IEICE TRANSACTIONS

on Information and Systems

VOL. E103-D NO. 7
JULY 2020

A PUBLICATION OF THE INFORMATION AND SYSTEMS SOCIETY
The Institute of Electronics, Information and Communication Engineers
Kikai-Shinko-Kaikan Bldg., 5-8, Shibakoen 3 chome, Minato-ku, TOKYO, 105-0011 JAPAN

INVITED PAPER *Special Section on Information and Communication System Security*

Intrusion Detection System Using Deep Learning and Its Application to Wi-Fi Network*

Kwangjo KIM[†a)], ***Member***

SUMMARY Deep learning is gaining more and more lots of attractions and better performance in implementing the Intrusion Detection System (IDS), especially for feature learning. This paper presents the state-of-the-art advances and challenges in IDS using deep learning models, which have been achieved the big performance enhancements in the field of computer vision, natural language processing, and image/audio processing than the traditional methods. After providing a systematic and methodical description of the latest developments in deep learning from the points of the deployed architectures and techniques, we suggest the pros-and-cons of all the deep learning-based IDS, and discuss the importance of deep learning models as feature learning approach. For this, the author has suggested the concept of the Deep-Feature Extraction and Selection (D-FES). By combining the stacked feature extraction and the weighted feature selection for D-FES, our experiment was verified to get the best performance of detection rate, 99.918% and false alarm rate, 0.012% to detect the impersonation attacks in Wi-Fi network which can be achieved better than the previous publications. Summary and further challenges are suggested as a concluding remark.

key words: *intrusion detection system, deep learning, feature learning, anomaly detection, deep-feature extraction and selection*

1. Introduction

Computer networks and the Internet are inseparable from human life today. Abundant applications rely on the Internet, including life-critical applications in healthcare, content sharing, military, *etc*. Moreover, extravagant financial transactions exist over the Internet every day. This rapid growth of the Internet has led to a significant increase in wireless network traffic in recent years. An Intrusion Detection System (IDS) plays as one of the most important roles in providing the security infrastructure to all kinds of the network [2] including wireless networks [3]. Machine Learning (ML) has been well adopted as the primary detection algorithm in IDS owing to their model-free properties and learnability [4]. Leveraging the recent development of ML such as Deep Learning (DL) [5] can be expected to bring significant benefits for improving existing IDS.

Some improvements in IDS could be achieved by embracing a breakthrough in ML [4], in particular DL of which applications have won numerous contests in pattern recognition, *etc.* [6]. DL belongs to a class of ML methods, where employs the consecutive layers of information-processing stages in hierarchical manners for pattern classification and feature or representation learning [7]. According to [8], there are three important reasons why DL has been becoming prominent recently.– First, processing abilities (*e.g.*, GPU) increased sharply. Second, computing hardware getting affordable, and last, a breakthrough in ML research.

Shallow and Deep Learners are distinguished by the depth of their credit assignment paths, which are chains of possibly learnable, causal links between actions and effects. Usually, DL plays an important role in image classification and is also commonly used for language, graphical modeling, pattern recognition, speech, audio, image, video, natural language and signal processing [7]. There are many DL methods such as Deep Belief Network (DBN), Boltzman Machine (BM), Restricted Boltzman Machine (RBM), Deep Boltzman Machine (DBM), Deep Neural Network (DNN), Auto Encoder, Deep / Stacked Auto Encoder (SAE), Stacked denoising Auto Encoder, Distributed representation and Convolutional Neural Network (CNN). The advancements in learning algorithms might improve IDS to obtain higher detection rate and lower false alarm rate.

On the other hand, the broad and rapid spread of computing devices using the Internet, especially Wi-Fi networks create complex, large, and high-dimensional data, which makes us difficult to countermeasure the attacks properly. Feature learning acts as an essential tool for improving the learning process of a machine-learning model. It consists of feature construction, extraction, and selection. Feature construction expands the original features to enhance their expressiveness, whereas feature extraction transforms the original features into a new form and feature selection eliminates unnecessary features [9]. Feature learning is a key to improve the performance of existing ML-based IDS.

There are various approaches of how to adopt DL in IDS applications. Several researches use DL methods in a partial sense only, while the rest still uses conventional neural networks. Also, DL method requires a lot of time to train correctly. However, some researchers have adopted DL method in implementing their IDS to achieve better performance despite of the training overhead.

Tran *et al.* [10] provided an example how to use DL in IDS. Classical ML algorithms, such as Naive Bayes and C4.5, assisted by high-level features, were generated using genetic programming and implemented in IDS. This approach is a common way of leveraging DL models in IDS, where the DL models can improve any classical ML with

Manuscript received November 27, 2019.
Manuscript revised January 21, 2020.
Manuscript publicized March 31, 2020.

[†]The author is with Korea Advanced Institute of Science and Technology (KAIST), Daejeon 34141, South Korea.

*This is an abstracted paper on a book [1] whose copyright is owned by the author.

a) E-mail: kkj@kaist.ac.kr

DOI: 10.1587/transinf.2019ICI0001

high-level features. This approach was also adopted by Aminanto *et al.* [11]. Hamed *et al.* [12] surveyed several pre-processing techniques in IDS researches, how to collect data from real world and honeypot and how to build a dataset from raw input data. Although most of IDS use DL models as their data pre-processing technique, their work focuses on reviewing DL-based IDS.

We have examined feature extraction using SAE, which can reduce the complexity of original features of the dataset. However, besides a feature extractor, SAE can also be used for classifying and clustering tasks. Aminanto *et al.* [13] used semi-supervised approach for IDS which contains feature extractor (unsupervised learning) and classifier (supervised learning). SAE was leveraged for feature extraction and regression layer with softmax function for the classifier. SAE as feature extractor was also used in [11], but Artificial Neural Network (ANN), Decision Tree (DT), and Support Vector Machine (SVM) were leveraged as feature selection. In other words, stacked feature extraction and weighted feature selections can be combined together. Their experiments [11] improved the feature learning process by combining stacked feature extraction with weighted feature selection. The feature extraction of SAE is capable of transforming the original features into a more meaningful representation by reconstructing its input and providing a way to check that the relevant information in the data has been captured. SAE can be efficiently used for unsupervised learning on a complex dataset.

Aminanto and Kim [14], [15] used SAE for other roles than a feature extractor, such as for classifying and clustering. ANN was adopted as feature selection since the weight from the trained models mimics the significance of the corresponding input [14]. By selecting the important features only, the training process becomes lighter and faster than before. In particular, Aminanto and Kim [14] exploited SAE as a classifier since this employs consecutive layers of processing stages in hierarchical manners for pattern classification and feature or representation learning. On the other hand, Aminanto and Kim [15] proposed a novel fully unsupervised method which can detect attacks without prior information on the data label. The scheme is equipped with an unsupervised SAE for extracting features, and a K-means clustering algorithm for clustering task. Detailed discussions will be suggested in Sect. 5.

The remainder of this paper is organized as follows: Sect. 2 introduces the overall of IDS including its history and classification based on the detection method. We re-examine the classification of DL methods depending on the intention of architectures and techniques in Sect. 3. In Sect. 4, we discuss about state-of-the-art of DL-based IDS by surveying the latest publications. Section 5 discusses two novel models which leverage deep learning as a feature learning approach to achieve the best performance of detecting the impersonation attacks using AWID dataset in Wi-Fi network. Finally, Sect. 6 provides the conclusion and future work.

2. Overview of IDS

An IDS becomes a popular security measure in computer networks. Unlike Firewall (FW), IDS usually located inside the network to monitor all incoming traffics. One may consider using both FW and IDS to protect the network efficiently. IDS is defined as automation of intrusion detection process of finding events of violation of security policies or standard security practices in computer networks [16]. Besides identifying the security incidents, IDS also has other functions: documenting existing threats and deterring adversaries. IDS requires particular properties which acts as a passive countermeasure, monitors whole or part of networks only and aims high attack detection and low false alarm rates.

2.1 History

IDS was firstly introduced in 1980 by Anderson, an information security expert. In his report, titled "Computer Security Threat Monitoring and Surveillance [17]," he proposed a system that is recognized as the pioneer of automated IDS. The system is based on rule-based detection approach that scans network traffic to detect malicious activities. Anderson's idea is considered as the foundation of IDS development; however, there is a limitation in detecting zero-day attack in his work. This limitation is a common weakness of rule-based IDS. Four years later, Denning published "An Intrusion Detection Model [18]," which introduced the model of commercial IDS. The growth of IDS continued during 1990's. At the early of 2000's, anomaly detection-based IDS was developed, but the performance was not so good because of high false positive. There was also no environment that IDS was really deployed. As a result, IDS seems to be on the path of undeployment. However, during 2010s, when big data and cloud computing have become to be popular, IDS received lots of attractions as a border control for the security of a network. Moreover, the development of machine learning helped IDS to get better performance. Due to these, IDS has arisen from brink of disappearance and became more important than ever. Nowadays, IDS plays an important role in big data security and still continues to grow in positive progression.

