
rule37	0.475409	0.575311	0.395607	0.281335	0.866238
rule38	0.468848	0.89154	0.026081	0.656689	-0.00754
rule39	0.685748	0.627479	0.416181	0.392638	0.415244
rule40	0.998556	0.827017	0.021562	0.706549	0.592641
rule41	0.956073	0.55833	0.918837	0.442208	0.662319
rule42	0.210139	0.429479	0.650655	-0.01814	0.643103
rule43	0.326526	0.368585	1.151617	0.534994	0.283283
rule44	0.17348	0.651012	0.545738	0.293967	0.888717

rule45	0.060504	0.742123	0.856654	0.345624	0.452401
rule46	0.739284	0.757098	0.399883	0.617105	0.079279
rule47	0.868321	0.29095	0.401631	0.130335	0.754388
rule48	0.692928	0.558599	0.781789	0.310089	0.667824
rule49	0.195698	0.357847	0.281991	0.790912	0.358728
rule50	0.145607	0.209672	0.963761	0.503654	0.247264
rule51	0.553083	0.765271	1.383853	0.295989	0.570781
rule52	0.191623	0.265268	0.300779	0.782875	0.671055
rule53	0.727228	0.177754	0.545358	0.720203	0.453408
rule54	0.722965	0.740491	0.41757	0.85167	0.69817
rule55	0.339832	0.422281	0.590118	0.4486	0.430758
rule56	0.614026	0.375941	0.965343	0.774231	0.21888
rule57	1.030698	0.256162	0.22633	0.895898	0.346057
rule58	0.786015	0.584065	0.105913	0.610659	0.235599
rule59	-0.05847	0.464951	0.63196	0.675635	0.906144
rule60	0.627862	0.399215	0.648889	0.351635	0.540221
rule61	0.546492	0.343717	0.588447	0.335539	0.184862
rule62	0.140986	0.815063	1.001376	0.422628	0.507533
rule63	0.439443	0.561035	1.115826	1.422124	0.715875
rule64	0.791725	0.274637	0.129505	0.751237	0.074114
rule65	0.653247	0.727141	0.512626	-0.06352	0.645469
rule66	0.459925	0.489808	0.49611	0.530405	0.476276
rule67	0.550534	1.149042	1.013977	1.418157	0.779472
rule68	0.847488	0.605599	0.430833	0.091578	0.850205

rule69	0.330978	0.230283	0.062311	0.777564	0.621303
rule70	0.451162	0.198432	0.539647	0.541165	-0.12088
rule71	0.593344	0.730746	0.348851	0.546106	0.408322
rule72	0.84485	0.172377	0.147415	0.789611	0.57423
rule73	0.877402	0.503185	0.265089	0.336141	0.564005
rule74	0.627594	0.060641	0.631104	0.569015	0.159324
rule75	0.318285	0.41702	0.188661	0.818802	0.045192
rule76	0.393858	0.450039	0.636952	0.385031	0.676705
rule77	0.477849	0.819584	0.491417	0.782125	0.618623
rule78	0.659454	0.292013	0.520912	0.989731	0.561013
rule79	0.051056	0.698918	0.251113	0.862162	1.049352
rule80	0.265772	0.817804	1.126143	1.510767	0.39945
rule81	0.484369	0.589235	-0.01979	0.421923	0.481006
rule82	0.318743	0.533833	0.452581	0.382472	0.238906
rule83	0.209871	0.214848	0.357171	0.853949	0.968735
rule84	0.049784	0.643548	0.752984	0.079718	0.843925
rule85	0.804818	1.018202	1.116343	0.585655	0.961342
rule86	0.135254	0.722275	0.750937	0.406545	0.464431
rule87	0.208558	0.425764	0.925276	0.820433	0.11912
rule88	0.516788	0.94207	1.04098	1.199384	0.539884
rule89	0.542736	0.537528	1.180152	0.828374	0.932607
rule90	0.671314	0.57764	1.005934	0.859143	0.907157
rule91	0.108745	0.726079	1.022413	0.155332	0.545775
rule92	0.734081	0.330248	0.523392	0.578456	0.479727

rule93	0.627273	0.066513	0.764483	0.018676	0.598536
rule94	0.816682	0.557913	0.926241	1.04599	0.491827
rule95	0.309113	0.325803	0.362186	0.419725	0.319224
rule96	0.465846	0.23077	0.392706	0.30574	0.604993
rule97	0.019184	0.329684	0.225219	0.051398	0.574166
rule98	0.70926	0.745275	0.120237	0.576066	0.446271
rule99	0.02769	0.729773	0.3296	1.055155	0.250526
rule100	-0.00592	0.722066	0.533514	0.054897	0.180086
rule101	0.61299	0.276181	0.432151	0.964365	0.61087
rule102	0.393978	0.15914	0.100855	-0.07083	0.844787
rule103	0.150174	0.603862	0.606104	0.730596	1.045714
rule104	0.917293	0.811642	0.206404	0.841854	-0.04239
rule105	0.23823	0.081645	0.466867	0.702217	0.762494
rule106	0.896152	0.897471	0.388808	0.884312	1.044085
rule107	0.697167	0.285053	0.406283	0.58152	0.683676
rule108	0.213734	0.802264	0.43455	0.355325	0.477339
rule109	0.483282	0.211509	0.379811	0.575951	0.948939
rule110	0.561145	0.812944	0.507936	0.077595	0.975243
rule111	0.028421	0.800632	0.301258	0.795896	0.522683
rule112	0.292436	0.504283	0.143487	0.931559	1.018348
rule113	0.97429	0.840139	0.151718	0.811623	0.289693
rule114	0.104329	0.944557	0.2063	0.315505	0.546477
rule115	0.63091	0.495399	0.910687	0.847731	0.667007
rule116	0.222907	0.947613	0.051837	0.513896	0.40646

