Journal of Network and Computer Applications 169 (2020) 102767

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Deep learning methods in network intrusion detection: A survey and an objective comparison

Sunanda Gamage *, Jagath Samarabandu

Department of Electrical and Computer Engineering, University of Western Ontario, Canada

ARTICLE INFO

Keywords:
Network intrusion detection
Deep learning
Deep neural networks
Survey

ABSTRACT

The use of deep learning models for the network intrusion detection task has been an active area of research in cybersecurity. Although several excellent surveys cover the growing body of research on this topic, the literature lacks an objective comparison of the different deep learning models within a controlled environment, especially on recent intrusion detection datasets. In this paper, we first introduce a taxonomy of deep learning models in intrusion detection and summarize the research papers on this topic. Then we train and evaluate four key deep learning models - feed-forward neural network, autoencoder, deep belief network and long short-term memory network - for the intrusion classification task on two legacy datasets (KDD 99, NSL-KDD) and two modern datasets (CIC-IDS2017, CIC-IDS2018). Our results suggest that deep feed-forward neural networks yield desirable evaluation metrics on all four datasets in terms of accuracy, F1-score and training and inference time. The results also indicate that two popular semi-supervised learning models, autoencoders and deep belief networks do not perform better than supervised feed-forward neural networks. The implementation and the complete set of results have been released for future use by the research community. Finally, we discuss the issues in the research literature that were revealed in the survey and suggest several potential future directions for research in machine learning methods for intrusion detection.

1. Introduction

Computer networks have expanded greatly in size, usage and complexity over the last decade. New types of devices and network architectures have emerged, such as cloud computing and the Internet of Things. With the expansion of these networks and systems, their security has become a key issue. According to the 2019 Cyberthreat Defense Report published by the CyberEdge group (CyberEdge, 2019), attacks on global networks of large enterprises have been increasing during the past five years (Fig. 1). These attacks include, among others, denial of service attacks, malware and ransomware attacks and advanced persistent threats (APT). APTs can be particularly dangerous and costly, as these are targeted, long-term attacks launched by resourceful malicious actors against government and private organizations with the intent of exfiltrating data and sabotaging infrastructure. The cybersecurity report M-Trends 2019 (FireEye, 2019) reveals that in 2018, these attacks had a mean dwell time of 184 days (the time that the attack is effective before it is detected).

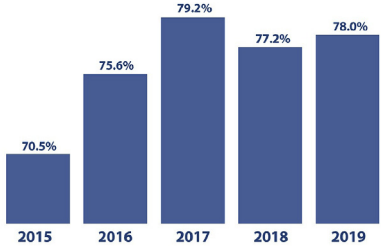

Fig. 1. Percentage of large enterprises with reported attacks (CyberEdge, 2019).

The first line of defense against cyber attacks on a computer system is formed by a well-designed security framework that enforces standard security practices, which includes confidentiality management, access control, authentication and other security policies. However, attacks may still pose a threat due to operational mishaps, system vulnerabili-

* Corresponding author.
E-mail addresses: sgamage2@uwo.ca (S. Gamage), jagath@uwo.ca (J. Samarabandu).

https://doi.org/10.1016/j.jnca.2020.102767
Received 1 February 2020; Received in revised form 28 May 2020; Accepted 7 July 2020
Available online 5 August 2020

ties (especially, those targeted by zero-day exploits) and other reasons. Intrusion detection systems (IDS) perform the crucial task of detecting such attacks on computers and networks and alerting the system operators. In one case, an IDS in a SCADA system of a water utility helped detect a crypto-mining malware on the system servers by alerting the security team of anomalous HTTP traffic (Radiflow, 2018).

An IDS may be placed in individual hosts in a network, in a dedicated central location, or distributed across a network. IDS's that are designed to detect attacks on a network of computers rather than a single host are called network intrusion detection systems (NIDS). These systems monitor network functionality in the form of network telemetry, which may include network traffic, metadata of network flows (eg: protocols such as NetFlow) and activity logs from hosts, and attempt to detect attack occurrences.

Techniques used for intrusion detection can be broadly categorized into two groups: signature-based and anomaly detection methods (Shimeall and Spring, 2014). Signature-based methods operate by matching the input telemetry data with a set of known attack patterns or signatures. These methods give accurate detections on previously known attacks, yet they fail at detecting unseen new attacks. Anomaly detection methods on the other hand establish a notion of normal behavior in the data, and flag deviations from this behavior (anomalies) as attacks. This method of detection enables the detection of preciously unseen attacks. However, since they flag any anomaly as an attack, these methods are known to yield a high number of false alarms.

Machine learning methods for intrusion detection have been studied by researchers for over 20 years (Mukkamala et al., 2002). The large volume of network telemetry and other types of security data has made the intrusion detection problem amenable to machine learning methods. Many modern commercial intrusion detection systems use machine learning based algorithms as part of their detection strategy (eg: security platforms Cisco Stealthwatch (Cisco, 2018) and Microsoft Azure Sentinel (Microsoft, 2019). These methods generally fall under the anomaly detection category of intrusion detection technique. The machine learning models can be broadly categorized into two groups: shallow learning or classic models and deep learning models. Generally, deep learning models in current use are neural network models with a high number of hidden layers. These models are capable of learning highly complex non-linear functions, and the hierarchical arrangement of layers enables them to learn useful feature representations from input data (Bengio, 2009). Deep learning methods have gained recent success in multiple domains such as image classification (Krizhevsky et al., 2012; He et al., 2016), speech recognition (Hinton et al., 2012; Amodei et al., 2016) and machine translation (Sutskever et al., 2014; Wu et al., 2016). In the domain of intrusion detection, both classic machine learning models (Buczak and Guven, 2016) and deep learning models (Berman et al., 2019) have been used with promising results.

Several recent survey papers cover the literature on deep learning methods for intrusion detection (Berman et al., 2019; Mishra et al., 2019). However, the literature is lacking in an empirical evaluation of deep learning models evaluated on standard intrusion datasets; especially, newer datasets. In this paper, we present a taxonomy and a broad survey of research papers on the topic of deep learning methods for intrusion detection. Four key deep learning models in the literature are trained and evaluated on two legacy datasets (KDD 99, NSL-KDD) and two modern datasets (CIC-IDS2017, CIC-IDS2018), and this benchmark implementation and its complete result set is made publicly available. The remainder of this paper is organized as follows. The rest of Section 1 briefly reviews the related literature and highlights the contributions of this paper. Section 2 is a preliminary discussion on the problem of intrusion detection, and section 3 presents the taxonomy and the survey of the literature on deep learning methods for intrusion detection. Section 4 describes in detail the series of experiments in the benchmark and the analysis of the results. Section 5 is a discussion on issues in current research and future directions, and section 6 gives concluding remarks.