2.2 Classification

We can divide IDS depending on their placement and the methodology deployed in a target network. By the positioning of the IDS module in the network, we might distinguish IDS into three classes: network-based, host-based and hybrid IDS. The first IDS, network-based IDS puts the IDS module inside the network where the whole traffics can be monitored. This IDS checks for malicious activities by inspecting all packets moving across the network. On the other hand, the host-based IDS which places the IDS module on each client of the network. The module examining

all inbound and outbound traffics of the corresponding client leads to detail monitoring of the particular client. Two types of IDS have specific drawbacks– the network-based IDS might burden of the workload then miss some malicious activities, while the host-based IDS does not monitor all the network traffics but having less workload than the network-based IDS. Therefore, the hybrid IDS places IDS modules in the network as well as clients to monitor both specific clients and network activities at the same time.

Based on the detection method, IDS can be divided into three typical types: misuse, anomaly, and specification-based IDS. A misuse-based IDS, known as a signature-based IDS [19], looks for any malicious activities by matching the known signatures or patterns of attacks with the monitored traffics. This IDS suits for known-attack detection; however, new or unknown attacks (also called as zero-day exploits) are difficult to be detected. An anomaly-based IDS detects an attack by profiling normal behavior and then triggers an alarm if there is any deviation from it. The strength of this IDS is its ability for unknown attack detection. Misuse-based IDS usually achieves higher detection performance for known attacks than anomaly-based IDS. A specification-based IDS manually defines a set of rules and constraints to express the normal operations. Any deviation from the rules and constraints during execution is flagged as malicious [20]. Table 1 summarizes the comparison of IDS types based on their methodology.

Table 1 Comparison of IDS types

Type	Misuse-based	Anomaly-based	Specification-based
Method	Identify known attack patterns	Identify unusual activity patterns	Identify violation of pre-defined rules
Detection Rate	High	Low	High
False Alarm Rate	Low	High	Low
Unknown Attack Detection	Incapable	Capable	Incapable
Drawback	Updating signatures is burdensome	Computing any ML is heavy	Relying on expert knowledge during defining rules is undesirable

Table 2 Comparison between supervised and unsupervised learnings

	Supervised	Unsupervised
Definition	The dataset are labeled with pre-defined classes	The dataset are labeled **without** pre-defined classes
Approach	Classification	Clustering
Method	Support Vector Machine, Decision Tree, *etc*	K-means clustering, Ant Clustering Algorithm, *etc*
Known Attack Detection	High	Low
Unknown Attack Detection	Low	High

We discuss further the machine learning-based IDS which belongs to a kind of the anomaly-based IDS [21]. There are two types of learning – supervised and unsupervised learnings. The unsupervised learning does not require a labeled dataset for training which is crucial for colossal network traffics recently, while the supervised learning requires a labeled dataset. Unsupervised learning capability is of critical significance as it allows a model to detect new attacks without creating costly labels or dependent variables. Table 2 outlines a comparison between supervised and unsupervised learnings.

3. Deep Learning Methods in Brief

DL originally comes from the advancements of Neural Network (NN) algorithm. Various methods have been applied in order to overcome the limitations of one hidden layer only in NN. Those methods employ consecutive hidden layers with hierarchically cascaded connection. Due to a variety of models belonging to DL, Aminanto and Kim [22] extended to classify several DL models based on Deng [7], [8] which differentiates DL into three sub-groups, generative, discriminative and hybrid. The classification is based on the intention of architectures and techniques, *e.g.*, synthesis/generation or recognition/classification. The classification of DL methods is illustrated in Fig. 1.

3.1 Generative Model

Generative model or so-called unsupervised learning uses the unlabeled data. The main concept of applying generative architectures to pattern recognition is unsupervised learning or pre-training [7]. Since the deeper the neural network becomes, the more complicated the learning process is required. Thus, with the limited training data, learning at each layer in layer-by-layer approach without relying on all the previous layers is essential. There are a number of methods that classified as the unsupervised learning.

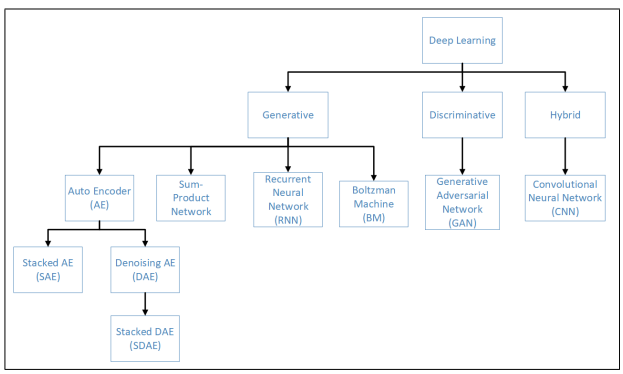

Fig. 1 Classification of DL methods

3.1.1 Stacked (Sparse) Auto Encoder

(Sparse) Auto Encoder (AE) can be used as DL technique by an unsupervised greedy layer-wise pre-training algorithm known as Stacked (Sparse) Auto Encoder (SAE). Here, pre-training refers to the training of a single AE using a single hidden layer. Each AE is trained separately before being cascaded afterward. This pre-training phase is required to construct a stacked AE. In this algorithm, all layers except the output layer are initialized in a multi-layer neural network. Each layer is then trained in an unsupervised manner as an AE, which constructs new representations of the input.

SAE is trained with the same neuron number of both input and output layers. Meanwhile, the nodes in the hidden layer are smaller than the input which represents a new less-feature set. This architecture leads to an ability that can reconstruct the data after complicated computations. AE aims to learn a compact set of data efficiently and can be stacked to build a deep network. Training results of each hidden layer are cascaded, which can provide new transformed features by different depths. To train more precisely, we can append an additional classifier layer with class labels [23]. The labels will act as a semi-supervised feature learner when we can have a large amount of tagged training samples. As a result, the additional layer will improve the accuracy during the learning process. Besides, a Denoising Auto Encoder (DAE) is trained to reconstruct a precise correction input from the corrupted input by noise [24], which may also be stacked to build deep networks as well.

3.1.2 Boltzman Machine

Boltzman Machine (BM) is a network of binary units that is symmetrically paired [25], which means all input nodes are linked to all hidden nodes. BM is a shallow model with one hidden layer only. BM has a structure of neuron units that make stochastic decisions about whether active or not [8]. If one BM output is cascaded into multiple BMs, they are called Deep BM (DBM). Meanwhile, Restricted Boltzmann machine (RBM) is a customized BM without connection among the input and hidden nodes. RBM consists of visible and hidden variables such that their relations can be figured out. *Visible* means neurons in input for training data. If multiple layers of RBM are stacked, a layer-by-layer scheme called Deep Belief Network (DBN). DBN could be used as a feature extraction method for dimensionality reduction when unlabeled dataset and back-propagation are used (which means unsupervised training). In contrast, DBN can be used for classification when appropriately labeled dataset with feature vectors are used (which means supervised training) [26].

3.1.3 Sum-Product Networks

Other deep generative model is Sum-Product Networks (SPN), which is a directed acyclic graph with variables as leaves, sum and product operations as internal nodes, and weighted edges [27]. The sum nodes provide mixture models while the product nodes express the feature hierarchy [8]. Therefore, we can consider SPN as a combination of mixture models and feature hierarchies.

3.1.4 Recurrent Neural Network

Recurrent Neural Network (RNN) is an extension of neural networks with cyclic links to process sequential information. This cyclic links placed between higher and lower layer neurons which enable RNN to propagate data from previous to current events. This property makes RNN having a memory of time series events [28].

One advantage of RNN is capable of connecting the previous information to present task; however, it cannot reach "far" previous memory. This problem is commonly known as long-term dependencies. Long-Short Term Memory Networks (LSTM) introduced by Hochreiter and Schmidhuber [29] to overcome this problem. LSTMs are an extension of RNN with four neural networks simplified as LSTM-RNN here in a single layer, where RNN have one only.

The main advantage of LSTM is the existence of state cell which is the line passing through in the top of every layer. The cell accounts for propagating information from the previous layer to the next one. Then, "gates" in LSTM would manage which information will be passed or dropped. There are three gates to control the information flow, namely input, forget and output gates [30]. These gates are composed of a sigmoid neural network.

3.2 Discriminative Model

Discriminative model or supervised learning is intended to distinguish some parts of data for pattern classification with labeled data [7]. An example of the discriminative architecture is Convolutional Neural Network (CNN) which employs a special architecture particularly suitable for image recognition. The main advantage of CNN is no need of hand-crafted feature extraction. CNN can train multilayer networks with gradient descent to learn complex, high-dimensional, nonlinear mappings from large collections of data [31] by using three basic concepts: local receptive fields, shared weights, and pooling [32].

