Hochweriges paper, KDD Cup Dataset nicht gut, mein Datensatz besser, NSL KDD besser aber auch syntehtisch generiert.

Struktur des Paper ist gut, könnte für mich auch passend sein.

Ansatz multy layer autoencoder aber ohne decoder sondern dann shallow learning classifier

A Deep Learning Approach to Network Intrusion Detection

Nathan Shone, Tran Nguyen Ngoc, Vu Dinh Phai, Qi Shi

Abstract—Network Intrusion Detection Systems (NIDSs) play a crucial role in defending computer networks. However, there are concerns regarding the feasibility and sustainability of current approaches when faced with the demands of modern networks. More specifically, these concerns relate to the increasing levels of required human interaction and the decreasing levels of detection accuracy. This paper presents a novel deep learning technique for intrusion detection, which addresses these concerns. We detail our proposed non-symmetric deep auto-encoder (NDAE) for unsupervised feature learning. Furthermore, we also propose our novel deep learning classification model constructed using stacked NDAEs. Our proposed classifier has been implemented in GPU-enabled TensorFlow and evaluated using the benchmark KDD Cup '99 and NSL-KDD datasets. Promising results have been obtained from our model thus far, demonstrating improvements over existing approaches and the strong potential for use in modern NIDSs.

Index Terms—deep learning, anomaly detection, auto-encoders, KDD, network security.

1 Introduction

One of the major challenges in network security is the provision of a robust and effective Network Intrusion Detection System (NIDS). Despite the significant advances in NIDS technology, the majority of solutions still operate using less-capable signature-based techniques, as opposed to anomaly detection techniques. There are several reasons for this reluctance to switch, including the high false error rate (and associated costs), difficulty in obtaining reliable training data, longevity of training data and behavioural dynamics of the system. The current situation will reach a point whereby reliance on such techniques leads to ineffective and inaccurate detection. The specifics of this challenge are to create a widely-accepted anomaly detection technique capable of overcoming limitations induced by the ongoing changes occurring in modern networks.

We are concerned with three main limitations, which contribute to this network security challenge. The first is the drastic growth in the *volume of network data*, which is set to continue. This growth can be predominantly attributed to increasing levels of connectivity, the popularity of the Internet of Things and the extensive adoption of cloud-based services. Dealing with these volumes requires techniques that can analyse data in an increasingly rapid, efficient and effective manner. The second cause is the *in-depth monitoring* and granularity required to improve effectiveness and accuracy. NIDS analysis needs to be more detailed and contextually-aware, which means shifting away from abstract and high-level observations. For example, behavioural changes need to be easily attributable to specific elements of a network, e.g. individual users, operating system versions or protocols. The final cause is the number of *different protocols and the diversity of data* traversing through modern networks. This is possibly the most significant challenge and introduces high-levels of difficulty and complexity when attempting to differentiate between normal and abnormal behaviour. It increases the difficulty in establishing an accurate norm and widens the scope for potential exploitation or zero-day attacks.

In recent years, one of the main focuses within NIDS research has been the application of machine learning and shallow learning techniques such as Naive Bayes, Decision Trees and Support Vector Machines (SVM) [1]. By enlarge, the application of these techniques has offered improvements in detection accuracy. However, there are limitations with these techniques, such as the comparatively high level of human expert interaction required; expert knowledge is needed to process data e.g. identifying useful data and patterns. Not only is this a labour intensive and expensive process but it is also error prone [2]. Similarly, a vast quantity of training data is required for operation (with associated time overheads), which can become challenging in a heterogeneous and dynamic environment.

To address the above limitations, a research area currently receiving substantial interest across multiple domains is that of deep learning. This is an advanced subset of machine learning, which can overcome some of the limitations of shallow learning. Thus far, initial deep learning research has demonstrated that its superior layer-wise feature learning can better or at least match the performance of shallow learning techniques [3]. It is capable of facilitating a deeper analysis of network data and faster identification of any anomalies.

In this paper, we propose a novel deep learning model to enable NIDS operation within modern networks. The model we propose is a combination of deep and shallow learning, capable of correctly analysing a wide-range of network traffic. More specifically, we combine the power of stacking our proposed non-symmetric deep auto-encoder (NDAE) (deep-learning) and the accuracy and speed of Random

- *N. Shone and Q. Shi are with the Department of Computer Science at Liverpool John Moores University, Liverpool, UK.*
E-mail: {n.shone/q.shi}@ljmu.ac.uk
- *T. Ngoc and V. Phai are with Department of Information Security at Le Quy Don Technical University, Hanoi, Vietnam.*
Email: ngoctn@mta.edu.vn, dinhphai88@gmail.com

Manuscript submitted 30 June 2017.

Forest (RF)(shallow learning). We have practically evaluated our model using GPU-enabled TensorFlow and obtained promising results from analysing the KDD Cup '99 and NSL-KDD datasets. We are aware of the limitations of these datasets but they remain widely-used benchmarks amongst similar works, enabling us to draw direct comparisons.

This paper offers the following novel contributions:

- A new NDAE technique for unsupervised feature learning, which unlike typical auto-encoder approaches provides non-symmetric data dimensionality reduction. Hence, our technique is able to facilitate improved classification results when compared with leading methods such as Deep Belief Networks (DBNs).
- A novel classifier model that utilises stacked NDAEs and the RF classification algorithm. By combining both deep and shallow learning techniques to exploit their respective strengths and reduce analytical overheads. We are able to better or at least match results from similar research, whilst significantly reducing the training time.

The remainder of this paper is structured as follows. Section 2 presents relevant background information. Section 3 examines existing research. Section 4 specifies our proposed solution, which is subsequently evaluated in Section 5. Section 6 discusses our findings from the evaluation. Finally the paper concludes in Section 7.

2 Background

In this section, we will provide background information necessary to understand our motivations and the concepts behind the model proposed in this paper.