1.1. Related work

Several reviews of the literature on machine learning methods for intrusion detection have been published in recent years, some of them focusing on deep learning methods. In one of the earlier surveys Tsai et al. (2009) review machine learning models in the intrusion detection domain published between 2000 and 2007. Since this is an older survey, it does not capture the research developments in intrusion detection during the past decade. It can be seen that during this period (2000–2007), classic machine learning models such as support vector machines, shallow artificial neural networks, and decision trees have been popular in research. The authors categorize the models into three groups: single classifiers, hybrid classifiers (cascade of different classifiers) and ensemble classifiers (combination of weak learners).

Buczak and Guven (2016) and Farah et al. (2015) cover a wide range of machine learning algorithms in the intrusion detection domain. Use of classic algorithms such as SVMs, decision trees and random forests are reviewed in detail, as well as hidden Markov models and evolutionary algorithms. Both these surveys review research published before 2015 and they do not explore any class of algorithms in depth. In particular, deep learning methods in intrusion detection are not covered in these surveys. Several surveys published in 2018 and 2019 (Mishra et al., 2019; Chaabouni et al., 2019; Sultana et al., 2019) review more recent research on this topic, which include some deep learning methods. However, the selection of deep learning algorithms in these articles is limited, as they are intended to be surveys of a broad set of machine learning models for intrusion detection.

Several survey papers focus on the topic of deep learning methods for intrusion detection. Hodo et al. (2017) propose a taxonomy which accommodates deep learning methods by classifying them into two groups: generative and discriminative architectures. However, important models such as deep feed forward networks and recurrent neural networks are not comprehensively covered in this review. Kwon et al. (2017) provide a detailed summary of a selection of seven research papers on deep learning models for intrusion detection. Xin et al. (2018) review three types of deep learning models in cybersecurity: deep belief networks, recurrent neural networks and convolutional networks. Both these surveys lack breadth of coverage of deep learning models used in intrusion detection. Al-Garadi et al. (2018) survey a breadth of deep learning models in the domain of IoT security. However, few of the reviewed works are relevant to intrusion detection.

Of the recent survey papers, Berman et al. (2019) provides perhaps the most comprehensive review of deep learning models used in cybersecurity tasks, including intrusion detection. The authors note that restricted Boltzmann machines, autoencoders and recurrent neural networks are particularly popular in the intrusion detection domain. They also note the difficulty in comparing the performance of different models, as researchers use different datasets and metrics for model evaluation. None of the survey papers mentioned above perform any benchmark or empirical analysis of the reviewed deep learning models themselves.

Yan et al. (2018) conduct an empirical performance analysis of four deep learning models (multi-layer perceptron (MLP), restricted Boltzmann machine (RBM), Sparse autoencoder (SAE) and MLP with feature embeddings) on two intrusion datasets (NSL-KDD and UNSW-NB15). However, their experiments do not cover some of the popular models like recurrent neural networks, and no evaluation is done on newer intrusion datasets. Further, the authors report only accuracy, precision and recall of the experiments, and the analysis lacks depth and insight into the performance of the models. In a recent and concurrent work, Ferrag et al. (2020) analyze the performance of seven deep learning models on CSE-CIC-IDS2018 and Bot-IoT datasets. This benchmark

