**Exploration of Machine Learning Approaches for**

**Intelligent Network Intrusion Detection System**

**CHEN Xi**

**Introduction**

***Keywords***: Network Intrusion Detection System, Machine Learning.

In modern society, information and communication technology systems and networks has become one of the most important parts of the human life. The risk of cyber security also continues to rise as the networks becomes more popular, which enlarges the attack surface for malicious attempts. The landscape of cyber threats encompasses both manual and machine-generated attacks that are increasingly sophisticated, diverse, and adept at obfuscation, often leading to undetected breaches. For example, the loss of some enterprises, like Yahoo and Bitcoin, caused from sensitive data breach was over $100M [1].

These challenges pose an urgent need for powreful and efficient network intrusion detection systems (NIDS) [2]. An NIDS is a proactive tool that can detect and identify invasive activity, such as attacks, violations of the security policies, and so on. In recent decades, machine learning methods have been used and designed as intelligent NIDS to detect malicious intrusion, which can learn from passing attacks to conclude some attacking patterns.

Different machine learning methods have been discussed by researchers, which mainly includes Support Vector Machine (SVM) based systems and deep learning (CNN, DNN) based approaches [3]. In 2013, Jayshree developed a SVM-based NIDS incoperating Gain Ratio (IGR) and K-means algorithm [4]. After 2016, the most high-cited papers are typically deep learning based NIDS. Nathan Shone proposed onsymmetric deep autoencoder (NDAE) for unsupervised feature learning and deep learning classification model constructed using stacked NDAEs [5]. Deep neural network (DNN), a type of deep learning model, is also used to develop a flexible and effective IDS to detect and classify unforeseen and unpredictable cyberattacks [6]. Since NIDS is real-time system, it is natural to develop sequential algorithms like Long Short-Term Memory (LSTM) and Recurrent Neural Network (RNN). For example, researchers (Muhammad Ashfaq Khan and Yangwoo Kim) utilized LSTM to detect temporal features and an Auto Encoder (AE) to more effciently detect global features [7].

Machine learning methods require large amounts of data for training. Also, intelligent intrusion detection systems can only be built if there is availability of an effective data set. Among the datasets used for NIDS research, ‘KDD Cup 99’ and ‘NSL-KDD’ datasets [8] are most typical datasets to evaluate the proposed methods. ‘NSL-KDD’ is the improved version of ‘KDD Cup 99’, and so the expriments in thie report are all conducted on ‘KDD Cup 99’

These novel research works trigger me to develop and compare different machine learning methods in NIDS. Therefore, in this report, I make the following effeorts:

1. Implement and compare SVM, KNN, DNN, and NDAE on NSL-KDD dataset.
2. Make some improvements to the novel method NDAE by modifying its activation function and loss function.

where the used methods would be introduced in the next section. My source code runs on Google colab and it is available from my github repository <https://github.com/66Tracy/SE6011-assignment/tree/main>.

**Background**

The aforemetioned novel methods combine different algorithms to construct the complicated detection systems. Therefore, in this paper, I try to implement different machine learning methods to explore the feasibility of intelligent NIDS. In this section, I would simply introduce the methods (SVM, KNN, LSTM, AE, DNN) I developed and the common dataset ‘NSL-KDD’ used for NIDS research. *The pictures showed are all made by myself.*

***A. NSL-KDD dataset***

NSL-KDD is a data set suggested to solve some of the inherent problems of the KDD'99 data set, which becomes the benchmark dataset for NIDS research [8]. This data set is comprised of four sub data sets: KDDTest+, KDDTest-21, KDDTrain+, KDDTrain+\_20Percent. And KDDTrain+ will be referred to as train and KDDTest+ will be referred to as test. In my experiments, only KDDTrain+ is used.