2.1 NIDS challenges

Network monitoring has been used extensively for the purposes of security, forensics and anomaly detection. However, recent advances have created many new obstacles for NIDSs. Some of the most pertinent issues include:

- **Volume** - The volume of data both stored and passing through networks continues to increase. It is forecast that by 2020, the amount of data in existence will top 44ZB [4]. As such, the traffic capacity of modern networks has drastically increased to facilitate the volume of traffic observed. Many modern backbone links are now operating at wirespeeds of 100Gbps or more. To contextualise this, a 100Gbps link is capable of handling 148,809,524 packets per second [5]. Hence, to operate at wirespeed, a NIDS would need to be capable of completing the analysis of a packet within 6.72ns. Providing NIDS at such a speed is difficult and ensuring satisfactory levels of accuracy, effectiveness and efficiency also presents a significant challenge.
- **Accuracy** - To maintain the aforementioned levels of accuracy, existing techniques cannot be relied upon. Therefore, greater levels of granularity, depth and contextual understanding are required to provide a more holistic and accurate view. Unfortunately, this comes with various financial, computational and time costs.
- **Diversity** - Recent years have seen an increase in the number of new or customised protocols being utilised in modern networks. This can be partially attributed to the number of devices with network and/or Internet connectivity. As a result, it is becoming increasingly difficult to differentiate between normal and abnormal traffic and/or behaviours.
- **Dynamics** - Given the diversity and flexibility of modern networks, the behaviour is dynamic and difficult to predict. In turn, this leads to difficulty in establishing a reliable behavioural norm. It also raises concerns as to the lifespan of learning models.
- **Low-frequency attacks** - These types of attacks have often thwarted previous anomaly detection techniques, including artificial intelligence approaches. The problem stems from imbalances in the training dataset, meaning that NIDS offer weaker detection precision when faced with these types of low frequency attacks.
- **Adaptability** - Modern networks have adopted many new technologies to reduce their reliance on static technologies and management styles. Therefore, there is more widespread usage of dynamic technologies such as containerisation, virtualisation and Software Defined Networks. NIDSs will need to be able to adapt to the usage of such technologies and the side effects they bring about.

2.2 Deep Learning

Deep learning is an advanced sub-field of machine learning, which advances Machine Learning closer to Artificial Intelligence. It facilitates the modelling of complex relationships and concepts [6] using multiple levels of representation. Supervised and unsupervised learning algorithms are used to construct successively higher levels of abstraction, defined using the output features from lower levels [7].

2.2.1 Auto-encoder

A popular technique currently utilised within deep learning research is auto-encoders, which is utilised by our proposed solution (detailed in Section 4). An auto-encoder is an unsupervised neural network-based feature extraction algorithm, which learns the best parameters required to reconstruct its output as close to its input as possible. One of it desirable characteristics is the capability to provide more a powerful and non-linear generalisation than Principle Component Analysis (PCA).

This is achieved by applying backpropagation and setting the target values to be equal to the inputs. In other words, it is trying to learn an approximation to the identity function. An auto-encoder typically has an input layer, output layer (with the same dimension as the input layer) and a hidden layer. This hidden layer normally has a smaller dimension than that of the input (known as an undercomplete or sparse auto-encoder). An example of an auto-encoder is shown in Fig. 1.

Most researchers [8], [9], [10] use auto-encoders as a non-linear transformation to discover interesting data structures, by imposing other constraints on the network, and compare the results with those of PCA (linear transformation). These methods are based on the encoder-decoder paradigm. The input is first transformed into a typically lower-dimensional space (encoder), and then expanded to reproduce the initial data (decoder). Once a layer is trained, its code is fed to the next, to better model highly non-linear dependencies in the input. This paradigm focuses on reducing the dimensionality of input data. To achieve this, there is a special layer - the code layer [9], at the centre of the deep auto-encoder structure. This code layer is used as a compressed feature vector for classification or for combination within a stacked auto-encoder [8].

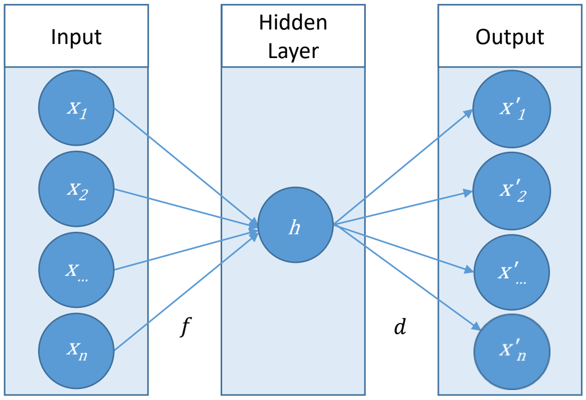


Fig. 1. An example of a single auto-encoder

The hidden layer is used to create a lower dimensionality version of high dimensionality data (known as encoding). By reducing the dimensionality, the auto-encoder is forced to capture the most prominent features of the data distribution. In an ideal scenario, the data features generated by the auto-encoder will provide a better representation of the data points than the raw data itself.

The aim of the auto-encoder is to try and learn the function shown in equation 1.

$$h_{W,b}(x) \approx x \quad (1)$$

Here, h is a non-linear hypothesis using the parameters W (weighting) and b (bias), which can fit the given data (x).

Simply, it tries to learn an approximation to the identity of a function, where x' is most similar to x. The learning process is described as a reconstruction error minimisation function, as shown in equation 2.

$$L(x, d(f(x))) \quad (2)$$

Here, L is a loss function penalising $d(f(x))$ for being dissimilar to x, d is a decoding function and f is an encoding function.

2.2.2 Stacked auto-encoder

Unlike a simple auto-encoder, a deep auto-encoder is composed of two symmetrical deep-belief networks, which typically have four or five shallow layers for encoding, and a second set of four or five layers for decoding. The work by Hinton and Salacukhudinov [9] has produced promising results by implementing a deep learning algorithm to convert high dimensional data to low dimensional data by utilising a deep auto-encoder.

Deep learning can be applied to auto-encoders, whereby the hidden layers are the simple concepts and multiple hidden layers are used to provide depth, in a technique known as a stacked auto-encoder. This increased depth can reduce computational costs and the amount of required training data, as well as yielding greater degrees of accuracy [6]. The output from each hidden layer is used as the input for a progressively higher level. Hence, the first layer of a stacked auto-encoder usually learns first-order features in raw input. The second layer usually learns second-order features relating to patterns in the appearance of the first-order features. Subsequent higher layers learn higher-order features. An illustrative example of a stacked auto-encoder is shown in Fig. 2. Here, the superscript numbers refer to the hidden layer identity and the subscript numbers signify the dimension for that layer.

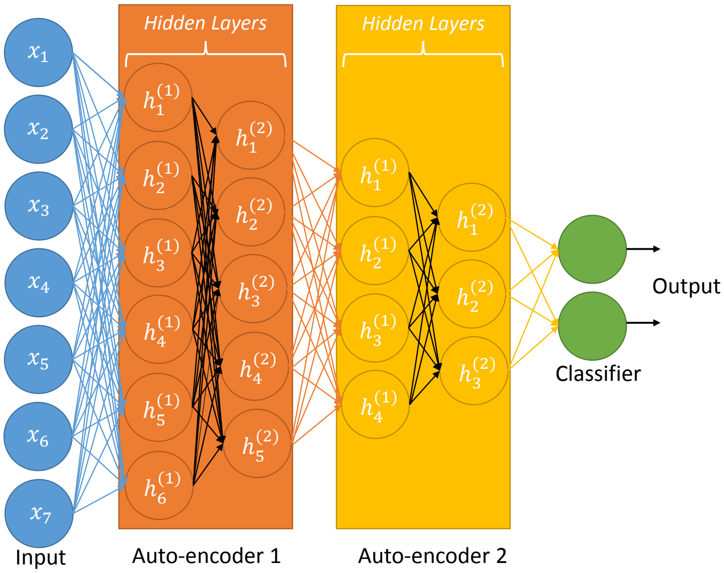


Fig. 2. An example of a stacked auto-encoder

3 Existing Work

Deep learning is garnering significant interest and its application is being investigated within many research domains, such as: healthcare [11], [12]; automotive design [13], [14]; manufacturing [15] and law enforcement [16], [17].