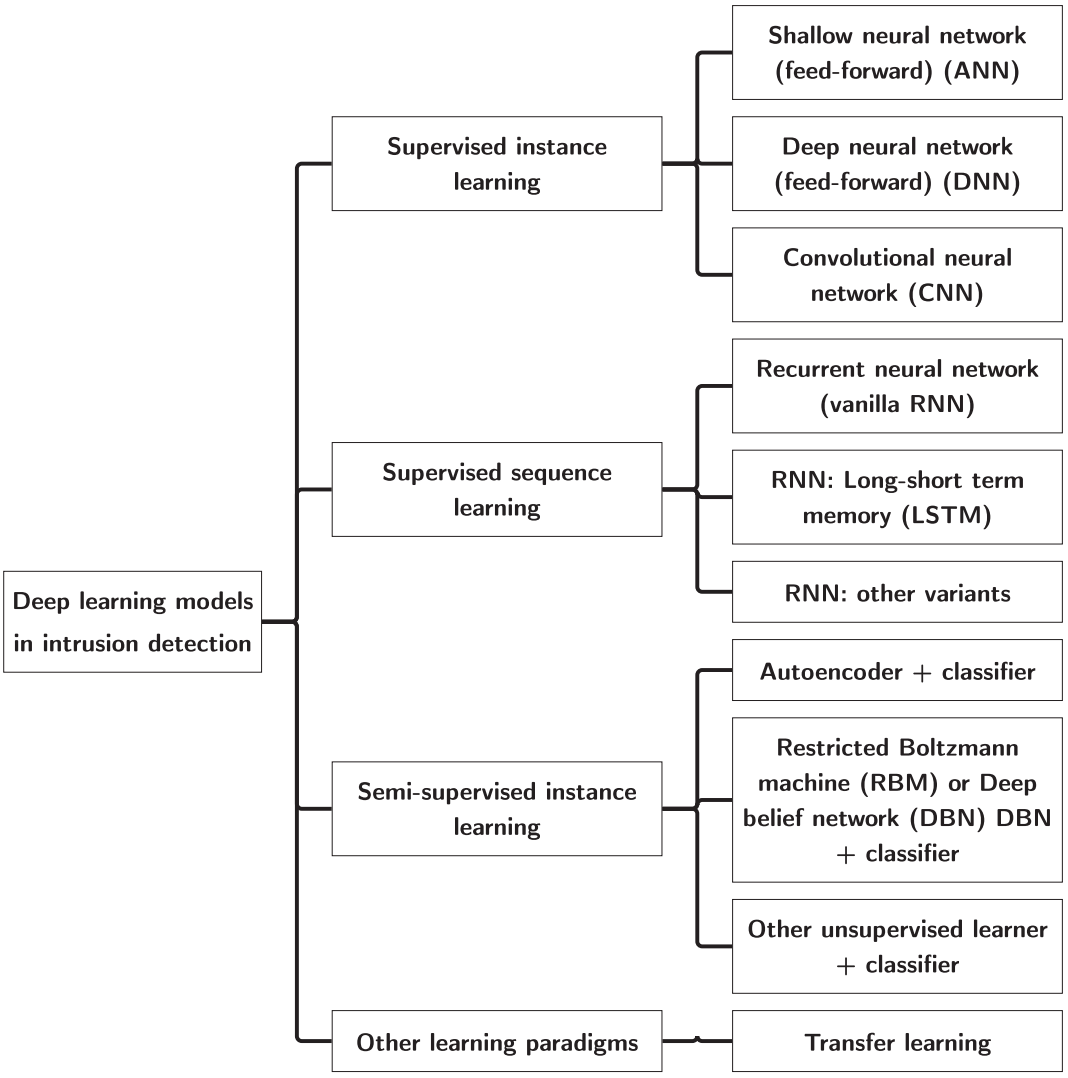

Fig. 2. Taxonomy of deep learning models in intrusion detection.

evaluates the models on only two datasets and report three evaluation metrics: per-class detection rate, overall accuracy and training time. We were unable to verify important parameters of the deep network models such as the number of hidden layers. In contrast, the full implementation of our study is publicly available to encourage objective comparisons and to promote transparency. We train and evaluate both shallow and deep versions of neural network models and report a diverse range of evaluation metrics (accuracy, F1-score, false positive and negative rates, training and inference time etc). Furthermore, we conduct experiments to compare supervised vs. semi-supervised models, a thorough neural architecture search to understand the effects of network depth and width on performance, and an analysis of data efficiency of feed-forward neural networks.

Table 4 lists prior surveys on the topic of deep learning methods in intrusion detection.

1.2. Contributions

In this paper, we attempt to address the shortcomings of the current surveys on deep learning models for intrusion detection as discussed above. The major contributions of this paper are as follows:

- We present a taxonomy of deep learning methods used for intrusion detection and summarize the recent relevant literature under this taxonomy.
- We implement a benchmark of four key deep learning models used in the intrusion detection literature, and evaluate them on two legacy datasets (KDD 99, NSL-KDD) and two modern datasets (CIC-IDS2017, CIC-IDS2018). The deep learning models have been selected from the top three branches of the taxonomy (Fig. 2), and they are representative of three different approaches to building deep learning models: feed-forward neural networks that classify flow instances, LSTM networks that classify sequences of flows, autoencoder and deep belief networks that are trained in a semi-supervised manner with both unlabeled and labeled data. This empirical comparison aims to address the difficulty of comparing the models by results reported in research works due to differences in evaluation metrics, operations on datasets, and limited evaluation on recent datasets.
- The implementation[1] and the complete set of experiment results[2] are released for further use and analysis by the research community.
- We discuss the issues in current research on machine learning for intrusion detection and highlight several future research directions guided by our findings.

[1] https://github.com/sgamage2/dl_ids_survey.
[2] https://github.com/sgamage2/dl_ids_survey/paper_results.

We believe that our survey of the literature, and the comprehensive empirical comparison of deep learning models provides a bird's-eye view of the field and insights to both new and experienced researchers in this field. Researchers may also find the open-source implementation of the benchmark useful as a reference point for their future work and experimentation.

2. Preliminary discussion

2.1. The network intrusion detection problem

The goal of an intrusion detection system (IDS) is to monitor the activity or behavior of a system, identify when attacks are occurring and generate alarms so that action can be taken to mitigate the consequences. In a computer system network, the input to the IDS (activity to be monitored) takes the form of network telemetry (traffic statistics, information extracted from packet headers and content) and information from hosts (process behavior, system call traces, application logs, file system changes). The output of the IDS could be a label or an attack score for each input or a sequence of inputs. The label can be binary: normal and attack, or multi-valued indicating different types of attacks.

From a machine learning perspective, this problem can formulated as either an anomaly detection or a classification problem. To train an anomaly detection model, a dataset of normal inputs is sufficient. To train a classifier, a labeled dataset of normal and attack inputs is necessary (unlabeled data maybe helpful, but most classification models need some amount labeled data for training). After a model is trained, it can be deployed to make detections on new data from the system.

2.2. Datasets

A number of network intrusion detection datasets are available publicly, some of which have been used for training and evaluating the models in the works surveyed. It is not in the scope of this paper to provide a comprehensive overview of all datasets (we direct the interested reader to Ring et al. (2019)). This section contains a brief summary of the more frequently used datasets below, and the rationale for using them in our experiments for model comparison. Key characteristics of the datasets are summarized in Tables 1 and 2

- KDD 99 (KDD, 1999): This is one of the earliest public datasets for intrusion detection and the most frequently used in the reviewed literature. It is a flow-based dataset (augmented with additional metadata from hosts) with 41 features and attacks of four categories: denial of service (DoS), probe, remote to local (R2L), and user to root (U2R). Despite its frequent usage, the dataset contains many deficiencies such as duplicate records, simulation artifacts and the fact that it does not reflect modern network traffic and attacks. In our experiments, we train and evaluate machine learning models on the KDD 99 dataset for two reasons: 1) as a first step to verify that the models are working as expected and 2) to be able to compare with previous results reported on this dataset.
- NSL-KDD (Tavallaee et al., 2009): The NSL-KDD dataset addresses some of the deficiencies in the KDD 99 set by removing duplicate records and selecting records that are difficult to classify. It is similar in structure to the KDD 99 dataset. Although this dataset is an improved version of the KDD 99 set, it is still not suitable for building intrusion detection models, and we use it in our experiments for the same reasons quoted for the KDD 99 dataset.
- CIC-IDS2017 (Sharafaldin et al., 2018): This dataset was recorded in a small network environment with simulated normal traffic. Six categories of modern attacks are launched from a separate network. Both the raw packet capture and Netflows with 80 features are made available. A small number of research papers reviewed in this survey use this dataset. We believe that this dataset is representative of modern network traffic, and therefore we use it in our experiments.
- CSE-CIC-IDS2018 (CIC, 2018): This dataset is similar to CIC-IDS2017 in structure and the contained attacks. However, it is prepared on a larger network representing a corporation as the victim network (420 client machines and 30 servers) and a larger attacker network (50 machines).

2.3. Evaluation metrics

There are several metrics and tools used by researchers to evaluate the performance of machine learning models in intrusion detection. These metrics measure different aspects of a model or an IDS system. A brief description is of each metric and tool is given below.