RNN can also be considered as a discriminative model when the output of RNN used as label sequences for the input [8]. One example of this network was proposed by Graves [33] who leveraged RNNs to build a probabilistic sequence transduction system, which can transform any input sequence into any finite, discrete output sequence.

3.3 Hybrid Model

The hybrid deep architecture combines both generative and discriminative architectures. The hybrid structure aims to

distinguish data as well as discriminative approach. However, in the early step, it has assisted in a significant way with the generative architectures results. An example of hybrid architecture is Deep Neural Network (DNN) [7]. However, confusion between DNN and DBN happens. In the open literature, DBN also uses backpropagation discriminative training as a "fine-tuning." This concept of DBN looks similar to DNN. According to Deng [8], DNN is defined as a multilayer network with cascaded fully connected hidden layers and is often use stacked RBM as a pre-training phase. Many other generative models can be considered as discriminative or hybrid models when classification task added with class labels.

Goodfellow *et al.* [34] introduced a novel framework which trains both generative and discriminative models at the same time, which the generative model G captures the data distribution and the discriminative model D distinguishes the original input data and the data coming from the model G. It is a zero-sum game of G and D models [35] that model G aims to counterfeit the original input data while model D aims to discriminate the original input and output of model G. According to Dimokranitou [35], the advantage of Generative Adversarial Network (GAN) is consistent in a sense that after equilibrium is achieved, GAN can be trained with missing or limited data without using approximate inference or Markov chain. On the other hand, the disadvantage of applying GAN must find the equilibrium between G and D models.

4. State-of-the-Art Deep Learning for IDS

This section reviews the state-of-the-art IDS leveraging DL models in the open literature from the year, 2016 to 2019. The critical issues like problem domain, methodology, dataset and experimental result of each publication will be discussed. All publications can be classified into three different categories according to DL classification as suggested in the previous section, namely generative, discriminative and hybrid models. The generative model consists of IDS that use DL models for feature extraction only and use shallow methods for the classification task. The discriminative model contains IDS that use a single DL method for both feature extraction and classification task. While the hybrid model includes IDS that use more than one DL methods for the sake of generative and discriminative IDS. Here, all IDS are compared to understand the advancement of DL in IDS researches.

4.1 Generative Model

We describe the latest researches on IDS under the generative model which uses DL for feature extraction only and use shallow methods for the classification task.

4.1.1 DNN as Generative

Roy *et al.* [36] proposed an IDS by leveraging DL models and validated that a DL approach can improve IDS performance. DNN is selected comprising of multilayer feedforward neural network with 400 hidden layers. Shallow models, rectifier and softmax activation functions, are used in the output layer. The advantages of feedforward neural network are to provide the capabilities for the precise approximation in a complex multivariate nonlinear function directly from the input values and for the robust modeling capabilities in a large class. Besides that, the authors claimed that DNN is better than DBN since the discriminating ability can suit pattern classification well by characterizing the posterior distributions of classes.

For the sake of validation, KDD Cup'99 [37] dataset was used. This dataset has 41 features that become the input to the network. The authors divided training data into 75% for training and 25% for validation. They also compared the performance of a shallow classifier, SVM. Based on their experiments, DNN outperforms SVM by the accuracy of 99.994%, while SVM achieved 84.635% only. This result shows the effectiveness of DNN for IDS.

4.1.2 Accelerated DNN as Generative

Another DNN but different architecture was proposed by Potluri and Diedrich [38] in 2016. Their work mainly focuses on improving DNN implementation for IDS by using multi-core CPUs and GPUs. This is important since DNN requires large computation for training [39]. They adapted SAE to construct the DNN in this approach. The architecture of this network has 41 input features from NSL-KDD [40] dataset, 20 neurons in the first hidden layer by the first AE, ten neurons in the second hidden layer by the second AE, and five neurons in the output layer containing softmax activation function. In the training phase, each AE is trained separately but in sequence since the hidden layer of the first AE becomes the input of second AEs. There are two steps for fine-tuning processes, the first by softmax activation function and the second by backpropagation through the entire network.

NSL-KDD dataset is a revised version of KDD Cup'99 dataset. It has the same number of features which is 41 but with more rational distributions and without redundant instances that appear in KDD Cup'99 dataset. The authors firstly tested the network with different attack class combinations from 2 to 4 classes. The smaller number of attack classes performs better than the higher number of attack classes as expected since the imbalance of the class distribution leads to a good result for the fewer attack types. Their experiments showed that the training using parallel CPU achieved three times faster than serial CPU. Unfortunately, the authors do not provide performance comparison regarding detection accuracy or false alarm rate.

4.1.3 Self-Taught Learning as Generative

Self-Taught Learning (STL) was proposed as a DL model for IDS by Javaid *et al.* [41]. The authors mentioned two

challenges to develop an efficient IDS. The first challenge is to select feature since the selected features for a particular attack might different for other attack types. The second challenge is to deal with the limited amounts of a labeled dataset for the training purpose. Therefore, a generative DL model was chosen in order to deal with this unlabeled dataset. STL consists of two stages – Unsupervised Feature Learning (UFL) and Supervised Feature Learning (SFL). For UFL, the authors leveraged sparse AE while softmax regression for SFL. The UFL accounts for feature extraction with unlabeled dataset while the SFL accounts for classification task with the labeled data.

The authors verified their approach using NSL-KDD dataset. Before the training process, the authors defined a pre-processing step for the dataset which contains 1-to-N encoding and min-max normalization. After 1-to-N encoding process, 121 features were ready for normalization step and input features for the UFL. 10-fold cross-validation and test dataset from NSL-KDD dataset were selected for the training and the testing, respectively. The authors also evaluated the STL for three different attack combinations, 2-class, 5-class, and 23-class. In general, their STL achieved higher than 98% of classification accuracy for all combinations during the training phase. In the testing phase, the STL achieved an accuracy of 88.39% and 79.10% for 2-class and 5-class classifications, respectively. They suggested that the future work is to develop real-time IDS using DL models and an IDS with on-the-go feature learning on the raw network traffic.

4.1.4 Stacked DAE as Generative

Yu *et al.* [42] introduced a session-based IDS using a DL architecture. They came up with common IDS shortcomings: high false positive and false negative, most attack features in common dataset are heavily structured and have special semantics involved in specific expert knowledge, and the heavily hand-crafted dataset is closely related to particular attack classes. Therefore, a DL model was leveraged since the unsupervised DL can learn the essential features automatically from the large data. Their approach consists of extracting features from the raw data and applying the unsupervised Stacked Denoising AE (SDAE) that itself contains two hidden layers and a softmax regression layer. For the denoising purpose, the authors randomly set the input features using zero value for 10%, 20% and 30% of the input features. The authors claimed that the advantages of using SDAE is to improve the learning capability of the important features from the unlabeled instances automatically, by adding robust denoising strategy from the missing and noisy input, and having the better dimensional reduction if the hidden layer is non-linear.

They measured accuracy, precision, recall, F-score and Receiver Operating Characteristic (ROC) curve as the performance metrics. Binary and multi-class classifications were used along with 43% of the dataset and whole dataset combinations to verify the performance of the SDAE, which is also compared to other DL models, namely SAE, DBN and AE-CNN models. In overall, the SDAE achieved the best performance with the highest accuracy rate of 98.11% of multi-class classification using the whole dataset.

4.1.5 LSTM-RNN as Generative

Kim *et al.* [30] adopted the generative approach of LSTM-RNN for IDS. They leveraged softmax regression layer as the output layer. While other hyper-parameters are 100 time step, 50 batch size and 500 epoch. Also, Stochastic Gradient Descent (SGD) and Mean Square Error (MSE) are used as the optimizer and loss function, respectively. The 41 input features were drawn from KDD Cup'99 dataset. Their experiments showed that if the best learning rate is 0.01 and the hidden layer size is 80, they can achieve 98.88% of the detection rate and 10.04% of the false alarm rate. Similar network topology also proposed by Liu *et al.* [43] with different hyper-parameters: 10 time step, 100 batch size, and 500 epoch. Using KDD Cup'99 dataset, they got 98.3% of detection rate and 5.58% of false alarm rate.

4.2 Discriminative Model

We describe the latest researches on IDS under the discriminative model which uses a single DL method for both feature extraction and classification task.

4.2.1 DNN as Discriminative

Software Defined Networks (SDN) is an emerging network technology of current applications since it has a unique property that can be constructed by the controller plane and the data plane. The controller plane decouples the network control and forwarding functions. The centralized approach of controller plane makes SDN controller suitable for IDS function due to the whole network captured by the controller. Unfortunately, due to the separation of control and data planes, it leads to some critical threats. Tang *et al.* [44] proposed a DNN approach for IDS in SDN context. The DNN architecture is 6-12-6-3-2 which means six input features, three hidden layers with twelve, six and three neurons for each layer and two classes output.