There are also several existing works within the domain of NIDS. In this section, we will discuss the most current notable works.

Dong and Wang undertook a literary and experimental comparison between the use of specific traditional NIDS techniques and deep learning methods [1]. The authors concluded that the deep learning-based methods offered improved detection accuracy across a range of sample sizes and traffic anomaly types. The authors also demonstrated that problems associated with imbalanced datasets can be overcome by using oversampling for which, they used the Synthetic Minority Oversampling Technique (SMOTE).

Zhao et al. [2] presented a state-of-the-art survey of deep learning applications within machine health monitoring.

They experimentally compared conventional machine learning methods against four common deep learning methods (auto-encoders, Restricted Boltzmann Machine (RBM), Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN). Their work concluded that deep learning methods offer better accuracy than conventional methods.

Our literature review identified several proposed deep learning methods specifically for NIDSs.

Alrawashdeh and Purdy [18] proposed using a RBM with one hidden layer to perform unsupervised feature reduction. The weights are passed to another RBM to produce a DBN. The pre-trained weights are passed into a fine tuning layer consisting of a Logistic Regression classifier (trained with 10 epochs) with multi-class soft-max. The proposed solution was evaluated using the KDD Cup '99 dataset. The authors claimed a detection rate of 97.90% and a false negative rate of 2.47%. This is an improvement over results claimed by authors of similar papers.

The work by Kim *et al.* [19] aspired to specifically target advanced persistent threats. They propose a Deep Neural Network (DNN) using 100 hidden units, combined with the Rectified Linear Unit activation function and the ADAM optimiser. Their approach was implemented on a GPU using TensorFlow, and evaluated using the KDD data set. The authors claimed an average accuracy rate of 99%, and summarised that both RNN and Long Short-Term Memory (LSTM) models are needed for improving future defences.

Javaid *et al.* [20] propose a deep learning based approach to building an effective and flexible NIDS. Their method is referred to as self-taught learning (STL), which combines a sparse auto-encoder with softmax regression. They have implemented their solution and evaluated it against the benchmark NSL-KDD dataset. The authors claim some promising levels of classification accuracy in both binary and 5-class classification. Their results show that their 5-class classification achieved an average f-score of 75.76%.

Potluri and Diedrich [21] propose a method using 41 features and their DNN has 3 hidden layers (2 auto-encoders and 1 soft-max). The results obtained were mixed, those focusing on fewer classes were more accurate than those with more classes. The authors attributed this to insufficient training data for some classes.

Cordero et al. [22] proposed an unsupervised method to learn models of normal network flows. They use RNN, auto-encoder and the dropout concepts of deep learning. The exact accuracy of their proposed method evaluated is not fully disclosed.

Similarly, Tang et al. [23] also propose a method to monitor network flow data. The paper lacked details about its exact algorithms but does present an evaluation using the NSL-KDD dataset, which the authors claim gave an accuracy of 75.75% using six basic features.

Kang and Kang [24] proposed the use of an unsupervised DBN to train parameters to initialise the DNN, which yielded improved classification results (exact details of the approach are not clear). Their evaluation shows improved performance in terms of classification errors.

Hodo *et al.* [25] have produced a comprehensive taxonomy and survey on notable NIDSs approaches that utilise deep and shallow learning. They have also aggregated some of the most pertinent results from these works.

In addition, there is other relevant work, including the DDoS detection system proposed by Niyaz *et al.* [26]. They propose a deep learning-based DDoS detection system for a software defined network (SDN). Evaluation is performed using custom generated traffic traces. The authors claim to have achieved binary classification accuracy of 99.82% and 8-class classification accuracy of 95.65%. However, we feel that drawing comparisons with this paper would be unfair due to the contextual difference of the dataset. Specifically, benchmark KDD datasets cover different distinct categories of attack, whereas the dataset used in this paper focuses on subcategories of the same attack.

You et al. [16] propose an automatic security auditing tool for short messages (SMS). Their method is based upon the RNN model. The authors claimed that their evaluations resulted in an accuracy rate of 92.7%, thus improving existing classification methods (e.g. SVM and Naive Bayes).

Wang et al. [27] propose an approach for detecting malicious JavaScript. Their method uses a 3 layer SdA with linear regression. It was evaluated against other classifier techniques, showing that it had the highest true positive rate but the second best false positive rate.

The work by Hou et al. [3] outlines their commercial Android malware detection framework, Deep4MalDroid. Their method involves the use of stacked auto-encoders with best accuracy resulting from 3 layers. The 10-fold cross validation was used, showing that in comparison to shallow learning, their approach offers improved detection performance.

Lee et al. [28] propose a deep-learning approach to fault monitoring in semiconductor manufacturing. They use a Stacked de-noising Auto-encoder (SdA) approach to provide an unsupervised learning solution. A comparison with conventional methods has demonstrated that throughout different use cases the approach increases accuracy by up to 14%. in different use cases. They also concluded that among the SdAs analysed (1-4 layers) those with 4 layers produced the best results.

The findings from our literature review have shown that despite the high detection accuracies being achieved, there is still room for improvement. Such weaknesses include the reliance on human operators, long training times, inconsistent or average accuracy levels and the heavy modification of datasets (e.g. balancing or profiling). The area is still in an infantile stage, with most researchers still experimenting on combining various algorithms (e.g. training, optimisation, activation and classification) and layering approaches to produce the most accurate and efficient solution for a specific dataset. Hence, we believe the model and work presented in this paper will be able to make a valid contribution to the current pool of knowledge.

4 Proposed Methodology

4.1 Non-symmetric deep auto-encoder

Decreasing the reliance on human operators is a crucial requirement for future-proofing NIDSs. Hence, our aim is to devise a technique capable of providing reliable unsupervised feature learning, which can improve upon the performance and accuracy of existing techniques.

This paper introduces our NDAE, which is an auto-encoder featuring *non-symmetrical* multiple hidden layers. Fundamentally, this involves the proposed shift from the encoder-decoder paradigm (symmetric) and towards utilising just the encoder phase (non-symmetric). The reasoning behind this is that given the correct learning structure, it is be possible to reduce both computational and time overheads, with minimal impact on accuracy and efficiency. NDAE can be used as a hierarchical unsupervised feature extractor that scales well to accommodate high-dimensional inputs. It learns non-trivial features using a similar training strategy to that of a typical auto-encoder. An illustrated example of this is presented in Fig. 3.