Table 1
Summary of dataset characteristics.

Dataset	No. of instances	Type of data	No. of features	Features	No. of classes
KDD 99	Train: 4,898,431 Test: 311,029	Connection records	41 (3 categorical, 39 numeric)	duration, src_bytes, serror_rate, su_attempted etc.	24 original classes mapped to 5 classes
NSL-KDD	Train: 125,973 Test: 22,544				
IDS 2017	Total: 2,827,808	Bi-directional NetFlows	78 (1 categorical, 77 numeric)	duration, packet counts and sizes in both directions etc.	12 classes
IDS 2018	Total: 16,137,183				15 classes

Table 2
Attack types present in the datasets.

Dataset	Attack types
KDD 99 and NSL-KDD	Probe: ipsweep, nmap, portsweep, satan DoS (Denial of Service): back, land, neptune, pod, smurf, teardrop U2R (User to Root): buffer_overflow, loadmodule, perl, rootkit R2L (Remote to Local): ftp_write, guesspasswd, imap, multihop, phf, spy, warezlient, warezmaster
IDS 2017	Bot, DDoS, DoS GoldenEye, DoS Hulk, DoS Slowhttptest, DoS slowloris, FTP-Patator, PortScan, SSH-Patator, Web Attack Brute Force, Web Attack XSS
IDS 2018	Bot, Brute Force - Web, Brute Force - XSS, DDOS HOIC, DDOS LOIC-UDP, DDoS LOIC-HTTP, DoS GoldenEye, DoS Hulk, DoS SlowHTTPTest, DoS Slowloris, FTP-BruteForce, Infilteration, SQL Injection, SSH - Bruteforce

Table 3
Confusion matrix for a binary class problem: TP = True Positives, TN = True Negatives, FP = False Positives, FN = False Negatives.

		Predicted class	
		Attack	Benign
Actual class	Attack	TP	FN
	Benign	FP	TN

The confusion matrix is a tabulation of classifications made by a model, typically with the actual class (ground truth) on rows and predicted class on columns. It can be obtained for both binary classifications (Table 3), and multi-class classifications. It shows the "classification distribution" of a model, and helps identify properties of the model, such as when it is consistently misclassifying one class as another. For example, a model may classify nearly all DDoS attempts as a data exfiltration. The researcher can then examine potential causes for this issue. This type of insight is not readily gleaned from other metrics.

Several other important metrics are calculated based on the values in the confusion matrix (Table 3). They include overall accuracy, precision, recall, specificity, false positive rate and F1 score. See Appendix A for definitions of the metrics.

For multi-class problems, these metrics can be calculated per-class, treating the classification as a one-vs-all problem. A final metric may be obtained by averaging the per-class metrics in one of three ways.

- Micro average: calculate the metrics using the sum of each type of classification (eg: $TP = TP_{class1} + TP_{class2} + \ldots$).
- Macro average: calculate the per-class metrics and average them (sum and divide by the no. of classes)
- Weighted macro average: similar to macro average, except the per-class metrics are weighted by the number of instances in each class to obtain the final average.

Note that a natural trade-off exists between sensitivity (recall) and false positive rate (FPR); i.e. a highly sensitive model (desirable) is likely to have a high false positive rate (undesirable). This relationship is illustrated by a Receiver Operating Characteristic (ROC) curve, which has FPR in the horizontal axis, and sensitivity (recall) in the vertical axis. The ROC curve is obtained by changing the threshold for classification into the positive class (attacks), and recording the corresponding FPR and recall. One can then choose a point on the ROC curve that is suitable to the application. The area under the ROC curve (AUC) is another metric for classifiers that reflects the ability of the model to distinguish between the two classes.

In addition to the metrics that evaluate classification performance of a model, several other important metrics reflect model complexity. In a deep learning model, complexity is a function of the number of parameters (weights) and architecture.

- Time to train: This can be measured as the number of training epochs (iterations over the dataset taken by an algorithm like gradient decent) needed to reach a goal metric on out-of-sample data (eg: a goal accuracy or a goal F1-score). The time taken (in seconds or minutes) for those training epochs can also be measured. If the model is trained offline, a longer time to train it may be acceptable. However, if the model must be trained in real-time, fast training is required.
- Time to predict (latency): this can be measured as the time taken to make a prediction on a fixed-sized batch of input instances. Since predictions in an intrusion detection system are made online (real-time), low latency figures are preferred.
- Throughput: the number of predictions that can be made in a time period. This is a function of prediction latency and the manner the model is deployed in an IDS (eg: parallel deployments yield high throughput). Given the large volume of network telemetry data that an IDS has to examine, high throughput figures are required for proper operation.

3. Taxonomy and survey of deep learning methods for intrusion detection

Various taxonomies have been presented in previous surveys on the topic of deep learning models in intrusion detection (Hodo et al., 2017; Kwon et al., 2017; Chaabouni et al., 2019). The taxonomy we propose for this survey is shown in Fig. 2. For this taxonomy, we have considered the machine learning practitioner's point of view rather than conceptual classification of machine learning models. For example, previous taxonomies identify generative vs. discriminative models, which is an important conceptual categorization. However, the distinctions we make are more likely to help practitioners in terms of data pre-processing and implementing a model building pipeline in addition to highlighting conceptual differences. For example, instance vs. sequence classification require different arrangements of the dataset, and semi-supervised methods typically involve a pre-training stage (with unlabeled data) in the machine learning pipeline. A transfer learning method requires downloading of a suitable pre-trained model from a repository.

We have surveyed the research literature on deep learning methods in intrusion detection and compiled a summary of the key works. Table 4 shows a list of current survey papers on this topic and Tables 5–8 shows research papers categorized under each branch of the taxonomy. Finally, four key neural network models from the first three branches of the taxonomy were implemented for empirical comparison (section 4).