They used NSL-KDD dataset to verify their approach. Since the dataset has 41 features, the authors selected 6 features that are fundamental features in SDN based on their expertise. They measured accuracy, precision, recall, F-score and ROC curve as the performance metrics. Based on the experimental result, the learning rate of 0.001 is the best hyper-parameter since the learning rate of 0.0001 already over-fitted. Their approach achieved 75.75% of accuracy, which is lower than other methods using whole 41 features but higher than other methods using 6 features only. From this fact, the authors claimed that the proposed DNN can be generalized to abstract the characteristics of network traffic with the limited features alone.

4.2.2 RNN as Discriminative

Yin *et al.* [45] highlighted the advantages of RNN implementation for improving the performance of IDS. RNN contains forward and backward propagation, where the latter is the same neural network which computes the residual of forwarding propagation. The proposed RNN-IDS begins with data pre-processing step which comprises of numericalization and normalization. Feature-ready data can be propagated into the training step of RNN. The output model from the training is used as testing step for the test dataset.

For the experiment, they used NSL-KDD dataset, for both training and testing. The original features are 41 features but became 122 features after numericalization which maps string to binary. Two types of classification were tested, namely binary and multi-class classifications. Based on the experimental results, the best hyper-parameter during the binary classification is the learning rate of 0.1, an epoch of 100 and the hidden nodes of 80 with an accuracy of 83.28% (using KDDTest$^+$) [46]. Meanwhile, during the multi-class classification, the best hyper-parameter is learning rate of 0.5 and the hidden nodes of 80 with an accuracy of 81.29%. The RNN-IDS outperformed other ML methodologies tested by the authors for both binary and multi-class classification.

4.2.3 CNN as Discriminative

Li *et al.* [47] experimented using CNN as the feature extractor and classifier for IDS. CNN achieved many successful implementations in image-related classification tasks, however, still a big challenge for text classification. Therefore, the main challenge of implementing CNN in IDS context is the image conversion step, which is proposed by Li *et al.* [47]. NSL-KDD dataset was used for the experiment. The image conversion step begins by mapping of 41 original features into 464 binary vectors. The mapping step comprises of two types mapping, one hot encoding and one hot encoder with 10 binary vectors for symbolic and continuous features, respectively. The image conversion step continues with converting 464 vectors into 8×8 pixel images. These images are ready for the training input of CNN. The authors decided to experiment with the learned CNN models, ResNet 50 and GoogLeNet. Their experimental results on KDDTest$^+$ show the accuracy of 79.14% and 77.14% using ResNet 50 and GoogLeNet, respectively. Although this result does not improve the state-of-the-art performance of IDS, this work demonstrated how to apply CNN with image conversion in IDS context.

4.2.4 LSTM-RNN as Discriminative

LSTM-RNN became more popular due to its successful applications in various research areas. Considering the previous events can be applied to increase the better performance of IDS.

Staudemeyer [28] experimented various network topologies of LSTM-RNN for network traffic modeling as a time series. Training data were extracted from KDD Cup'99 dataset. The author also selected the subset of salient features by using decision tree algorithm and compared to the whole and subset features performance in the experiment. Based on the experimental results, the best performance was achieved by four memory blocks containing two cells, with forget gates and shortcut connections, 0.1 of learning rate and up to 1,000 epochs. The overall accuracy is 93.82%. They also mentioned in the conclusion that LSTM-RNN is suitable for classifying attacks with a big number of records and poor for a limited number of attack instances.

Bontemps *et al.* [48] leveraged LSTM-RNN in IDS for two objectives: a time series anomaly detector and collective anomaly detector by proposing a circular array. Collective anomaly itself is a collection of related anomalous data instances concerning the whole dataset [48]. They used KDD Cup'99 dataset for their experiment and explained the preprocessing steps needed to build a time series dataset from KDD Cup'99 dataset.

Putchala [49] implemented a simplified form LSTM, called Gated Recurrent Unit (GRU) in IoT environments. GRU is suitable for IoT due to its simplicity which caused by reducing a number of gates in the network. GRU merges both forget and input gate to an update gate and combines the hidden and cell state to become a simple structure.

The author then adopted a multi-layer GRU, which is GRU cells used in each hidden layer of RNN and feature selection also done by using random forest algorithm. Their experiments were conducted using KDD Cup'99 dataset and achieved 98.91% and 0.76% of accuracy and false alarm rate, respectively.

Bediako [50] proposed a Distributed Denial of Service detector using LSTM-RNN. They checked the performance of LSTM-RNN using both CPU and GPU. NSL-KDD dataset was used for experiments. The notable detection accuracy is 99.968%.

4.3 Adversarial Networks as Hybrid

We introduce an IDS in brief that uses more than one DL model for both generative and discriminative purposes. Dimokranitou *et al.* [35] proposed an abnormal events detector in images using an adversarial networks. Although the detector was not an IDS, it has the same objective to detect anomalies. They implemented an adversarial autoencoder which combines autoencoders and GAN. The network attempts to match the aggregated posterior of the hidden code vector of AE, with an arbitrary prior distribution. The reconstruction error of learned AE is low for normal events and high for irregular events.

4.4 Deep Reinforcement Learning as Hybrid

Choi and Cho [51] proposed an adaptive IDS for database applications using an Evolutionary Reinforcement Learn-

Table 3 Model comparisons on KDD cup'99 dataset

Model	*Feature Extractor*	*Classifier*	*Accuracy* (%)
DNN [36]	FF-NN	Softmax	99.994
LSTM-RNN-K [30]	LSTM-RNN	Softmax	96.930
LSTM-RNN-L [43]	LSTM-RNN	Softmax	98.110
LSTM-RNN-S [28]	LSTM-RNN	LSTM-RNN	93.820
GRU [49]	GRU	GRU	98.920

ing (ERL), which combines evolutionary learning and reinforcement learning used for the learning process of a population and an individual, respectively. The approach comprises of two Multilayer Perceptrons (MLPs): a behavior and an evaluation network. Since the evaluation network used for evolutionary learning, the network evolves to explore the optimal model. Their experiments were done using a particular scenario, called TPC-E, which is an online transaction processing workload of a brokerage company. A 90% of classification accuracy was achieved after 25 generations.

Feng and Xu [52] concerned about detecting unknown attacks in Cyber Physical System (CPS). Then a novel deep RL based optimal strategy was proposed. The novelty came as an explicit cyber state dependent dynamics and a model of the zero-sum game to solve the Hamiltonian-Jacobi-Isaac (HJI) equation. A deep Reinforcement Learning (RL) algorithm with game theoretical actor structure was developed to address the HJI equation. The deep RL network consists of 3 multi-layer NNs; the first is used in critic part, the second is used to approximate the possible worst attack policy and the last is used to estimate the optimal defense policy in real-time. Firstly, the critic-NN solves the HJI equation. Then, the computation result is used by the Worst-Attack-Policy-NN to approximate the possible attack. After that, the solution from the critic-NN and the Worst-Attack-Policy-NN become the input of Optimal-Defense-Policy NN to approximate the worst cyber attack policy. In order to improve the accuracy, online back propagation method is used by approximating the cost function on Critic-NN. HJI equation is used to train the NN weight and find the most optimum weight with smallest error.

4.5 Comparison of State-of-the-Art Methods

We compare and summarize all the previous publications based on KDD Cup'99 and NSL-KDD datasets in Tables 3 and 4, respectively. The performance of IDS on KDD Cup'99 are promising, as expected, more than 90% of accuracy. Three IDS in Table 3 are using LSTM-RNN approach which means that a time series analysis is suitable for distinguishing benign and anomalies in network traffic.

Even more, GRU [49] demonstrated that a lightweight DL model is possible to be implemented in IoT environments which is crucial for low-power applications. There is still a space for improvement when we are using NSL-KDD dataset as shown in Table 4. The most accurate model

Table 4 Model comparisons on NSL-KDD dataset

Model	*Feature Extractor*	*Classifier*	*Accuracy* (%)
STL [41]	AE	Softmax	79.10
DNN-SDN [44]	NN	NN	75.75
RNN [45]	RNN	RNN	81.29
CNN [47]	CNN	CNN	79.14

is RNN [45] with 81.29% of accuracy. Again, this fact infers that a time series analysis may improve IDS performance. Although IDS using CNN achieved not the best performance, it is noticed that by applying a proper text-to-image conversion, we may benefit full potential of CNN as already shown in image recognition researches.

5. Deep Feature Learning for IDS in Wi-Fi Networks

Feature Learning (FL) is a technique that models the behavior of data from a subset of attributes only. It also shows the correlation between detection performance and traffic model quality efficiently [53]. One advantage of deep learning models is to process the underlying data from the input which suits for the task of FL. Therefore, we discuss this critical role of deep learning in IDS as (i) deep feature extraction and (ii) deep learning for assisted clustering.