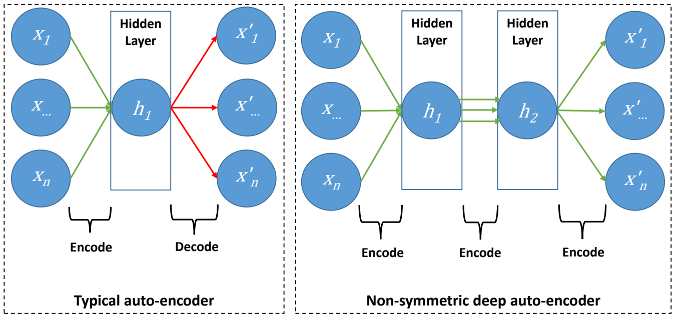


Fig. 3. Comparison of a typical auto-encoder and a NDAE

The proposed NDAE takes an input vector $x \in R^d$ and step-by-step maps it to the latent representations $h_i \in R^{d_i}$ (here d represents the dimension of the vector) using a deterministic function shown in equation (3) below:

$$h_i = \sigma(W_i.h_{i-1} + b_i); i = \overline{1, n}, \tag{3}$$

Here, $h_0 = x$, σ is an activation function (in this work we use sigmoid function $\sigma(t) = 1/(1 + e^{-t})$) and n is the number of hidden layers.

Unlike a conventional auto-encoder and deep auto-encoder, the proposed NDAE does not contain a decoder and its output vector is calculated by a similar formula to equation (4) as the latent representation.

$$y = \sigma(W_{n+1}.h_n + b_{n+1}) \tag{4}$$

The estimator of the model $\theta = (W_i, b_i)$ can be obtained by minimising the square reconstruction error over m training samples $(x^{(i)}, y^{(i)})_{i=1}^m$, as shown in equation (5).

$$E(\theta) = \sum_{i=1}^{m} (x^{(i)} - y^{(i)})^2 \tag{5}$$

4.2 Stacked non-symmetric deep auto-encoders

In this subsection, we detail the novel deep learning classification model we have created to address the problems identified with current NIDSs.

Fundamentally, our model is based upon using our NDAE technique (outlined in Section 4.1) for deep learning. This is achieved by stacking our NDAEs to create a deep learning hierarchy. Stacking the NDAEs offers a layer-wise unsupervised representation learning algorithm, which will allow our model to learn the complex relationships between different features. It also has feature extraction capabilities, so it is able to refine the model by prioritising the most descriptive features.

Due to the data that we envisage this model using, we have designed the model to handle large and complex datasets (further details on this are provided in 6). Despite the 42 features present in the KDD Cup '99 and NSL-KDD datasets being comparatively small, we maintain that it provides a benchmark indication as to the model's capability.

However, the classification power of stacked auto-encoders with a typical soft-max layer is relatively weak compared to other discriminative models including RF, KNN and SVM. Hence, we have combined the deep learning power of our stacked NDAEs with a shallow learning classifier. For our shallow learning classifer, we have decided upon using Random Forest. Current comparative research such as that by Choudhury and Bhowal [29], and Anbar *et al.* [30] shows that RF is one of the best algorithms for intrusion detection. These are findings that were replicated by our own initial tests. Additionally, there are many examples of current intrusion detection research also utilising RF such as [31] and [32].

RF is basically an ensemble learning method, the principle of which is to group 'weak learners' to form a 'strong learner' [33]. In this instance, numerous individual decision trees (the weak learners) are combined to form a forest. RF can be considered as the bagging (records are selected at random with replacement from the original data) of these un-pruned decision trees, with a random selection of features at each split. It boasts advantages such as low levels of bias, robustness to outliers and overfitting correction, all of which would be useful in a NIDS scenario.

In our model, we train the RF classifier using the encoded representations learned by the stacked NDAEs to classify network traffic into normal data and known attacks.

In deep learning research, the exact structure of a model dictates its success. Currently, researchers are unable to explain what makes a successful deep learning structure. The exact structure of our model has resulted from experimented with numerous structural compositions to achieve the best results. The final structure of our proposed model is shown in Fig. 4.

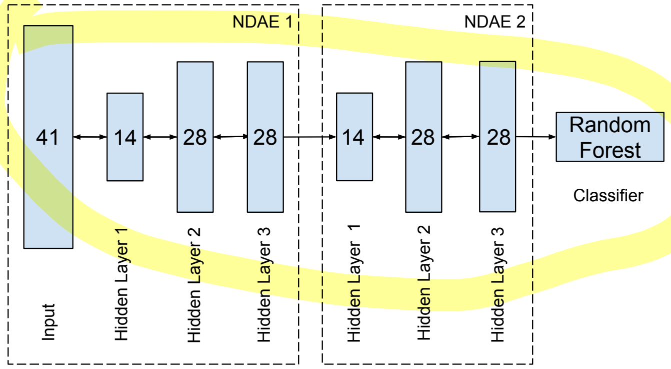


Fig. 4. Stacked NDAE Classification Model

As per Fig. 4, our model uses two NDAEs arranged in a stack, and is combined with the RF algorithm. Each NDAE has 3 hidden layers, with each hidden layer using the same number of neurons as that of features (indicated by the numbering in the diagram). These exact parameters were determined by cross-validating numerous combinations (i.e. numbers of neurons and hidden layers), until the most effective was identified. This allows for performance evaluation without the risk of overfitting. For our experiments, we used the 10-fold cross-validation approach on the NSL-KDD dataset using Scikit Learn. The result for our final model structure was 0.995999 +/- 0.000556, which is a very promising result.

5 Evaluation & Results

Similar to most existing deep learning research, our proposed classification model (Section 4.2) was implemented using TensorFlow. All of our evaluations were performed using GPU-enabled TensorFlow running on a 64-bit Ubuntu 16.04 LTS PC with an Intel Xeon 3.60GHz processor, 16 GB RAM and an NVIDIA GTX 750 GPU.

To perform our evaluations, we have used the KDD Cup '99 and NSL-KDD datasets. Both of these datasets are considered as benchmarks within NIDS research. Furthermore, using these datasets assists in drawing comparisons with existing methods and research.

Throughout this section, we will be using the metrics defined below:

- *True Positive (TP)* - Attack data that is correctly classified as an attack.
- *False Positive (FP)* - Normal data that is incorrectly classified as an attack.
- *True Negative (TN)* - Normal data that is correctly classified as normal.
- *False Negative (FN)* - Attack data that is incorrectly classified as normal.