KDDTrain+ contains 125973 samples with 42 features. The features are ‘duration time’, ‘protocol type’, ‘servie’, and so on, which recored the states of the network system when it is attacked or normal. And the labels classify these samples into 23 different classes according to attacks they faced. Considering the classification effectiveness, researchers generally re-category the labels items into 5 major kinds of attacks or normal state: ‘normal’, ‘DoS’, ‘Probe’, ‘R2L’, and ‘U2R’. The numbers of these different categories are imbalanced and their ratio relationship is showed below (Fig.1):

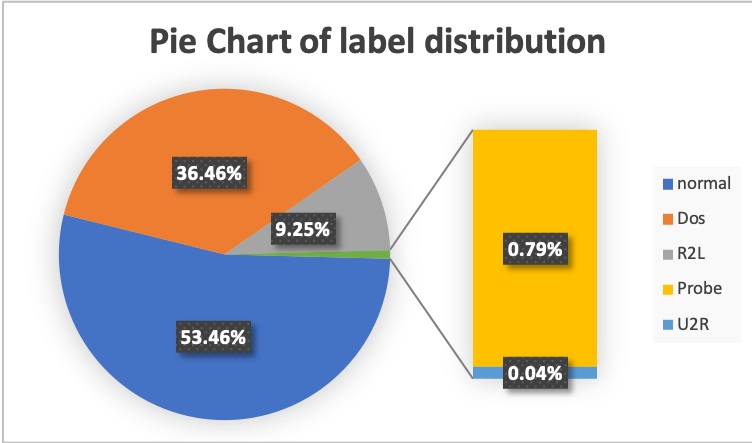


Fig.1 Pie Chart of distribution of labels (made from Excel)

***B. SVM and KNN***

To save the space, this part would simply introduce what is SVM and KNN. SVM is an effective classifier, which aims to learn the largest classificaiton margin [9]. And KNN is a naïve classifier to match and assign label to the target sample based on the k-nearest samples’ labels. The mechanisms of SVM and KNN are showed as Fig.2 below. Seeing the Fig.2 (right), KNN method would classify the target sample in the centre to be ‘red’ since its 5 nearest neighbors are all ‘red’.

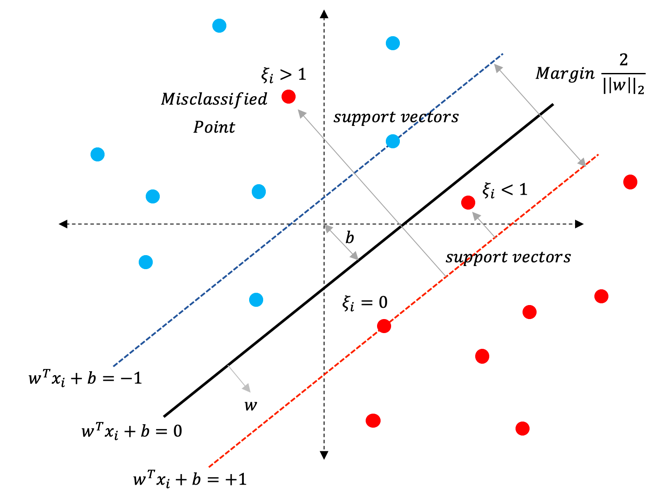
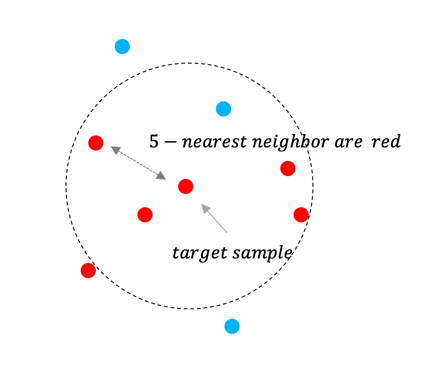
 

Fig.2 (left) SVM’s mechanism, (right) KNN’s mechanism

***C. DNN, AE, and NDAE***

Deep Neural Network(DNN), Auto Encoder(AE), and LSTM are all deep learning algorithms, so I introduce them together. DNN is actually a multi-layer perceptron model which is composed of several fully-conntected layers. The first layer is Input Layer. There are three or four Hidden Layers in the middle, and the last layer is Output Layer, in which each layer has several neurons. The computation between each two layers is mathematically defined as: , where is the non-linear activation function. The output layer of DNN uses softmax function to give the prediction possibility of each class, and generally the class with largest output possibility is assigned as prediction label.