4. Empirical comparison

4.1. Overview of experiments: models, datasets and evaluation metrics

The intrusion detection problem is frequently formulated as a classification problem in the literature reviewed in this paper. For empirical analysis, four neural network classifiers were selected from the different categories in the model taxonomy (Fig. 2), and they were trained and evaluated on four popular intrusion detection datasets. For each deep learning model type, a shallow model (single hidden layer) and a deep model (multiple hidden layers) were implemented. These models were selected as they are representative of the deep learning models frequently used in the intrusion detection research literature (applicability), and they represent supervised, semi-supervised and sequential types of models (diversity and coverage). The following models were selected for the empirical comparison:

1. *Artificial neural network: feed-forward (ANN).* Feed-forward neural networks yield state-of-the-art performance on classification tasks in many domains, and it is one of the key models in the deep learning literature (LeCun et al., 2015). For network intrusion detection, both shallow and deep feed-forward neural networks (DNNs) have been used since the KDD 99 competition (Ingre and Yadav, 2015; Vinayakumar et al., 2019). Therefore, feed-forward neural networks were selected as the main model for experimentation on the intrusion detection datasets.
2. *Autoencoder followed by ANN (AE + ANN).* In this semi-supervised learning mechanism, first an autoencoder with a symmetric encoder-decoder architecture is trained with unlabeled data (a portion of the training set). This unsupervised training phase is expected to capture feature transformations from the data. Next, the labeled data of the training set are transformed using the autoencoder, and the output of the final encoding layer (feature transformation) is input to a supervised classification model. In our experiments, a feed-forward neural network (ANN) after the final encoding layer

Table 4
List of survey papers on deep learning and other machine learning methods for intrusion detection. (DNN = deep neural network, RNN = recurrent neural network, CNN = convolutional neural network, RBM = restricted Boltzmann machine, DBN = deep belief network, AE = autoencoder, SOM = self-organizing map, IDS = intrusion detection system).

Survey paper	Coverage of machine learning models and remarks
Ferrag et al. (2020)	Reviews 10 DNN papers, 7 RNN papers, 7 CNN papers, 9 RBM papers, 5 DBN papers and 7 AE papers. Includes an empirical comparison of seven deep learning models on two recent datasets.
Berman et al. (2019)	Reviews 3 DNN papers, 11 AE papers, 2 CNN papers, 8 RNN papers and 7 RBM papers. Includes deep learning papers on other cybersecurity applications (eg: malware detection).
Mishra et al. (2019)	Reviews a selection of 38 papers that use different machine models for the intrusion detection problem. Contains an analysis of network attacks and features used in IDS.
Chaabouni et al. (2019)	Presents a taxonomy of machine learning algorithms in network intrusion detection, with a focus on the Internet of Things. A limited selection of prior research is reviewed in detail. Discusses other aspects such as IDS architectures and deployment methods.
Sultana et al. (2019)	Summarizes five papers that use deep learning for SDN-based network intrusion detection. Discusses challenges in intrusion detection in an SDN environment.
Xin et al. (2018)	Reviews 7 DBN papers, 6 RNN papers and 5 CNN papers.
Hodo et al. (2017)	Gives a taxonomy of machine learning methods in intrusion detection. 15 papers on various neural network models are reviewed in detail (DNN, RBM, RNN, AE, CNN, SOM).
Kwon et al. (2017)	Gives a taxonomy of machine learning methods in intrusion detection. 7 deep learning IDS papers are reviewed in detail (includes DNN, AE, DBN). Introduces a 4-layer fully connected DNN with results on the NSL-KDD dataset.
Buczak and Guven (2016)	Covers a range of machine learning methods, including decision trees, Bayesian networks and Ensemble methods. Neural network models are not given focus.
Farah et al. (2015)	Covers a range of machine learning methods, including SVM, Naïve Bayes and decision trees. Neural network models are not given focus.

Table 5
List of research papers on supervised instance classification models for intrusion detection. (Acc = accuracy, Prec = precision, Rec = recall, FAR = false alarm rate, F1 = F1-score).

Model type	Paper	Dataset and problem	Model details	Results
DNN	Vinayakumar et al. (2019)	NSL-KDD, CICIDS 2017, UNSW-NB15, Kyoto, WSN-DS	ANN has 5 hidden layers with 1024, 768, 512, 256, 128 nodes. ReLU activation	In order NSL-KDD, IDS 2017: Acc = 78.5, 95.6%, Prec = 81.0, 96.2%, Rec = 78.5, 95.6%, F1 = 76.5, 95.7%
CNN	Zeng et al. (2019)	ISCX VPN-nonVPN, ISCX 2012.5-class classification	Deep CNN: 2 1D convolutional layers, 1 fully connected layers	Acc = 99.85%
DNN	Wang (2018)	NSL-KDD, 5-class classification. Original training set is split to 90% training, 10% test sets	ANN has 4 hidden layers with 245 nodes each. ReLU activation	Acc = 86.35%, Prec = 81.86%, Rec = 77.32%, F1 = 73.89%, FAR = 0.1619
CNN	Kwon et al. (2018)	NSL-KDD, Kyoto, MAWILab, binary classification	Deep CNN: 3 1D convolutional, 2 max-pooling, 1 flatten, 3 fully connected layers	F1 = 79%
DNN	Diro and Chilamkurti (2018b)	NSL-KDD, binary and 4-class classification	ANN has 3 hidden layers with 150, 120, 50 nodes	Acc = 98.27%, Rec = 96.5%, FAR = 0.0257
DNN	Jin et al. (2017)	KDD 99 (10% subset), binary classification	ANN has 4 hidden layers with 100 nodes each. ReLU activation	Acc = 99.01%, Rec = 99.81%, FAR = 0.0047
CNN	Wei Wang et al. (2017)	Custom dataset (USTC-TFC2016), binary, 10-class and 20-class classification	Traffic flows are converted to gray-scale images. CNN has 2 convolutional, 2 max-pooling, 2 fully-connected layers	20-class classifier: Acc = 99.17%, Prec = 99%, Rec = 98%, F1 = 98%
DNN	Tang et al. (2016)	NSL-KDD, binary classification	6 features are manually selected. ANN has 3 hidden layers with 12, 6, 3 nodes	Acc = 75.75%, Prec = 83%, Rec = 76%, F1 = 75%
DNN	Ma et al. (2016)	Subsets of KDD 99 and NSL-KDD, 5-class classification	Dataset is clustered using spectral clustering. Multiple ANNs are trained (one per cluster). Result is aggregated	Acc = 72.64%, Prec = 57.48%
DNN	Hodo et al. (2016)	Simulated IoT network data, binary classification	ANN has one hidden layer with 3 nodes. Sigmoid activation, MSE loss function	Acc = 99%
ANN	Ingre and Yadav (2015)	NSL-KDD, binary and 5-class classification	Reduce features to 29. ANN has single hidden layer with 29 nodes	Acc = 79.9%

of the autoencoder acts as the classification model. Semi-supervised approaches using autencoders for network intrusion detection have been explored in Shone et al. (2018) and Javaid et al. (2016).