We will be using the following measures to evaluate the performance of our proposed solution:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{6}$$

The accuracy measures the proportion of the total number of correct classifications.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{7}$$

The precision measures the number of correct classifications penalised by the number of incorrect classifications.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{8}$$

The recall measures the number of correct classifications penalised by the number of missed entries.

$$\text{False Alarm} = \frac{FP}{FP + TN} \tag{9}$$

The false alarm measures the proportion of benign events incorrectly classified as malicious.

$$\text{F-score} = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{10}$$

The F-score measures the harmonic mean of precision and recall, which serves as a derived effectiveness measurement.

5.1 Datasets

This paper utilises the KDD Cup '99 and NSL-KDD benchmark datasets. Both of which have been used extensively in IDS research involving traffic with both normal and abnormal connections.

5.1.1 KDD Cup '99

The KDD Cup '99 dataset was used in DARPA's IDS evaluation program [34]. The data consists of 4 gigabytes-worth of compressed tcpdump data resulting from 7 weeks of network traffic. This can be processed into about 5 million connection records, each with about 100 bytes. It consists of approximately 4,900,000 single connection vectors each of which contains 41 features. These include Basic features (e.g. protocol type, packet size), Domain knowledge features (e.g. number of failed logins) and timed observation features (e.g. % of connections with SYN errors). Each vector is labelled as either normal or as an attack (of which there are 22 specific attack types, as outlined in Table 1).

It is common practice to use 10% of the full size dataset, as this provides a suitable representation with reduced computational requirements. This 10% subset is produced and disseminated alongside the original dataset. In this paper, we use the 10% (herein referred to as KDD Cup '99) subset, which contains 494,021 training records and 311,029 testing records. The exact composition is shown in Table 1.

The KDD Cup '99 dataset needs pre-processing to be successfully utilised with our proposed stacked NDAE model. This is because our model operates using only numeric values but one record in the dataset has a mixture of numeric and symbolic values, so a data transformation was needed to convert them. In addition integer values also need normalisation as they were mixed with floating point values between 0 and 1, which would make learning difficult.

5.1.2 NSL-KDD

The newer NSL-KDD dataset, which was produced by Tavallaee *et al.* to overcome the inherent problems of the KDD '99 data set, which are discussed in [35]. Although, this new version of the dataset still suffers from some of the problems discussed by McHugh in [36] and may not be a perfect representation of existing real networks. Most current NIDS research still uses this dataset, so we believe it remains an effective benchmark to help researchers compare different methods.

The NSL-KDD dataset is fundamentally the same structure as the KDD Cup '99 dataset (i.e. it has 22 attack patterns or normal traffic, and fields for 41 features). We will be using the whole NSL-KDD dataset for our evaluations, the composition of which is also shown in Table 1.

In Table 1, some of the attack patterns have been highlighted. This indicates attack patterns that contain less than 20 occurrences in the dataset. 20 is the minimum threshold required for accurate levels of training and evaluation. So, for this paper these attacks have been omitted.

One of the most prominent techniques currently used within deep learning research is DBNs [7], [1], [2]. One notable publication on the technique is by Alrawashdeh and Purdy [18], where the authors propose the use of a DBN model for NIDSs. Hence, for our evaluation we draw

TABLE 1
Composition of Datasets

		10% KDD '99		NSL-KDD	
Category	**Attack Type**	**Train**	**Test**	**Train**	**Test**
DoS	'back'	2203	1098	956	359
	'land'	21	9	18	7
	'neptune'	107201	58001	41214	4657
	'pod'	264	87	201	41
	'smurf'	280790	164091	2646	665
	'teardrop'	979	12	892	12
Probe	'ipsweep'	1247	306	3599	141
	'nmap'	231	84	1493	73
	'portsweep'	1040	354	2931	157
	'satan'	1589	1633	3633	735
R2L	'ftp_write'	8	3	8	3
	'guess_password'	53	4367	53	1231
	'imap'	12	1	11	1
	'multihop'	7	18	7	18
	'phf'	4	2	4	2
	'spy'	2	0	2	0
	'warezclient'	1020	0	890	0
	'warezmaster'	20	1602	20	944
U2R	'loadmodule'	9	2	9	2
	'buffer_overflow'	30	22	30	20
	'rootkit'	10	13	10	13
	'perl'	3	2	3	2
Normal		97278	60593	67343	9711
Total		494021	292300	125973	18794

a direct comparison between the results obtained from our proposed model and the DBN model. We will also compare the results of our model against those published by Alrawashdeh and Purdy.

5.2 KDD Cup '99

In this section, we evaluate the 5-class classification performance of our proposed classification model against the DBN model published in [18], using the KDD Cup '99 dataset as outlined in the previous subsection.

The results obtained from the 5-class analysis of the KDD Cup '99 dataset by both the DBN model in [18] and our stacked NDAE model are presented in Table 2. By comparing the results of both models, we can see that overall our stacked NDAE model the effectiveness and accuracy of our results are better than, or comparable with those achieved by the model in [18]. However, notable exceptions to this are the "U2R" and "R2L" classes, which will be discussed in Section 6.

Time efficiency is an important consideration for our model, particularly when applied within a NIDS. Hence, we have measured the training time required by our stacked NDAE model and a DBN model to analyse the KDD '99 dataset. However, it would not be a fair to draw comparisons with [18] in this respect, due to differences in the hardware and software used. Therefore, we have implemented a DBN model in TensorFlow, and the results obtained are presented in Table 3.

As Table 3 shows, the non-symmetric approach of our model is able to accomplish a significant reduction in required training time, offering an average reduction of 97.72%. Hence, it is promising that our model can maintain the high levels of accuracy, whilst drastically reducing the required training time.

TABLE 3
KDD '99 Training Time

No. Neurons in Hidden Layers	Training Time (s)		Time Saving (%)
	DBN	S-NDAE	
8	54660	2024	96.30
14	122460	2381	98.06
22	204900	2446	98.81

5.3 NSL-KDD

Unfortunately, the paper [18] does not provide evaluations using the NSL-KDD dataset. Thus we will be using the previously-discussed TensorFlow DBN model for comparisons. To maximise comparability we have undertaken two separate evaluations based on (a) *5-class classification* as used in KDD Cup '99, and (b) *13-class classification* from NSL-KDD (this selection is explained in Section 5.1).

5.3.1 5-Class Classification

By using the same 5 generic class labels as used in the KDD Cup '99 dataset, we can compare the performance of the two models between the two datasets. It also aids comparability against similar works adopting this strategy. The performance results are presented in Table 4 and illustrated by the Receiver Operating Characteristic (ROC) curve in Figure 5.