5.1 Wi-Fi Dataset

In Wi-Fi network area, there is a dataset so-called Aegean Wi-Fi Intrusion Dataset (AWID) developed by Kolias *et al.* [54]. There are two types of AWID dataset. The first type named "CLS", has four target classes, whereas the second, named "ATK", has 16 target classes. The 16 classes of the "ATK" dataset belong to the four attack categories in the "CLS" dataset. As an example, the *Caffe-Latte*, *Hirte*, *Honeypot* and *EvilTwin* attack types listed in the "ATK" dataset, are categorized as an impersonation attack in the "CLS" dataset. Based on the size of the data instances included, the AWID dataset comprises both full and reduced versions. There are 1,795,595 instances in the reduced training dataset, with 1,633,190 and 162,385 normal and attack instances, respectively. There are 575,643 instances in the reduced test dataset, with 530,785 and 44,858 normal and attack instances, respectively.

The data contained in the AWID dataset are diverse in value, discrete, continuous, and symbolic, with a flexible value range. These data characteristics could make it difficult for the classifiers to learn the underlying patterns correctly [55]. The target classes were mapped to one of these integer-valued classes: 1 for normal instances, 2 for an impersonation, 3 for flooding, and 4 for an injection attack. Meanwhile, symbolic attributes such as a receiver, destination, transmitter, and source address were mapped to integer values with a minimum value of 1 and a maximum value, which is the number of all symbols. The pre-processing phase thus includes mapping symbolic valued attributes to numeric values, according to the normalization steps and

Table 5 Distribution of each class for both balanced and unbalanced dataset

Class		Training	Test
Normal	Unbalanced	1,633,190	530,785
	Balanced	163,319	53,078
Attack	Impersonation	48,522	20,079
	Flooding	48,484	8,097
	Injection	65,379	16,682
	Total	162,385	44,858

AWID dataset mimics the natural unbalanced network distribution between normal and attack instances. "Balanced" means to make equal distribution between the number of normal instances (163,319) and total attack instances (162,385). 15% of training data were withdrawn for validation data.

dataset-balancing process described in Algorithm 1.

Algorithm 1 Dataset Pre-processing Function

```
function DATASET PRE-PROCESSING(Raw Dataset)
    function DATASET NORMALIZATION(Raw Dataset)
        for each data instance do
            cast into integer value
            normalize (Eq. (1))
            NormalizedDataset
        end for
    end function
    function DATASET BALANCING(NormalizedDataset)
        Pick 10% of normal instances randomly
        BalancedDataset
    end function
    InputDataset ← BalancedDataset
    return InputDataset
end function
```

Some dataset attributes such as the WEP Initialization Vector (IV) and Integrity Check Value (ICV) were hexadecimal data, which need to be transformed into integer values as well. The continuous data such as the timestamps were also left for the normalization step. Some of the attributes have question marks, ?, to indicate unavailable values. One alternative was selected in which the question mark was assigned to a constant zero value [56]. After all, data were transformed into numerical values; attribute normalization is needed [57]. Data normalization is a process; hence, all value ranges of each attribute were equal. The mean range method [58] was adopted in which each data item is linearly normalized between zero and one to avoid the undue influence of different scales [56]. Equation (1) shows the normalizing formula.

$$z_i = \frac{x_i - min(x)}{max(x) - min(x)}, \tag{1}$$

where z_i denotes the normalized value, x_i refers to the corresponding attribute value and $min(x)$ and $max(x)$ are the minimum and maximum values of the attribute, respectively. The class distribution in the dataset is shown in Table 5.

The ratio between the normal and attack instances is 10:1 for both unbalanced training and the test dataset. The reduced "CLS" data are a good representation of a real network, in which normal instances significantly outnumber attack instances. This property might be biased to the training model and affect the model performance [59], [60]. To alleviate this, the dataset was balanced by selecting 10% of the normal instances randomly. However, a specific value was set as the seed of the random number generator for reproducibility. The ratio between normal and attack instances became 1:1, which is an appropriate proportion for the training phase [60].

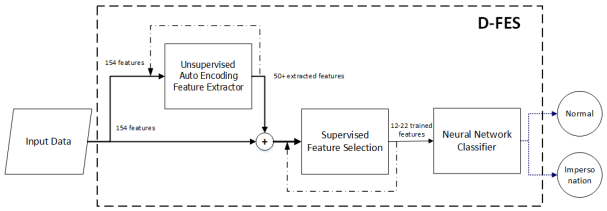

Fig. 2 Stepwise procedure of D-FES with two target classes: normal and impersonation attack

5.2 Deep Feature Extraction

The recent advances in mobile technologies have resulted in IoT-enabled devices becoming more pervasive and integrated into our daily lives. The security challenges that need to be overcome mainly stem from the open nature of a wireless medium such as a Wi-Fi network. An impersonation attack is an attack in which an adversary is disguised as a legitimate party in a system or communications protocol. The connected devices are pervasive, generating high-dimensional data on a large scale, which complicates simultaneous detections. Feature learning, however, can circumvent the potential problems that could be caused by the large-volume nature of network data. Aminanto *et al.* [11] presented a novel Deep-Feature Extraction and Selection (D-FES), which combines stacked feature extraction and weighted feature selection. The stacked autoencoding is capable of providing representations that are more meaningful by reconstructing the relevant information from its raw inputs. These representations were then combined with modified weighted feature selection inspired by an existing shallow-structured machine learner. D-FES empowers machine learning algorithm to perform the feature learning. It is shown that the ability of the condensed set of features to reduce the bias of a machine learner model as well as the computational complexity. Feature extraction and selection techniques are adopted in D-FES. Figure 2 shows the stepwise procedure of D-FES with two target classes.

A pre-processing procedure, which comprises the normalization and balancing steps, is necessary. D-FES starts by constructing SAE-based feature extractor with two consecutive hidden layers to optimize the learning capability and the execution time [61]. The SAE outputs 50 extracted features, which are then combined with the 154 original features existing in the AWID dataset [54]. Weighted feature

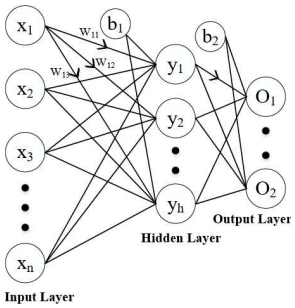

Fig. 3 ANN network with one hidden layer only

selection methods were then utilized using well-referenced machine learners including SVM, ANN, and C4.5 to construct the candidate models, namely D-FES-SVM, D-FES-ANN, and D-FES-C4.5, respectively. SVM separates the classes using a support vector (hyperplane). Then, ANN optimizes the parameters related to hidden layers that minimize the classifying error concerning the training data, whereas C4.5 adopts a hierarchical decision scheme such as a tree to distinguish each feature [62]. The final step of the detection task involves learning an ANN classifier with 12–22 trained features only. Figure 3 shows an ANN network with one hidden layer only where b_1 and b_2 represent the bias values for the corresponding hidden and output layer, respectively.

To select the essential features, the weight values between the first two layers were considered. The weight represents the contribution from the input features to the first hidden layer. A w_{ij} value close to zero means that the corresponding input feature x_j is meaningless for further propagation, thus having one hidden layer is sufficient for this particular task. The important value of each input feature is shown in Eq. (2).

$$V_j = \sum_{i=1}^{h} |w_{ij}|, \tag{2}$$

where h is the number of neurons in the first hidden layer. The feature selection process involves selecting the features of which the V_j values are higher than the threshold value after the input features were sorted according to their V_j values in descending order. Following the weighted feature selection, ANN is also used as a classifier. When learning with ANN, a minimum global error function was executed. It has two learning approaches, supervised and unsupervised. This study uses a supervised approach since knowing the class label may increase the classifier performance [63]. Also, a scaled conjugate gradient optimizer, which is suitable for a large scale problem, is used [64].

The second feature selection model is SVM-RFE (Recursive Feature Elimination), which used in D-FES-SVM by using the linear case [65]. The inputs are training instances and class labels. First, a feature ranked list was initialized that is filled by a subset of important features that are used for selecting training instances. The scheme continues by train the classifier and computes the weight vector of the dimension length. After the value of the weight vector was obtained, it computes the ranking criteria and finds the feature with the smallest ranking criterion. From this, the feature ranking list was updated, and the feature with the smallest ranking criterion was eliminated. A feature ranked list was finally created as its output. The last feature selection model is Decision Tree C4.5. The feature selection process begins by selecting the top-three level nodes. It then removes the equal nodes and updates the list of selected features.