3. *ANN initialized with deep belief network (DBN + ANN).* In this semi-supervised method, first a deep belief network (DBN) is trained with unlabeled data (a portion of the training set). Next, the weights learned by the DBN are used as the initial weights for training a neural network (DNN) of the same architecture. The ANN is then trained with the available labeled data in a supervised fashion. This is a different method of semi-supervised learning to the autoencoder; it employs weight transfer instead of data transformation (Erhan et al., 2010). Semi-supervised approaches using DBNs for network intrusion detection have been explored in Alom et al. (2015) and Alrawashdeh and Purdy (2016).
4. *Long-short term memory network (LSTM)*: Network traffic is sequential in nature, which makes recurrent neural networks such as LSTMs good candidates for the network intrusion detection problem (Bontemps et al., 2017; Diro and Chilamkurti, 2018). In our experiments,

Table 6
List of research papers on supervised sequence classification models for intrusion detection. (Acc = accuracy, Prec = precision, Rec = recall, FAR = false alarm rate, F1 = F1-score, GRU = Gated Recurrent Unit).

Model type	Paper	Dataset and problem	Model details	Results
LSTM	Diro and Chilamkurti (2018a)	ISCX 2012, AWID. Binary classification	30 embedding layers, 10 LSTM layers, and sigmoid output layer	ISCX dataset: Acc = 99.91%, Prec = 99.85%, Rec = 99.96%
GRU	Xu et al. (2018)	KDD 99, NSL-KDD, 5-class classification	GRU and Bidirectional GRU (BGRU) nets. Model has one layer with 128 GRU nodes, 3 feed-forward layers with 48 nodes	BGRU gives best results with fast convergence. On NSL-KDD: Acc = 99.24%, Rec = 99.31%, FAR = 0.84%
LSTM	Radford et al. (2018)	ISCX IDS, anomaly detection by predicting future tokens (unsupervised)	Token embedding layer, 2 bidirectional LSTM layers, 2 dense layers	AUC = 0.84, ROC curves for attack classes given
LSTM	Jiang et al. (2019)	NSL-KDD, binary classification	Training multiple LSTM nets (one hidden layer) for different features extracted (channels). Majority voting.	Acc = 98.94%, Rec = 99.23%, FAR = 9.86%
LSTM	Loukas et al. (2018)	Vehicle network dataset (custom) with 3 attack types	3 LSTM layers with 100, 800, 1000 nodes	Acc = 86.9%
RNN	Yin et al. (2017)	NSL-KDD, binary and 5-class classification	RNN has 1 hidden layer with 80 nodes, learning rate = 0.5	5-class classifier: Acc = 81.29%
LSTM	Bontemps et al. (2017)	KDD 99 (converted to time-series), anomaly detection by predicting future packets	One hidden LSTM layer with 23 nodes. Input time steps = 3. High prediction errors in a time window (12 steps) indicate attacks	Rec = 94–98%, FAR = Variable
LSTM	Le et al. (2017)	KDD 99, 5-class classification	LSTM network details are unspecified	Acc = 97.54%, Prec = 97.69%, Rec = 98.95%, FAR = 9.98%
LSTM	Taylor et al. (2016)	CAN bus data (custom), anomaly detection by predicting next word	Network has 2 dense layers, 2 LSTM layers. Input seq. length is 20 words	AUC ranging from 94.4 to 99.8
LSTM	Kim et al. (2016)	KDD 99, 5-class classification	One hidden LSTM layer with 80 nodes. Input time steps = 100. MSE loss function	Acc = 96.93%, Rec = 98.88%, FAR = 10.04%
LSTM	Staudemeyer (2015)	KDD 99, 5-class classification	LSTM network has 20 hidden nodes and variable no. of cell memory blocks	Acc = 93.82%. Comprehensive experiments and results are reported.

input sequences to the LSTM model have been formed by grouping adjacent flows or records of the datasets, as they occur in a temporal sequence. The output of the model is a label sequence; one label for each flow in the input sequence. The expectation is that the LSTM network will capture temporal relationships among adjacent flows.

5. *Random forest (RF)*: In addition to the 4 neural network models, a random forest was also trained and evaluated on the datasets. It represents the classical machine learning category of models, and it is commonly used in the intrusion detection literature. It provides a benchmark for comparison of the deep learning models.

We used two legacy datasets (KDD 99, NSL-KDD) and two modern datasets (CIC-IDS2017, CIC-IDS2018) for training and testing the machine learning models. A description of the datasets used in the empirical evaluation is given in section 2.2.

The following metrics are calculated for each model evaluation. The definition of the metrics are given in section 2.3.

- Per-class metrics: accuracy, precision, recall), false alarm rate, false negative rate, F1 score. These per-class metrics are available in the results repository .[3]
- Overall accuracy.
- Weighted macro averages of precision, false alarm rate, false negative rate, F1 score.
- Training time: the number of training epochs and the time taken to reach the average accuracy achieved by a model (epochs to convergence).
- Testing or inference time: time taken to classify a fixed size batch of 1 million instances from the dataset.

4.2. Details of the experiments

In conducting the experiments of this empirical analysis, we have placed a high importance on the reproducibility of results. To this end, we have closely followed the Reproducibility Checklist of the NeurIPS 2019 conference (Pineau, 2019). The relevant details of the experiments are outlined in the following subsections.

4.2.1. Data pre-processing

The following pre-processing steps were performed on the datasets.

- Removing invalid flow records: the IDS 2017 and IDS 2018 datasets contain flow records with invalid values in the fields (eg: missing values, strings in numeric fields). Such records were removed from both datasets (2867 in the IDS 2017 dataset, and 95,819 in the IDS 2018 dataset).
- Changing class labels to formulate the 5-class problem: the KDD 99 and NSL-KDD datasets contain 38 types of attacks (class labels), which fall into 4 attack classes (DOS, R2L, U2R and probing). Since we are developing models for the 5-class version of the problem, the 38 class labels are mapped to the 4 attack classes. With the "Normal" (benign) label, this yields 5 class labels for classification.
- One-hot encoding of categorical features: the KDD 99 and NSL-KDD datasets contain 3 non-numeric (categorical) features: *protocol_type*, *service* and *flag*. These 3 features were one-hot encoded, which yielded 84 encoded features.

[3] https://github.com/sgamage2/dl_ids_survey/paper_results.