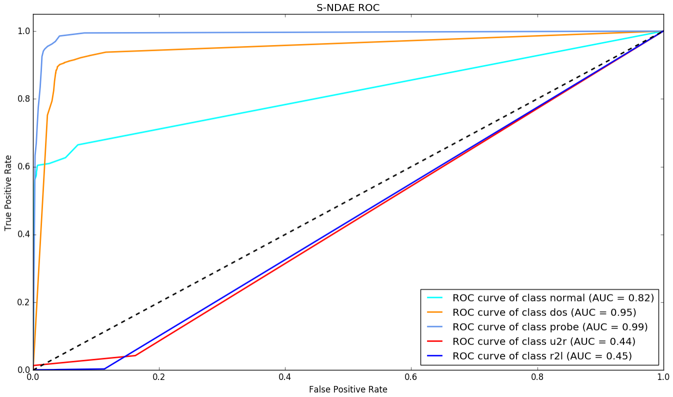


Fig. 5. ROC Curve for NSL-KDD 5-Class

From the table, it is evident that our model offers increased accuracy, precision, recall, effectiveness (F-score) and the false alarm rate, when compared to the DBN approach.

5.3.2 13-Class Classification

As discussed previously, our model is designed to work with larger and complex datasets. Therefore, we evaluate our model's classification capabilities on a 13-class dataset. These 13 labels are those with more than the minimum 20 entries. The purpose of this analysis is to compare the stability of our model when the number of attack classes increases. Therefore, we do not compare these results against another model. The corresponding performance analysis is presented in Table 5. It is evident when these results are compared to those in Table 4 (the *5-class* performance) that overall it performs better, with the average accuracy increasing from 85.42% to 89.22%. One of our initial goals was to support the granularity required by modern networks.

TABLE 2
KDD Cup '99 Performance

Attack	No.	No.	Accuracy (%)		Precision (%)		Recall (%)		F-Score (%)		False Alarm (%)	
Class	Training	Attacks	DBN	S-NDAE	DBN	S-NDAE	DBN	S-NDAE	DBN	S-NDAE	DBN	S-NDAE
Normal	97278	60593	99.49	99.49	94.51	100.00	99.49	99.49	96.94	99.75	5.49	8.92
DoS	391458	223298	99.65	99.79	98.74	100.00	99.65	99.79	99.19	99.89	1.26	0.04
Probe	4107	2377	14.19	98.74	86.66	100.00	14.19	98.74	24.38	99.36	13.34	10.83
R2L	1126	5993	89.25	9.31	100.00	100.00	89.25	9.31	94.32	17.04	0.00	0.71
U2R	52	39	7.14	0.00	38.46	0.00	7.14	0.00	12.05	0.00	61.54	100.00
Total	494021	292300	97.90	97.85	97.81	99.99	97.91	97.85	97.47	98.15	2.10	2.15

TABLE 4
NSL-KDD 5-class Performance

Attack	No.	No.	Accuracy (%)		Precision (%)		Recall (%)		F-Score (%)		False Alarm (%)	
Class	Training	Attacks	DBN	S-NDAE	DBN	S-NDAE	DBN	S-NDAE	DBN	S-NDAE	DBN	S-NDAE
DoS	45927	5741	87.96	94.58	100.00	100.00	87.96	94.58	93.60	97.22	8.80	1.07
Normal	67343	9711	95.64	97.73	100.00	100.00	95.64	97.73	97.77	98.85	24.29	20.62
Probe	11656	1106	72.97	94.67	100.00	100.00	72.97	94.67	84.37	97.26	18.40	16.84
R2L	995	2199	0.00	3.82	0.00	100.00	0.00	3.82	0.00	7.36	0.00	3.45
U2R	52	37	0.00	2.70	0.00	100.00	0.00	2.70	0.00	5.26	0.00	50.00
Total	125973	18794	80.58	85.42	88.10	100.00	80.58	85.42	84.08	87.37	19.42	14.58

Therefore, these results are a promising indication that our model can perform better when faced with more detailed and granular datasets.

Timeliness is critical in modern NIDS, thus we also evaluate the training time required for the NSL-KDD dataset. The results of this comparison are shown in Table 6.

TABLE 6
NSL-KDD Time Comparison

No. Neurons	Training Time (s)		Time
in Hidden Layers	DBN	S-NDAE	Saving (%)
8	1198.08	644.84	46.18
14	10984.04	722.54	93.42
22	21731.76	1091.97	94.98

From these results, we can see that through the different hidden layer compositions, our model is able to consistently reduce the required training time compared with DBN.

6 Discussion

Our evaluations show that our proposed stacked NDAE model has produced a promising set of results.

6.1 5-Class KDD Cup '99 Classification

With regards to the KDD Cup '99 dataset evaluation, the results show that our model is able to offer an average accuracy of 97.85%. more specifically, the results show that our accuracy is better than or comparable with the work in [18], in 3 out of 5 classes. It is also a significant improvement on other deep learning methods such as [23]. However, it is noted that the results for "R2L" and "U2L" attack classes are anomalous. The stacked NDAE model requires greater amounts of data to learn from. Unfortunately, due to the smaller number of training datum available, the results achieved are less stable. Despite this, it is evident from the performance analysis that our model can offer improved precision, recall and F-score, especially for larger classes.

Furthermore, our model managed to produce these comparable performance results, whilst consistently reducing the required training time by an average of 97.72%.

6.2 5-Class NSL-KDD Classification

With regards to the NSL-KDD dataset, we can see from the results that throughout all of the measures our model yields superior level of performance in 3 of the 5 classes. Notably, the model offered a total accuracy rate of 85.42%, which improves upon the DBN model by just under 5%. It also offered a 4.84% reduction in the false alarm rate. The results also re-emphasise the point made, that our model doesn't handle smaller classes ("R2L" and "U2R") as well.

Another important factor is that the time required to train our model is drastically reduced, yielding an average time saving of 78.19% against DBN. This is of critical importance particularly for application in a NIDS.

6.3 13-Class NSL-KDD Classification

The results from the 13-Class classification evaluate demonstrate that our model was able to offer a 3.8% improvement on its own accuracy simply by using a more granular dataset. This supports our claim that the model is able to work more effectively with larger and complex datasets.

Furthermore, the larger dataset gives a better insight into the weakness in our model. As it can be seen from the results, there is a direct correlation between the size of the training datasets for each label and the accuracy/error rates. This supports our observation that the smaller classes (in this case "back", "guess_password", "tear_drop" and "warez_client") yield lower levels of accuracy using our model.

However, it must also be noted that the larger classes yielded consistently high rates throughout all of the performance measures.

6.4 Comparison With Related Works

We have also compared the results from our stacked NDAE model against the results obtained from similar deep learning-based NIDSs.