AE is a variant of DNN for unsupervised learning, which can be used as a feature extractor. The learning purpose of AE is to learn the best parameters required to reconstruct its output as close to its input vector as possible. Researchers (Nathan Shone .et al) [5] expect to utilize AE as a non-linear generalization of Principle Component Analysis (PCA) [10]. However, AE is not strictly a complete NIDS since it cannot classify the input by itself. So in [5], Non-symmetric Deep AutoEncoder (NDAE) is proposed to be stacked by two Non-symmetric AE and followed by a Random Forest [11] to predict the output. The architecture of NDAE is showed as Fig.3.

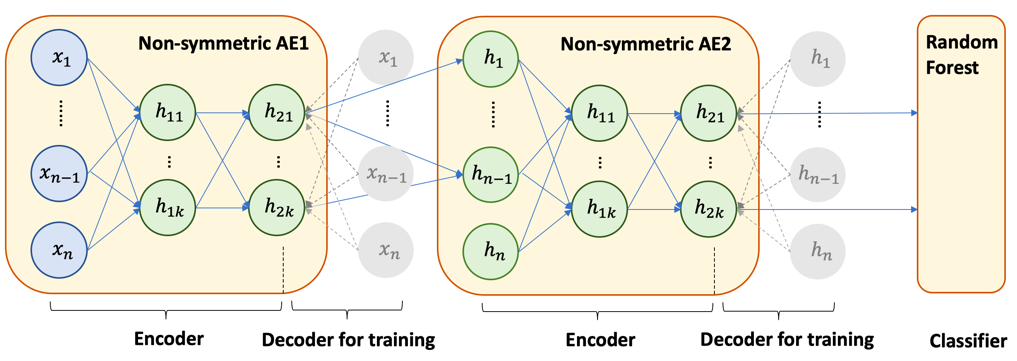


Fig.3 The architecture of NDAE

As drawn by Fig.3, NDAE has three main parts: two Non-symmetric AE and a Random Forest. In training phase, both Encoder and Decoder are used for unsupervised learning. In inference phase, the Decoder is removed (the grey units in Fig.3) and only Encoder is kept to extract the represented embeddings for the input. In fact, these two Non-symmetric AEs are trained respectively, Non-symmetric AE1 is firstly trained by restructing the input vectors. And then, the Decoder of Non-symmetric AE1 is removed. The learned embedding from Non-symmetric AE1’s last hidden layer is used as training data for Non-symmetric AE2. Lastly, random forest is used to classify the output embedding of Non-symmetric AE2.

**Technical Details**

In this section, my experimental process and code details would be simply discussed. The key point of this section would be experimental results analysis.

***A. Environment Setting***

The experiments are runned on Google Colab using python language. Open-source libraries, such as Sklearn, keras, tensorflow, etc, are used to implement the used ML methods. My source code runs on Google colab and it is available from my github repository <https://github.com/66Tracy/SE6011-assignment/tree/main>.

***B. Experiments***

The NSL-KDD dataset used for experiments are preprocessed to replace some word descriptions with one-hot labels. Therefore the number of features are extended to 82. Firstly, I extensively run SVM, KNN, DNN, and NDAE on NSL-KDD dataset. And I suprisingly find that NDAE had a poor performance on NSL-KDD dataset. So I do some unit tests and find that, in the training phase, the reconstruction outcome of the Auto-Encoder in original NDAE suffers from the problem of gradient vanish and the correctness rate is around 20%. After doing some random test, I find that ‘Relu’ activation function and ‘mean avreage loss’ loss function works bettern for NDAE. So I make some improvements to NDAE and named it as NDAE\* in the following tables.

Table.1 The experimental results of SVM

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Accuracy | Recall | F1-score | Number |
| Dos | 0.95 | 0.96 | 0.96 | 11484 |
| Probe | 0.86 | 0.79 | 0.82 | 2947 |
| R2L | 0.61 | 0.60 | 0.60 | 274 |
| U2R | 0.00 | 0.00 | 0.00 | 15 |
| Normal | 0.97 | 0.98 | 0.98 | 16774 |
| Weighted Avg | 0.95 | | | |

Table.2 The experimental results of DNN

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Accuracy | Recall | F1-score | Number |
| Dos | 0.99 | 0.98 | 0.98 | 11484 |
| Probe | 0.96 | 0.93 | 0.94 | 2947 |
| R2L | 0.83 | 0.74 | 0.78 | 274 |
| U2R | 0.00 | 0.00 | 0.00 | 15 |
| Normal | 0.98 | 0.99 | 0.98 | 16774 |
| Weighted Avg | 0.98 | | | |