Table 7
List of research papers on semi-supervised instance classification models for intrusion detection. (Acc = accuracy, Prec = precision, Rec = recall, FAR = false alarm rate, F1 = F1-score. AE = Autoencoder, RF = Random Forest, SM = Softmax, DBN = Deep Belief Network).

Model type	Paper	Dataset and problem	Model details	Results
AE + RF	Shone et al. (2018)	KDD 99 (10% subset), NSL-KDD, 5-class and 13-class classification	2 stacked AE's (each with 3 hidden layers: 14, 28, 28 nodes): unsupervised training. Followed by RF classifier	5-class classifier on NSL-KDD: Acc = 85.42%, Prec = 100%, Rec = 85.42%, F1 = 87.37%, FAR = 14.58%
AE + SM	Potluri and Diedrich (2016)	NSL-KDD, binary and 5-class classification	AE with 2 layers (20, 10) is pre-trained layer-wise, followed by fine-tuning a softmax layer	Overall results are not given
AE + SM	Javaid et al. (2016)	NSL-KDD, 2-class, 5-class and 13-class classification	Sparse AE for unsupervised training followed by softmax regression classifier	2-class classifier: Acc = 88.39%, Prec = 85.44%, Rec = 95.95%, F1 = 90.94%
AE + SM	Abeshu and Chilamkurti (2018)	NSL-KDD, binary classification	Stacked AE (3 layers with 150, 120, 50 nodes) followed by softmax layer	Acc = 99.2%, Rec = 99.27%, FAR = 0.85%
AE + DBN	Li et al. (2015)	KDD 99 (10% subset), binary classification	AE training round (5 layers with 41, 300, 150, 75 nodes), followed by a RBM-style training of each layer, followed by fine-tuning with a final softmax layer	Acc = 92.1%, Rec = 92.2%, FAR = 1.58%
DBN	Alom et al. (2015)	NSL-KDD (40% subset), 5-class classification	DBN model details are unspecified	Acc = 97.45%
DBN	Gao et al. (2014)	KDD 99, 5-class classification	DBN has 4 layers with 122, 150, 90, 50 nodes, followed by softmax layer (fine-tuned)	Acc = 93.49%, Rec = 92.33%, FAR = 0.76%
DBN	He et al. (2017)	Custom IEEE 118-bus dataset, binary classification	Conditional DBN (5 hidden layers with 50 nodes each). Time window = 5	Acc = 96.16%, Rec = 95.89%, FAR = 3.57%
DBN + LR	Alrawashdeh and Purdy (2016)	KDD 99 (10% subset), 5-class classification	DBN has 4 layers with 72, 52, 40, 5 nodes (pretrained), followed by logistic regression (softmax: fine-tuned)	Acc = 97.9%, Rec = 97.5%, FNR = 2.47%
DBN + DNN	Kang and Kang (2016)	Custom CAN dataset, 3-class classification	DBN pre-trained weights are used to initialize DNN	Acc = 97.8%, FAR = 1.6%, FNR = 2.8%
DRBM	Fiore et al. (2013)	KDD 99 (10% subset) and custom dataset, binary classification	Discriminative RBM (parameters are unspecified)	On custom dataset: Acc = 94%

Table 8
List of research papers on transfer learning models for intrusion detection. (TL = Transfer Learning).

Model type	Paper	Dataset and problem	Model details	Results
TL	Kneale and Sadeghi (2019)	Transfer models between UNSW NB-15 and CICIDS2017 datasets (train on one, transfer to other)	Adversarial Siamese neural network	Outperforms non-TL baseline and the CORAL TL method
TL	Zhao et al. (2019)	NSL-KDD, create source sets and target sets with attacks not present in the source set	Find a latent space where know and new attacks have minimum distance. Agnostic of the ML model	Outperforms all non-TL baselines (models trained only on target domain) and several other TL methods
TL	Taghiyarrenani et al. (2018)	Transfer models from KDD 99 to Kyoto2006	Uses similarity measures and manifold alignment between the two feature spaces	Outperforms non-TL baseline (SVM)

- Scaling features: All features of all 4 datasets were scaled by standard normalization.
- Preparing sequences for the LSTM model: inputs to recurrent neural networks are sequences of instances. For the LSTM model, sequences were formed by grouping a number of consecutive flow records (adjacent in time). The length of a sequence then becomes a hyper-parameter of the model. The class label of each flow record was attached to the individual records in a sequence (i.e. input and output is synchronized). All sequences have a fixed length.

4.2.2. Sample allocation and hyper-parameter tuning

The 3-way holdout method was used for allocating the data for training, hyper-parameter tuning and model evaluation.

- In the KDD 99 and NSL-KDD datasets, separate test sets are available (in the NSL-KDD dataset, the test is named KDDTest+). The rest of the dataset was split into two sets: 80% for training and 20% for validation. The split was made in a stratified fashion in order to preserve the original class proportions.
- In the IDS 2017 and IDS 2018 datasets, a separate test set is not available. The dataset was split into three sets: 60% for training, 20% for validation and 20% for testing. The split was made in a stratified fashion in order to preserve the original class proportions.

Hyper-parameter tuning. The final sets of hyper-parameters for each model on each dataset were found by a random search of the parameter space. We used hyper-parameter values reported in the intrusion detection literature as a guide to define the value ranges for the random search. Table 9 shows the range of values used in the random search of hyper-parameters. The configurations with the highest weighted macro-average F1 score on the validation set were selected as the final hyper-parameter configurations (Table 10).

In order to minimize overfitting of neural network models, dropout, batch normalization and early stopping of training was performed. In each model-dataset combination, the hyper-parameter configurations

Table 9
Hyper-parameter configurations/ranges used in the tuning process.

Model	Range of hyper-parameters
ANN	no. of layers = 1 - 5 no. of nodes in layers = 64, 128, 256, 512, 1024, 2048 layer activations = ReLU output layer activation = softmax loss function = cross-entropy weight initialization = He (He et al., 2015) layer dropout rates = 0.05, 0.20, 0.50 batch normalization for each layer batch size = 32, 64, 256, 1024 optimizer = Adam (Kingma and Ba, 2017)
AE + ANN	Similar configurations to ANN, except: unsupervised training data percentage (for AE) = 25%, 50%, 80% no. of nodes in AE encoder layers = 64, 128, 256, 512, 1024, 2048 symmetric encoder and decoder output layer activation of AE = sigmoid, ReLU loss function of AE = binary cross-entropy, mean squared error
DBN + ANN	Similar configurations to ANN, except: unsupervised training data percentage (for DBN) = 25%, 50%, 80% DBN optimizer = Persistent Contrastive Divergence (PCD) (Tieleman, 2008) DBN learning rate = 0.0001, 0.0005 0.001, 0.1 DBN pre-train epochs (each layer) = 20, 50
LSTM	Similar configurations to ANN, except: recurrence relation to self in all hidden nodes cell type = LSTM sequence length (time steps) = 16, 32, 50
RF	no. of trees = 100, 500, 1000 minimum no. of samples required to split a node = 2, 10, 100 (this parameter controls tree depth)

that gave the highest weighted macro-average F1 score were selected as the winning models.