Table 6 Model comparisons on selected features

Model	*DR* (%)	*FAR* (%)	*Acc* (%)	F_1 (%)	*Mcc* (%)	TBM (s)
CFS	94.85	3.31	96.27	92.04	89.67	80
Corr	92.08	0.39	97.88	95.22	93.96	2
ANN	99.79	0.47	97.88	99.10	98.84	150
SVM	**99.86**	0.39	**99.67**	99.28	99.07	10,789
C4.5	99.43	**0.23**	99.61	**99.33**	**99.13**	1,294

Table 7 Model comparisons on D-FES feature set

Model	*DR* (%)	*FAR* (%)	*Acc* (%)	F_1 (%)	*Mcc* (%)	TBM (s)
CFS	96.34	0.46	98.80	97.37	96.61	1,343
Corr	95.91	1.04	98.26	96.17	95.05	1,264
ANN	99.88	0.02	99.95	99.90	99.87	1,444
SVM	**99.92**	**0.01**	**99.97**	**99.94**	**99.92**	12,073
C4.5	99.55	0.38	99.60	99.12	98.86	2,595

A set of experiments was conducted to evaluate the performance of the proposed D-FES method in Wi-Fi impersonation detection. Choosing a proper dataset is an important step in the IDS research field [66]. The AWID Dataset [54] which comprises of Wi-Fi network data collected from real network environments, is used in this study. Fair model comparison and evaluation were achieved by performing the experiments on the same testing sets as in [54]. It is noteworthy to mention that Tables 6 and 7 summarize the experiment results in [11]. Table 6 listed the performance of each algorithm on the selected feature set only.

SVM achieved the highest Detection Rate (*DR*) (99.86%) and Matthews correlation coefficient (*Mcc*) (99.07%). However, it requires CPU Time to Build Model (TBM) of 10,789s to build a model, the longest time among the models observed. As expected, the filter-based methods (CFS and Corr) built their models quickly; however, they attained the lowest *Mcc* for CFS (89.67%). Table 7 compared the performances of the candidate models on the feature sets that were produced by D-FES.

SVM again achieved the highest *DR* (99.92%) and *Mcc* (99.92%). It also achieved the highest False Alarm Rate (*FAR*) with a value of only 0.01%. Similarly, the lowest *Mcc* was achieved by Corr (95.05%). This concludes that wrapper-based feature selections outperform filter-based feature selections. The following patterns were observed from Tables 6 and 7: Only two out of five methods (Corr and C4.5) showed lower *FAR* without D-FES, which is expected to minimize the *FAR* value of the proposed IDS. This phenomenon might exist because the original and extracted fea-

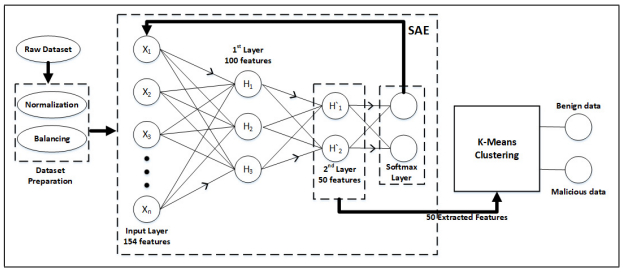

Fig. 4 Their proposed scheme contains feature extraction and clustering tasks

tures were not correlated because Corr and C4.5 measure the correlation between each feature. Filter-based feature selection methods require much shorter CPU time compared to the CPU time taken by D-FES. However, D-FES improves the filter-based feature selections performance significantly.

5.3 Deep Learning for Assisted Clustering

IDS has been becoming a vital measure in any networks, especially Wi-Fi networks. Wi-Fi networks growth is undeniable due to a vast amount of tiny devices connected via Wi-Fi networks. Regrettably, adversaries may take advantage by launching an impersonation attack, a typical wireless network attack. Any IDS usually depends on classification capabilities of machine learning, which supervised learning approaches give the best performance to distinguish benign and malicious data. However, due to massive traffic, it is difficult to collect labeled data in Wi-Fi networks. Therefore, Aminanto and Kim [15] proposed a novel fully unsupervised method which can detect attacks without prior information on the data label. The method is equipped with an unsupervised stacked autoencoder for extracting features, and a *k*-means clustering algorithm for clustering task. There are two main tasks, feature extraction, and clustering tasks. Figure 4 shows their proposed scheme which contains two main functions in cascade.

A real Wi-Fi networks-trace, AWID dataset [54] is used, which contains 154 original features. Before the scheme starts, normalizing and balancing process should be done to achieve best training performance. The scheme starts with two cascading encoders, and the output features from the second layer then forwarded to the clustering algorithm. The first encoder has 100 neurons as the first hidden layer while the second encoder comes with 50 neurons only. A standard rule for choosing the number of neurons in a hidden layer is using 70% to 90% of the previous layer. In this paper, k=2 was defined since they considered two classes only. The scheme ends by two clusters formed by *k*-means clustering algorithm. These clusters represent benign and malicious data.

There are two hidden layers in the SAE network in [15] with 100 and 50 neurons accordingly. The encoder in the second layer fed with features formed by the first layer of the encoder. The softmax activation function was implemented in the final stage of the SAE to optimize the

Table 8 Evaluation of clustering output

Input	*DR*(%)	*FAR*(%)	*Acc*(%)	*Prec*(%)	F_1(%)
Original data	100.00	57.17	55.93	34.20	50.97
1st hidden layer	100.00	57.48	55.68	34.08	50.83
2nd hidden layer	92.18	4.40	94.81	86.15	89.06

Table 9 IDS leveraging SAE

Publication	Role of SAE	Combined with
AK16a [14]	Classifier	ANN
AK16b [13]	Feature Extractor	Softmax Regression
AK17 [15]	Clustering	*K*-means Clustering
ACTYK17 [11]	Feature Extractor	SVM, DT, ANN

SAE training. The 50 features extracted from the SAE were then forwarded to *k*-means clustering algorithm as input. Random initialization was used for *k*-means clustering algorithm. However, a particular value must be defined as a random number seed for reproducibility. Clustering results were compared from three inputs: original data, features from the first hidden layer of the SAE and features from the second hidden layer of the SAE as shown in Table 8.

It was observed that the limitation of a traditional *k*-means algorithm, which unable to clusters complex and high dimensional data of AWID dataset, as expressed by 55.93% of accuracy only. Although 100 features coming from the 1st hidden layer achieved 100% of detection rate, the false alarm rate was still unacceptable with 57.48%. The *k*-means algorithm fed by 50 features from the 2nd hidden layer achieved the best performance among all as shown by the highest F_1 score (89.06%) and *Acc* (94.81%), also the lowest *FAR* (4.40%). Despite a bit lower detection rate, the scheme improved the traditional *k*-means algorithm in overall by almost twice F_1 score and accuracy.

5.4 Comparison on AWID Dataset

The goal of deep learning method is learning feature hierarchies from the lower level to higher level features [67]. The technique can learn features independently at multiple levels of abstraction, and thus discover complicated functions mapping between the input to the output directly from raw data without depending on customized features by the experts. In higher-level abstractions, humans often have no idea to see the relation and connection from the raw sensory input. Therefore, the ability to learn sophisticated features, also called as feature extraction, will become necessarily needed as the amount of data increased sharply. SAE is one good instance of feature extractors. Therefore, several previous works which implement SAE as the feature extractor and other roles in the IDS module, are discussed as shown in Table 9. Feature extraction by SAE can reduce the complexity of original features of the dataset. However, besides a feature extractor, SAE can also be used for classifying and clustering tasks as shown in Table 9.

AK16b [13] used semi-supervised approach for IDS which contains feature extractor (unsupervised learning)

Table 10 Comparison on impersonation detection

Method	*DR* (%)	*FAR* (%)
AK16a [14]	65.178	0.143
AK16b [13]	92.674	2.500
AK17 [15]	92.180	4.400
ACTYK17 [11]	99.918	0.012
KKSG15 [54]	22.008	0.021

and classifier (supervised learning). SAE was leveraged for feature extraction and regression layer with softmax activation function for the classifier. SAE as feature extractor also used in ACTYK17 [11], but ANN, DT, and SVM were leveraged as a feature selection. In other words, it combines stacked feature extraction and weighted feature selections. The experiment in [11] shows that D-FES has improved the feature learning process by combining stacked feature extraction with weighted feature selection. The feature extraction of SAE is capable of transforming the original features into a more meaningful representation, so it can be efficiently used for unsupervised learning on a complex dataset.

Unlike two previous approaches, AK16a [14] and AK17 [15] used SAE for other roles than a feature extractor, namely classifying and clustering methods, respectively. ANN was adopted as a feature selection since the weight from trained models mimics the significance of the corresponding input [14]. By selecting the important features only, the training process becomes lighter and faster than before. AK16a exploited SAE as a classifier since this employs consecutive layers of processing stages in hierarchical manners for pattern classification and feature or representation learning. On the other hand, AK17 proposed a novel fully unsupervised method which can detect attacks without prior information on data label. The scheme is equipped with an unsupervised SAE for extracting features, and a *K*-means clustering algorithm for clustering task. Kolias *et al.* [54] tested many existing machine learning models on the dataset in a heuristic manner. The lowest detection rate is observed particularly on impersonation attack reaching an accuracy of 22% only. The comparison of previous approaches on impersonation detection are summarized in Table 10.