rule117	1.003579	0.988965	0.207782	1.046368	0.851803
rule118	0.219336	0.005138	0.058818	0.611554	0.354736
rule119	0.34387	0.575318	0.434394	0.847056	0.397105
rule120	0.407224	0.803403	0.096226	0.413767	0.54815
rule121	0.70775	0.983243	0.834835	0.305208	0.558642
rule122	0.875175	0.373821	0.654578	0.833404	0.500715
rule123	0.992998	0.771465	0.439894	0.140971	0.07685
rule124	0.142845	0.148552	0.141418	0.603993	0.882034
rule125	0.922651	0.80176	0.827601	0.981265	0.443057
rule126	0.099436	0.43608	1.019762	0.992915	0.280691
rule127	0.917546	0.184614	0.406586	0.471007	0.455267
rule128	0.757699	0.776775	0.515081	0.578087	0.234517
rule129	0.047954	0.60107	0.564271	0.492066	0.335002
rule130	0.639817	0.769837	0.044666	0.962139	0.413103
rule131	0.665849	0.510502	0.586085	0.728187	0.261712
rule132	0.518435	0.771178	0.325487	0.262511	0.407755
rule133	0.199087	0.243063	0.296454	0.060987	0.572213
rule134	0.253362	0.684186	0.048293	0.519434	0.030942
rule135	0.722377	0.391324	0.871248	0.169676	0.406744
rule136	0.740587	0.520255	0.139598	0.855376	0.389171
rule137	0.843238	0.988904	0.256578	0.277925	0.866742
rule138	0.460817	0.951896	0.573259	0.628497	0.023905
rule139	0.063265	0.202415	0.512907	0.610438	0.992849
rule140	0.129077	0.350881	0.62042	0.68109	0.82064

rule141	0.424064	0.390391	0.357499	0.491212	0.704056
rule142	0.003016	0.29208	0.040589	0.159815	0.44147
rule143	0.37117	0.9473	0.557161	0.716039	0.900072
rule144	0.635305	0.778237	0.849675	0.687785	0.206005
rule145	0.657602	0.336005	0.568323	0.478858	0.099595
rule146	0.865575	0.964117	0.301855	0.804728	0.647579
rule147	0.877329	0.444876	0.577131	0.833787	0.395462
rule148	0.863434	0.527854	0.290147	0.328022	1.078513
rule149	0.060178	0.728686	0.433266	0.635906	0.598342
rule150	0.432021	0.854879	0.70218	0.469783	0.97849
rule151	0.775644	0.898081	0.280301	0.751201	0.497864
rule152	0.683358	0.45353	0.749698	0.580363	0.737391
rule153	0.548196	0.557985	0.808054	0.986616	0.564081
rule154	0.546934	0.588664	0.944723	0.067819	0.145811
rule155	0.370292	0.381728	0.709366	0.8076	0.859828
rule156	0.058802	0.559329	0.469946	0.492679	-0.05949
rule157	0.797422	0.149581	0.219101	0.889443	0.265188
rule158	0.517654	0.350691	0.757572	0.065115	0.372651
rule159	0.497563	0.411005	0.802262	0.856048	0.257874
rule160	0.709665	0.658796	0.213929	0.50281	0.13291
rule161	0.017588	0.25296	0.271126	0.39967	-0.08666
rule162	0.738399	0.724209	0.379092	0.587959	0.137481
rule163	0.411407	0.504483	0.984355	0.054932	0.098076
rule164	0.995091	0.686217	0.310613	0.887946	0.992194

rule165	0.267061	0.952425	0.716349	0.516644	0.474661
rule166	0.801805	0.42817	0.258226	0.767611	0.702713
rule167	0.729098	0.020312	0.146317	0.789705	0.477274
rule168	0.15322	0.89065	0.567911	0.329944	0.388958
rule169	0.197699	0.539256	0.071556	0.09564	0.185483
rule170	0.735088	0.491843	0.660396	0.488934	0.02076
rule171	0.488136	0.453889	0.085279	0.696769	0.468574
rule172	0.599427	0.719189	0.789491	0.040867	0.191347
rule173	0.832365	0.803908	0.551915	0.934337	0.81993
rule174	0.575267	0.131369	0.013446	0.354325	0.817822
rule175	0.3196	0.070525	0.027346	0.535203	0.231197
rule176	0.354807	0.054105	0.074034	0.304565	0.72267
rule177	0.921	0.951704	0.055893	0.437245	0.684778
rule178	0.435546	0.266007	0.007266	0.122337	0.838051
rule179	0.359876	0.084389	0.652676	0.111837	0.083516
rule180	0.574667	0.968465	0.988367	0.359502	0.688754
rule181	0.369674	0.934281	0.892211	0.640171	0.918309
rule182	0.912216	0.863562	0.020988	0.092147	0.160088
rule183	0.644438	0.316598	0.505288	0.569678	0.566854
rule184	0.867501	0.142921	0.818562	0.033571	0.812078
rule185	0.000642	0.275571	0.265659	0.982813	0.901653
rule186	0.187034	0.087334	0.157935	0.221658	0.267184
rule187	0.61672	0.103255	0.289266	0.371908	0.507444
rule188	0.200001	0.072509	0.902741	0.689803	0.828619

rule189	0.954574	0.718731	0.870528	0.37079	0.396648
rule190	0.42777	0.24291	0.768513	0.449121	0.206469
rule191	0.990945	0.942687	0.963984	0.91202	0.287073

Appendix E: Correlation Matrix of 43 Attributes