Table.3 The comparison between KNN and NDAE\*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Accuracy | | Recall | | F1-score | | Number |
|  | KNN | NDAE\* | KNN | NDAE\* | KNN | NDAE\* |  |
| Dos | 0.99 | 0.99 | **0.99** | 0.98 | 0.99 | 0.99 | 11484 |
| Probe | **0.96** | 0.95 | 0.96 | 0.96 | 0.96 | 0.96 | 2947 |
| R2L | **0.90** | 0.88 | **0.87** | 0.84 | **0.88** | 0.86 | 274 |
| U2R | **0.40** | 0.33 | 0.13 | **0.20** | 0.20 | **0.25** | 15 |
| Normal | **0.99** | 0.98 | 0.99 | 0.99 | **0.99** | 0.98 | 16774 |
| Weighted Avg | KNN=0.98  NDAE\*=0.98 | | | | | | |

***C. Experimental Results Analysis***

The testing results of 4 methods are showed in Table.1, 2, 3. In terms of the classification performance, KNN >= NDAE\* > DNN > SVM. So I list the results of SVM and DNN in Table.1 and Table.2 respectively, and compare KNN and NDAE\* in the Table.3. Basd on these three tables, I can conclude that:

1) For the overall performance, it has KNN >= NDAE\* > DNN > SVM. KNN and NDAE\* have even correctness on most of the metrics, and both of them have better performance than DNN and SVM.

2) SVM and DNN cannot deal with the ‘U2R’ class since their both make error predictions. Meanwhile, KNN and NDAE\* are much better with accuracy 0.40 and 0.33 respectively. One of the possible reasons for the bad performance on ‘U2R’ of SVM and DNN is the imbalanced distribution of the 5 classes. Although U2R is relatively low poiibility to happen, it can cause sever harm if it is not detected. NDAE\* has better F1-score means it has better overall precision to balance classification accuracy and recall on ‘U2R’.

**Conclusion**

In this report, extensive experiments of Intelligent NIDS are runned to explore the benchmark of some mainstream machine learning methods. I find that KNN has surprisingly good performance on NSL-KDD dataset. (I cannot write more within the limit of 5 pages.)

**Reference**

[1] Larson D. Distributed denial of service attacks–holding back the flood[J]. Network Security, 2016, 2016(3): 5-7.

[2] Khraisat A, Gondal I, Vamplew P, et al. Survey of intrusion detection systems: techniques, datasets and challenges[J]. Cybersecurity, 2019, 2(1): 1-22.

[3] Zhang Z, Ning H, Shi F, et al. Artificial intelligence in cyber security: research advances, challenges, and opportunities[J]. Artificial Intelligence Review, 2022: 1-25.

[4] Jha J, Ragha L. Intrusion detection system using support vector machine[J]. International Journal of Applied Information Systems (IJAIS), 2013, 3: 25-30.

[5] Shone N, Ngoc T N, Phai V D, et al. A deep learning approach to network intrusion detection[J]. IEEE transactions on emerging topics in computational intelligence, 2018, 2(1): 41-50.

[6] Vinayakumar R, Alazab M, Soman K P, et al. Deep learning approach for intelligent intrusion detection system[J]. Ieee Access, 2019, 7: 41525-41550.

[7] Khan M A, Kim Y. Deep Learning-Based Hybrid Intelligent Intrusion Detection System[J]. Computers, Materials & Continua, 2021, 68(1).

[8] Protić D D. Review of KDD Cup ‘99, NSL-KDD and Kyoto 2006+ datasets[J]. Vojnotehnički glasnik/Military Technical Courier, 2018, 66(3): 580-596.

[9] Lai Z, Chen X, Zhang J, et al. Maximal Margin Support Vector Machine for Feature Representation and Classification[J]. IEEE Transactions on Cybernetics, 2023.

[10] Jangamreddy N. Visualizing and Understanding the Relationship between K-Means Clustering, PCA and Linear Auto encoder[J]. 2019.

[11] Rigatti S J. Random forest[J]. Journal of Insurance Medicine, 2017, 47(1): 31-39.