DR refers to the number of attacks detected divided by the total number of attack instances in the test dataset while FAR is the number of normal instances classified as an attack divided by the total number of normal instances in the test dataset. From Table 10, it is observed that SAE can improve the performance of IDS compared to KKSG15 [54]. It is verified that SAE achieved high-level abstraction of complex and huge Wi-Fi network data. The SAE's model free properties and learnability on complex and large-scale data fit into the open nature of Wi-Fi networks. Among all IDS, the one using SAE as a classifier achieved the lowest impersonation attack detection rate with 65.178% only. It shows that SAE can be a classifier but not excellent as the original role of SAE is a feature extractor. The usability of SAE as a feature extractor validated by AK16b [13] and ACTYK17 [11] which achieved highest DR. Even more, by a combination of SAE extractor and weighted selection [11], the best performance of DR and FAR among other was achieved. Besides that, an interesting fact is that SAE can assist *K*-means clustering algorithm to achieve better performance with DR of 92.180% [15]. However, it is required to analyze further to reduce the FAR since it achieved the highest FAR which is undesirable in IDS.

6. Summary and Further Challenges

In summary, DL is a derivative of ML models, where exploits the cascaded layers of data processing stages in a hierarchical structure for unsupervised feature learning and pattern classification. The principle of DL is to process hierarchical features of the provided input data, where the higher-level features are composed of lower-level features. Furthermore, the DL models can integrate a feature extractor and classifier into one framework which learns feature representations from unlabeled data autonomously, and thus the security experts don't need to craft the desired features manually [68].

The goal of the DL model is to learn and output feature representation which makes those models are more suitable for feature engineering. Feature engineering here includes feature/representation learning and feature selection [69]. The ability to model the traffic behavior from the most characterizing raw input internal dynamics is crucial to show the correlation between anomaly detection performance and the traffic model quality [70].

Based on our previous work, we recommend the followings for future directions in IDS researches, but are not limited to:

1. Training load in DL methods are usually huge. Therefore, how to apply this DL model in a constrained-computation device is a really challenging task.
2. Incorporating DL models as a real-time classifier will be challenging.
3. Improving unsupervised approach since huge labeled data are difficult to obtain. Therefore an IDS leveraging unsupervised approach is desirable.
4. Building an IDS that is able to detect zero-day attacks with high detection rate and low false alarm rate.
5. A comprehensive measure not only detection but also prevention is needed in the future.
6. A time series analysis by using LSTM-networks promise a good anomaly detector. However, again, the training workload still high for real-time analysis. Therefore, lightweight models of this network are desirable as shown in [49].
7. CNN has achieved the outstanding results in many research areas, especially in image recognition fields. However, in IDS researches, not so many researches can get the expected benefit by using CNN. By applying a proper text-to-image conversion, we expect to get

the full potential of CNN as shown in image recognition researches before.
8. Developing a machine learning-based alert classification IDS will become a good challenge for next work.

Acknowledgments

This work was partially supported by ID Quantique Ltd. (former SKT) (Study on cryptographic strength and performance of quantum random number generator) and Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No. 2017-0-00555, Towards Provable-secure Multi-party Authenticated Key Exchange Protocol based on Lattices in a Quantum World). The author is very indebted to M. E. Aminanto and H. C. Tanuwidjaja for their contribution to prepare for this paper.

References

[1] K. Kim, M.E. Aminanto, and H.C. Tanuwidjaja, Network Intrusion Detection Using Deep Learning: A Feature Learning Approach, Springer, 2018.
[2] C. Kolias, G. Kambourakis, and M. Maragoudakis, "Swarm intelligence in intrusion detection: A survey," Computers & Security, vol.30, no.8, pp.625–642, 2011.
[3] A.G. Fragkiadakis, V.A. Siris, N.E. Petroulakis, and A.P. Traganitis, "Anomaly-based intrusion detection of jamming attacks, local versus collaborative detection," Wireless Communications and Mobile Computing, vol.15, no.2, pp.276–294, 2015.
[4] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," Proc. Symp. Security and Privacy, Berkeley, California, pp.305–316, IEEE, 2010.
[5] G. Anthes, "Deep learning comes of age," Communications of the ACM, vol.56, no.6, pp.13–15, 2013.
[6] J. Schmidhuber, "Deep learning in neural networks: An overview," Neural networks, vol.61, pp.85–117, 2015.
[7] L. Deng, "A tutorial survey of architectures, algorithms, and applications for deep learning," APSIPA Transactions on Signal and Information Processing, vol.3, 2014.
[8] L. Deng and D. Yu, "Deep learning: methods and applications," Foundations and Trends® in Signal Processing, vol.7, no.3-4, pp.197–387, 2014.
[9] H. Motoda and H. Liu, "Feature selection, extraction and construction," Communication of IICM (Institute of Information and Computing Machinery), Taiwan, vol.5, pp.67–72, 2002.
[10] B. Tran, S. Picek, and B. Xue, "Automatic feature construction for network intrusion detection," Asia-Pacific Conference on Simulated Evolution and Learning, vol.10593, pp.569–580, Springer, 2017.
[11] M.E. Aminanto, R. Choi, H.C. Tanuwidjaja, P.D. Yoo, and K. Kim, "Deep abstraction and weighted feature selection for Wi-Fi impersonation detection," IEEE Transactions on Information Forensics and Security, vol.13, no.3, pp.621–636, 2018.
[12] T. Hamed, J.B. Ernst, and S.C. Kremer, "A survey and taxonomy on data and pre-processing techniques of intrusion detection systems," Computer and Network Security Essentials, pp.113–134, Springer, 2018.
[13] M.E. Aminanto and K. Kim, "Detecting active attacks in Wi-Fi network by semi-supervised deep learning," Conference on Information Security and Cryptography 2017 Winter, 2016.
[14] M.E. Aminanto and K. Kim, "Detecting impersonation attack in Wi-Fi networks using deep learning approach," Information Security Applications: 17th International Workshop, WISA 2016, vol.10144, pp.136–147, 2016.
[15] M.E. Aminanto and K. Kim, "Improving detection of Wi-Fi impersonation by fully unsupervised deep learning," Information Security Applications: 18th International Workshop, WISA 2017, vol.10763, pp.212–223, 2017.
[16] K. Scarfone and P. Mell, "Guide to intrusion detection and prevention systems (idps)," NIST special publication, vol.800, no.2007, 2007.
[17] J.P. Anderson, "Computer security threat monitoring and surveillance," Technical Report, James P. Anderson Company, 1980.
[18] D.E. Denning, "An intrusion-detection model," IEEE Transactions on software engineering, vol.SE-13, no.2, pp.222–232, 1987.
[19] A.H. Farooqi and F.A. Khan, "Intrusion detection systems for wireless sensor networks: A survey," Proc. Future Generation Information Technology Conference, Jeju Island, Korea, vol.56, pp.234–241, Springer, 2009.
[20] R. Mitchell and I.-R. Chen, "Behavior rule specification-based intrusion detection for safety critical medical cyber physical systems," IEEE Trans. Dependable Secure Comput., vol.12, no.1, pp.16–30, Jan. 2015.
[21] I. Butun, S.D. Morgera, and R. Sankar, "A survey of intrusion detection systems in wireless sensor networks," IEEE Commun. Surveys Tuts., vol.16, no.1, pp.266–282, 2014.
[22] M.E. Aminanto and K. Kim, "Deep learning in intrusion detection system: An overview," International Research Conference on Engineering and Technology 2016, 2016.
[23] Z. Wang, "The applications of deep learning on traffic identification," Conf. BlackHat, Las Vegas, USA, UBM, 2015.
[24] 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.
[25] R. Salakhutdinov and G. Hinton, "Deep Boltzmann machines," Artificial Intelligence and Statistics, pp.448–455, 2009.
[26] M.A. Salama, H.F. Eid, R.A. Ramadan, A. Darwish, and A.E. Hassanien, "Hybrid intelligent intrusion detection scheme," Soft computing in industrial applications, vol.96, pp.293–303, 2011.
[27] H. Poon and P. Domingos, "Sum-product networks: A new deep architecture," 2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops), pp.689–690, IEEE, 2011.
[28] R.C. Staudemeyer, "Applying long short-term memory recurrent neural networks to intrusion detection," South African Computer Journal, vol.56, no.1, pp.136–154, 2015.
[29] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural computation, vol.9, no.8, pp.1735–1780, 1997.
[30] J. Kim, J. Kim, H.L.T. Thu, and H. Kim, "Long short term memory recurrent neural network classifier for intrusion detection," 2016 International Conference on Platform Technology and Service (PlatCon), pp.1–5, IEEE, 2016.
[31] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," Proceedings of the IEEE, vol.86, no.11, pp.2278–2324, 1998.
[32] M.A. Nielsen, Neural Networks and Deep Learning, vol.2018, p.170, Determination Press, 2015.
[33] A. Graves, "Sequence transduction with recurrent neural networks," arXiv preprint arXiv:1211.3711, 2012.
[34] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," Advances in Neural Information Processing Systems, pp.2672–2680, 2014.
[35] A. Dimokranitou, Adversarial Autoencoders for Anomalous Event Detection in Images, Ph.D. thesis, Purdue University, 2017.
[36] S.S. Roy, A. Mallik, R. Gulati, M.S. Obaidat, and P.V. Krishna, "A deep learning based artificial neural network approach for intrusion detection," International Conference on Mathematics and Computing, vol.655, pp.44–53, Springer, 2017.
[37] "Kdd Cup'99," http://kdd.ics.uci.edu/databases/kddcup99/