In [26], the authors claim their 5-class classification of the NSL-KDD dataset produced an f-score of 75.76%. Their

TABLE 5
NSL-KDD 13-class Performance

Label	No. Training	No. Attack	Accuracy (%)	Precision (%)	Recall (%)	F-score (%)	False Alarm (%)
’back’	956	359	36.77	100.00	36.77	53.77	0.00
’buffer_overflow’	30	20	0.00	0.00	0.00	0.00	100.0
’guess_password’	53	1231	0.00	0.00	0.00	0.00	0.00
’ipsweep’	3599	141	98.58	100.00	98.58	99.29	6.71
’neptune’	41214	4657	98.05	100.00	98.05	99.01	0.00
’nmap’	1493	73	100.00	100.00	100.00	100.00	0.00
’normal’	67343	9711	97.91	100.00	97.91	98.94	14.70
’pod’	201	41	92.68	100.00	92.68	96.20	28.30
’portsweep’	2931	157	95.54	100.00	95.54	97.72	44.03
’satan’	3633	735	82.45	100.00	82.45	90.38	13.92
’smurf’	2646	665	99.10	100.00	99.10	99.55	0.15
’teardrop’	892	12	100.00	100.00	100.00	100.00	75.51
’warezclient’	890	0	0.00	0.00	0.00	0.00	0.00
Total	125881	17802	89.22	92.97	89.22	90.76	10.78

recall and precision results are not listed but the bar charts show them to be around 69% and 83% respectively. Our model has produced superior results by offering f-score of 87.37%, recall of 85.42% and precision of 100.00%.

Tang *et al*. [23] claim that their Deep Neural Network (DNN) approach achieved an accuracy of 75.75% when performing a 5-class classification of the NSL-KDD dataset. This is result is lower than our achieved accuracy of 85.42%.

Whilst classifying the KDD Cup ’99 dataset, Kim *et al*. [37] claim they have achieved an accuracy of accuracy of 96.93%. Also Gao *et al*. [38] claim their deep learning DBN model achieved an accuracy of 93.49%. Both of these results are less than the 97.85% accomplished by our model.

These comparisons show that our model’s results are very promising when compared to other current deep learning-based methods.

7 Conclusion & Future Work

In this paper, we have discussed the problems faced by existing NIDS techniques. In response to this we have proposed our novel NDAE method for unsupervised feature learning. We have then built upon this by proposing a novel classification model constructed from stacked NDAEs and the RF classification algorithm.

We have implemented our proposed model in TensorFlow and performed extensive evaluations on its capabilities. For our evaluations we have utilised the benchmark KDD Cup ’99 and NSL-KDD datasets and achieved very promising results.

Our results have demonstrated that our approach offers high levels of accuracy, precision and recall together with reduced training time. Most notably, we have compared our stacked NDAE model against the mainstream DBN technique. These comparisons have demonstrated that our model offers up to a 5% improvement in accuracy and training time reduction of up to 98.81%. Unlike most previous work, we have evaluated the capabilities of our model based on both benchmark datasets, revealing a consistent level of classification accuracy.

Although our model has achieved the above promising results, we acknowledge that it is not perfect and there is further room for improvement.

In our future work, the first avenue of exploration for improvement will be to assess and extend the capability of our model to handle zero-day attacks. We will then look to expand upon our existing evaluations by utilising real-world backbone network traffic to demonstrate the merits of the extended model.

Acknowledgements

The authors would like to thank the Royal Academy of Engineering for their support provided through the Newton Research Collaboration Programme.

References

[1] B. Dong and X. Wang, “Comparison deep learning method to traditional methods using for network intrusion detection,” in *2016 8th IEEE International Conference on Communication Software and Networks (ICCSN)*. Beijing, China: IEEE, jun 2016, pp. 581–585.

[2] R. Zhao, R. Yan, Z. Chen, K. Mao, P. Wang, and R. X. Gao, “Deep Learning and Its Applications to Machine Health Monitoring: A Survey,” *submitted to IEEE Transactions on Neural Networks and Learning Systems*, vol. 14, no. 8, pp. 1–14, dec 2016. [Online]. Available: http://arxiv.org/abs/1612.07640

[3] S. Hou, A. Saas, L. Chen, and Y. Ye, “Deep4MalDroid: A Deep Learning Framework for Android Malware Detection Based on Linux Kernel System Call Graphs,” in *2016 IEEE/WIC/ACM International Conference on Web Intelligence Workshops (WIW)*. Omaha, Nebraska, USA: IEEE, oct 2016, pp. 104–111.

[4] IDC, “Executive Summary: Data Growth, Business Opportunities, and the IT Imperatives — The Digital Universe of Opportunities: Rich Data and the Increasing Value of the Internet of Things,” IDC, MA, USA, Tech. Rep., 2014. [Online]. Available: https://www.emc.com/leadership/digital-universe/2014iview/executive-summary.htm

[5] Juniper Networks, “Juniper Networks - How many Packets per Second per port are needed to achieve Wire-Speed?” 2015. [Online]. Available: https://kb.juniper.net/InfoCenter/index?page=content{\&}id=KB14737

[6] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[7] L. Deng, “Deep Learning: Methods and Applications,” *Foundations and Trends in Signal Processing*, vol. 7, no. 3-4, pp. 197–387, aug 2014.

[8] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, “Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion,” *Journal of Machine Learning Research*, vol. 11, no. Dec, pp. 3371–3408, 2010.

[9] G. E. Hinton and R. R. Salakhutdinov, “Reducing the dimensionality of data with neural networks,” *science*, vol. 313, no. 5786, pp. 504–507, 2006.

[10] Y. Wang, H. Yao, and S. Zhao, “Auto-encoder based dimensionality reduction,” *Neurocomputing*, vol. 184, pp. 232–242, 2016.

[11] Z. Liang, G. Zhang, J. X. Huang, and Q. V. Hu, “Deep learning for healthcare decision making with EMRs,” in *2014 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, no. Cm. IEEE, nov 2014, pp. 556–559.

[12] S. P. Shashikumar, A. J. Shah, Q. Li, G. D. Clifford, and S. Nemati, "A deep learning approach to monitoring and detecting atrial fibrillation using wearable technology," in *2017 IEEE EMBS International Conference on Biomedical & Health Informatics (BHI)*. Florida, USA: IEEE, 2017, pp. 141–144.

[13] F. Falcini, G. Lami, and A. M. Costanza, "Deep Learning in Automotive Software," *IEEE Software*, vol. 34, no. 3, pp. 56–63, may 2017. [Online]. Available: http://ieeexplore.ieee.org/document/7927925/

[14] A. Luckow, M. Cook, N. Ashcraft, E. Weill, E. Djerekarov, and B. Vorster, "Deep learning in the automotive industry: Applications and tools," in *2016 IEEE International Conference on Big Data (Big Data)*. IEEE, dec 2016, pp. 3759–3768. [Online]. Available: http://ieeexplore.ieee.org/document/7841045/

[15] H. Lee, Y. Kim, and C. O. Kim, "A Deep Learning Model for Robust Wafer Fault Monitoring With Sensor Measurement Noise," *IEEE Transactions on Semiconductor Manufacturing*, vol. 30, no. 1, pp. 23–31, feb 2017.