kddcup99.html, accessed 1 Oct. 2019.
[38] S. Potluri and C. Diedrich, "Accelerated deep neural networks for enhanced intrusion detection system," 2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA), pp.1–8, IEEE, 2016.
[39] H. Larochelle, Y. Bengio, J. Louradour, and P. Lamblin, "Exploring strategies for training deep neural networks," Journal of Machine Learning Research, vol.10, no.Jan, pp.1–40, 2009.
[40] "Nsl-kdd dataset." https://www.unb.ca/cic/datasets/nsl.html, accessed 1 Oct. 2019.
[41] A. Javaid, Q. Niyaz, W. Sun, and M. Alam, "A deep learning approach for network intrusion detection system," Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS), pp.21–26, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2016.
[42] Y. Yu, J. Long, and Z. Cai, "Session-based network intrusion detection using a deep learning architecture," Modeling Decisions for Artificial Intelligence, vol.10571, pp.144–155, Springer, 2017.
[43] Y. Liu, S. Liu, and Y. Wang, "Route intrusion detection based on long short term memory recurrent neural network," DEStech Transactions on Computer Science and Engineering, no.cii, 2017.
[44] 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," 2016 International Conference on Wireless Networks and Mobile Communications (WINCOM), pp.258–263, IEEE, 2016.
[45] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," IEEE Access, vol.5, pp.21954–21961, 2017.
[46] "Kddtest+," https://github.com/defcom17/NSL_KDD/blob/master/KDDTest%2B.txt, accessed 1 Oct. 2019.
[47] Z. Li, Z. Qin, K. Huang, X. Yang, and S. Ye, "Intrusion detection using convolutional neural networks for representation learning," International Conference on Neural Information Processing, vol.10638, pp.858–866, Springer, 2017.
[48] L. Bontemps, V.L. Cao, J. McDermott, and N.A. Le-Khac, "Collective anomaly detection based on long short-term memory recurrent neural networks," International Conference on Future Data and Security Engineering, vol.10018, pp.141–152, Springer, 2016.
[49] M.K. Putchala, Deep Learning Approach for Intrusion Detection System (IDS) in the Internet of Things (IoT) Network using Gated Recurrent Neural Networks (GRU), Ph.D. thesis, Wright State University, 2017.
[50] P.K. Bediako, "Long short-term memory recurrent neural network for detecting ddos flooding attacks within tensorflow implementation framework," master's thesis, Lulea University, 2017.
[51] S.-G. Choi and S.-B. Cho, "Adaptive database intrusion detection using evolutionary reinforcement learning," International Joint Conference SOCO17, CISIS17, ICEUTE17, Spain, Sept. 6–8, 2017, Proceeding, vol.649, pp.547–556, Springer, 2017.
[52] M. Feng and H. Xu, "Deep reinforecement learning based optimal defense for cyber-physical system in presence of unknown cyber-attack," 2017 IEEE Symposium Series on Computational Intelligence (SSCI), pp.1–8, IEEE, 2017.
[53] F. Palmieri, U. Fiore, and A. Castiglione, "A distributed approach to network anomaly detection based on independent component analysis," Concurrency and Computation: Practice and Experience, vol.26, no.5, pp.1113–1129, 2014.
[54] C. Kolias, G. Kambourakis, A. Stavrou, and S. Gritzalis, "Intrusion detection in 802.11 networks: empirical evaluation of threats and a public dataset," IEEE Commun. Surveys Tuts., vol.18, no.1, pp.184–208, 2015.
[55] M. Sabhnani and G. Serpen, "Application of machine learning algorithms to KDD intrusion detection dataset within misuse detection context," Proc. Int. Conf. Machine Learning; Models, Technologies and Applications (MLMTA), Las Vegas, USA, pp.209–215, 2003.
[56] D.T. Larose, Discovering Knowledge in Data: An introduction to data mining, John Wiley & Sons, 2014.
[57] H. Bostani and M. Sheikhan, "Modification of supervised OPF-based intrusion detection systems using unsupervised learning and social network concept," Pattern Recognition, vol.62, pp.56–72, 2017.
[58] W. Wang, X. Zhang, S. Gombault, and S.J. Knapskog, "Attribute normalization in network intrusion detection," Proc. Int. Symp. Pervasive Systems, Algorithms, and Networks (ISPAN), Kaohsiung, Taiwan, pp.448–453, IEEE, Dec. 2009.
[59] N.Y. Almusallam, Z. Tari, P. Bertok, and A.Y. Zomaya, "Dimensionality reduction for intrusion detection systems in multi-data streams—a review and proposal of unsupervised feature selection scheme," Emergent Computation, vol.24, pp.467–487, 2017.
[60] Q. Wei and R.L. Dunbrack Jr, "The role of balanced training and testing data sets for binary classifiers in bioinformatics," Public Library of Science (PloS) one, vol.8, no.7, pp.1–12, 2013.
[61] Q. Xu, C. Zhang, L. Zhang, and Y. Song, "The learning effect of different hidden layers stacked autoencoder," Proc. Int. Con. Intelligent Human-Machine Systems and Cybernetics (IHMSC), Zhejiang, China, pp.148–151, IEEE, Aug. 2016.
[62] H.Z.M. Shafri and F.S.H. Ramle, "A comparison of support vector machine and decision tree classifications using satellite data of langkawi island," Information Technology Journal, vol.8, no.1, pp.64–70, 2009.
[63] L. Guerra, L.M. McGarry, V. Robles, C. Bielza, P. Larrañaga, and R. Yuste, "Comparison between supervised and unsupervised classifications of neuronal cell types: a case study," Developmental neurobiology, vol.71, no.1, pp.71–82, 2011.
[64] M.F. Møller, "A scaled conjugate gradient algorithm for fast supervised learning," Neural Networks, vol.6, no.4, pp.525–533, 1993.
[65] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, "Gene selection for cancer classification using support vector machines," Machine Learning, vol.46, no.1-3, pp.389–422, 2002.
[66] A. Özgür and H. Erdem, "A review of KDD99 dataset usage in intrusion detection and machine learning between 2010 and 2015," PeerJ PrePrints, vol.4, p.e1954v1, 2016.
[67] Y. Bengio, "Learning deep architectures for ai," Foundations and trends® in Machine Learning, vol.2, no.1, pp.1–127, 2009.
[68] 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, 2016.
[69] P. Louvieris, N. Clewley, and X. Liu, "Effects-based feature identification for network intrusion detection," Neurocomputing, vol.121, pp.265–273, 2013.
[70] F. Palmieri, U. Fiore, and A. Castiglione, "A distributed approach to network anomaly detection based on independent component analysis," Concurrency and Computation: Practice and Experience, vol.26, no.5, pp.1113–1129, 2014.

Kwangjo Kim received the B.Sc. and M.Sc. degrees in electronic engineering from Yonsei University, Seoul, Korea, in 1980 and 1983, respectively, and the Ph.D. degree from the Division of Electrical and Computer Engineering, Yokohama National University, Yokohama, Japan, in 1991. He worked at Electronics and Telecommunications Research Institute (ETRI) from 1979 to 1997 as a head in coding section #1. He was a Visiting Professor with the Massachusetts Institute of Technology, Cambridge, MA, USA and the University of California at San Diego, La Jolla, CA, USA, in 2005 and the Khalifa University of Science, Technology and Research, Abu Dhabi, UAE, in 2012 and an Education Specialist with the Bandung Institute of Technology, Bandung, Indonesia, in 2013. He is currently a Full Professor with the School of Computing and Graduate School of Information Security, Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea, the Korean representative to IFIP TC-11 and the honorable President of the Korea Institute of Information Security and Cryptography (KIISC). Prof. Kim served as a Board Member of the International Association for Cryptologic Research (IACR) from 2000 to 2004, the Chairperson of the Asiacrypt Steering Committee from 2005 to 2008 and the President of KIISC in 2009. He is a Fellow of IACR, a member of IEICE, IEEE, ACM, and a member of IACR Fellow Selection Committee. He serves an Editor-in-Chief of online journal Cryptography, an Editor of Journal of Mathematical Cryptology and an Associate Editor of IEEE Trans. on Dependable and Secure Computing. Moreover, Prof. Kim serves as General Chairs of Asiacrypt2020 and PQCrypto2021 which will be held at Daejeon, Korea in 2020 and 2021, respectively. His current research interests include the theory of cryptology, information security and its applications.