[16] L. You, Y. Li, Y. Wang, J. Zhang, and Y. Yang, "A deep learning-based RNNs model for automatic security audit of short messages," in *2016 16th International Symposium on Communications and Information Technologies (ISCIT)*. Qingdao, China: IEEE, sep 2016, pp. 225–229.

[17] R. Polishetty, M. Roopaei, and P. Rad, "A Next-Generation Secure Cloud-Based Deep Learning License Plate Recognition for Smart Cities," in *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*. Anaheim, California, USA: IEEE, dec 2016, pp. 286–293.

[18] K. Alrawashdeh and C. Purdy, "Toward an Online Anomaly Intrusion Detection System Based on Deep Learning," in *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*. Anaheim, California, USA: IEEE, dec 2016, pp. 195–200.

[19] Jin Kim, Nara Shin, S. Y. Jo, and Sang Hyun Kim, "Method of intrusion detection using deep neural network," in *2017 IEEE International Conference on Big Data and Smart Computing (BigComp)*. Hong Kong, China: IEEE, feb 2017, pp. 313–316.

[20] A. Javaid, Q. Niyaz, W. Sun, and M. Alam, "A deep learning approach for network intrusion detection system," in *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies*, ser. BICT'15. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2016, pp. 21–26. [Online]. Available: http://dx.doi.org/10.4108/eai.3-12-2015.2262516

[21] S. Potluri and C. Diedrich, "Accelerated deep neural networks for enhanced Intrusion Detection System," in *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*, vol. 2016-Novem. Berlin, Germany: IEEE, sep 2016, pp. 1–8.

[22] C. Garcia Cordero, S. Hauke, M. Muhlhauser, and M. Fischer, "Analyzing flow-based anomaly intrusion detection using Replicator Neural Networks," in *2016 14th Annual Conference on Privacy, Security and Trust (PST)*. Auckland, New Zeland: IEEE, dec 2016, pp. 317–324.

[23] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, "Deep learning approach for network intrusion detection in software defined networking," in *2016 International Conference on Wireless Networks and Mobile Communications (WINCOM)*. IEEE, Oct 2016, pp. 258–263.

[24] M.-J. Kang and J.-w. Kang, "Intrusion Detection System Using Deep Neural Network for In-Vehicle Network Security," *PLOS ONE*, vol. 11, no. 6, p. e0155781, jun 2016.

[25] E. Hodo, X. J. A. Bellekens, A. Hamilton, C. Tachtatzis, and R. C. Atkinson, "Shallow and deep networks intrusion detection system: A taxonomy and survey," *CoRR*, vol. abs/1701.02145, 2017. [Online]. Available: http://arxiv.org/abs/1701.02145

[26] Q. Niyaz, W. Sun, and A. Y. Javaid, "A deep learning based ddos detection system in software-defined networking (SDN)," *CoRR*, vol. abs/1611.07400, 2016. [Online]. Available: http://arxiv.org/abs/1611.07400

[27] Y. Wang, W.-d. Cai, and P.-c. Wei, "A deep learning approach for detecting malicious JavaScript code," *Security and Communication Networks*, vol. 9, no. 11, pp. 1520–1534, jul 2016.

[28] H.-W. Lee, N.-r. Kim, and J.-h. Lee, "Deep Neural Network Self-training Based on Unsupervised Learning and Dropout," *The International Journal of Fuzzy Logic and Intelligent Systems*, vol. 17, no. 1, pp. 1–9, mar 2017. [Online]. Available: http://www.ijfis.org/journal/view.html?doi=10.5391/IJFIS.2017.17.1.1

[29] S. Choudhury and A. Bhowal, "Comparative analysis of machine learning algorithms along with classifiers for network intrusion detection," in *2015 International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials (ICSTM)*, May 2015, pp. 89–95.

[30] M. Anbar, R. Abdullah, I. H. Hasbullah, Y. W. Chong, and O. E. Elejla, "Comparative performance analysis of classification algorithms for intrusion detection system," in *2016 14th Annual Conference on Privacy, Security and Trust (PST)*, Dec 2016, pp. 282–288.

[31] Y. Chang, W. Li, and Z. Yang, "Network intrusion detection based on random forest and support vector machine," in *2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC)*, July 2017, pp. 635–638.

[32] Y. Y. Aung and M. M. Min, "An analysis of random forest algorithm based network intrusion detection system," in *2017 18th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*, June 2017, pp. 127–132.

[33] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[34] S. J. Stolfo, W. Fan, W. Lee, A. Prodromidis, and P. K. Chan, "Cost-based modeling for fraud and intrusion detection: Results from the jam project," in *In Proceedings of the 2000 DARPA Information Survivability Conference and Exposition*. IEEE, 2000, pp. 130–144.

[35] M. Tavallaee, E. Bagheri, W. Lu, and A.-A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *Second IEEE Symposium on Computational Intelligence for Security and Defence Applications*. IEEE, 2009, pp. 53–58.

[36] J. McHugh, "Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory," *ACM Transactions on Information and System Security*, vol. 3, no. 4, pp. 262–294, 2000.

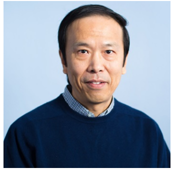
[37] J. Kim, J. Kim, H. L. T. Thu, and H. Kim, "Long short term memory recurrent neural network classifier for intrusion detection," in *2016 International Conference on Platform Technology and Service (PlatCon)*. IEEE, Feb 2016, pp. 1–5.

[38] N. Gao, L. Gao, Q. Gao, and H. Wang, "An intrusion detection model based on deep belief networks," in *2014 Second International Conference on Advanced Cloud and Big Data*, Nov 2014, pp. 247–252.

Nathan Shone is a Lecturer in the Department of Computer Science at Liverpool John Moores University (LJMU) in the UK. He completed his PhD at LJMU in Network Security, focusing on misbehaviour detection in complex system-of-systems. His research interests include anomaly detection, misbehaviour monitoring, IoT security and security monitoring.

Tran Nguyen Ngoc is the Head of Department for Information Security at Le Quy Don Technical University in Vietnam. He received PhD in System analysis, control and information processing from Don State Technical University, Russia. His research interests focus on the pattern recognition, cyber security and artificial intelligence.

Vu Dinh Phai is a Researcher in the Department for Information Security at Le Quy Don Technical University in Vietnam. Since 2013, he has been involved in various research projects and teaching at LQDU. He received his Masters degree in Information Systems from LQDU in 2016. His research interests include network security, wireless security and machine learning.



Qi Shi is a Professor in Computer Security in the Department of Computer Science at Liverpool John Moores University (LJMU) in the UK. He received his PhD in Computing from the Dalian University of Technology, P.R. China. His research interests include security protocol deign, ubiquitous computing security, cloud security, sensor network security, computer forensics and intrusion detection.