4.2.3. Experiments with the best hyper-parameter configurations

The best-performing (highest weighted macro-average F1 score on validation set) hyper-parameter configurations selected from the hyper-parameter tuning process are listed in Table 10. The following experiments were conducted with these hyper-parameter configurations of the 8 neural networks (shallow and deep ANN, AE + ANN, DBN + ANN, LSTM) and the random forest model.

- *Validating the benchmark implementation and comparing model performance.* The models were trained (with the best hyper-parameter configurations) on the training sets and evaluated on the test sets of the four datasets (KDD 99, NSL-KDD, IDS 2017; IDS 2018). This training and testing process was repeated 5 times for each model-dataset combination to observe any variance of performance caused by random initialization of model weights and to verify the stability of the selected hyper-parameter configurations. Performance metrics were recorded for the 36 model-dataset combinations shown in 10 and the weighted macro average of metrics are reported (mean and standard deviation over 5 runs). As a form of validating our benchmark implementation, the metrics were compared to the results reported in the literature. A general comparison of the performance of different models along different metrics was also conducted.
- *Measuring model training and testing (inference) time.* For each model-dataset combination, the average test set accuracy from the previous experiment was set as the out-of-sample goal metric. The models were trained until they achieved the goal accuracy, and the number of training epochs and the time taken were recorded. This process was repeated 3 times for each model-dataset combination to obtain a distribution of results (mean result is reported). In the case of semi-supervised models, the autoencoders and deep beliefs were pre-trained for a fixed number of epochs (10), before fine-tuning until convergence to the goal accuracy on the test set. Since the random forest is not trained by an iterative algorithm, its training time was simply measured in minutes. The testing (inference) time was measured as the time taken (in minutes) to classify a fixed size batch of 1 million instances from the dataset (repeated 5 times for a distribution; mean and standard deviation are reported).
- *Comparing semi-supervised methods (AE + ANN and DBN + ANN models) with the supervised feed-forward neural network (ANN).* The training set of the NSL-KDD and IDS 2017 datasets were split 75%–25% in a stratified fashion. The first 75% (labels removed) was used during the unsupervised training phase (autoencoder and deep belief network pre-training), and the second 25% was used in the supervised training phase (fine-tuning). These models were then compared with an ANN model that was trained (supervised) *only* on the just the second 25% of the split. Thus, the semi-supervised models have a 75% data advantage in the form of unsupervised pre-training over the ANN model. A second experiment followed the same process with a 50% unlabeled, 50% labeled split of the datasets. A shallow network architecture with 64 nodes in the hidden layer was used for training all the models.
- *Neural network architecture search.* The initial result set showed that feed-forward neural networks (ANN) yielded better evaluation metrics on all four datasets. Therefore, a set of experiments was designed to identify how different ANN architectures impact performance in the intrusion detection task. Three types of architectures were tested. 1) Shallow but wide architectures (3 hidden layers with up to 3000 nodes in the first hidden layer). 2) Deep architectures with the same number of nodes in each layer (up to 10 hidden layers with 64 nodes in each layer). 3) Narrowing deep architectures (up to 10 hidden layers with 800 nodes in the first hidden layer narrowing down to 16 nodes in the last hidden layer). The exact network architectures and the number of parameters in the models are given in Tables B.19, B.20 and B.21. These models were trained and evaluated on three datasets: NSL-KDD and IDS 2017 and IDS 2018.
- *Measuring the effect of training set size on ANN performance.* Network intrusion detection is a domain where large quantities of data can be acquired, both by simulation and recording packet flows in real networks. In order to understand the ability of an ANN to improve performance as more data is made available to it for training, the

Table 10
Hyper-parameter configurations selected from the tuning process (configurations that give the highest validation weighted macro-average F1 score). mse = mean squared error, bce = binary cross-entropy.

Model hyper-parameters	KDD 99		NSL-KDD		IDS 2017		IDS 2018	
	Shallow	Deep	Shallow	Deep	Shallow	Deep	Shallow	Deep
ANN								
no. of layers	1	3	1	3	1	4	1	4
no. of nodes in layers	256	64,32,16	256	64,32,16	64	256,128, 64,32	512	128,64, 32,16
layer dropout rates	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2
batch size	256	256	256	256	256	256	256	256
AE + ANN								
no. of layers	1	3	1	3	1	3	1	3
no. of nodes in encoder layers	32	128,64,32	32	128,64,32	32	128,64,32	32	128,64,32
layer dropout rates	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2
batch size	256	256	256	256	256	256	256	256
unsupervised training data percentage	50%	25%	50%	25%	50%	25%	50%	25%
output layer of AE	relu	sigmoid	sigmoid	sigmoid	sigmoid	sigmoid	sigmoid	sigmoid
loss function of AE	mse	bce	bce	bce	bce	bce	bce	bce
AE pre-train epochs	200	200	200	200	200	200	40	40
DBN + ANN								
no. of layers	1	3	1	3	1	3	1	3
no. of nodes in layers	32	128,64,32	32	128,64,32	32	128,64,32	32	128,64,32
layer dropout rates	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2
batch size	256	256	256	256	256	256	256	256
unsupervised training data percentage	50%	25%	50%	25%	50%	50%	50%	50%
DBN learning rate	0.0001	0.001	0.0001	0.001	0.0001	0.0001	0.0001	0.0001
DBN pre-train epochs	50	50	50	50	50	50	12	12
LSTM								
no. of layers	1	2	1	2	1	2	1	2
no. of nodes in layers	32	32,16	32	64,32	32	64,32	32	64,32
layer dropout rates	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2
batch size	256	256	256	256	256	256	256	256
sequence length	32	32	32	16	32	32	32	32
Random forest								
no. of trees	100		100		100		100	
minimum no. of samples required to split a node	2		2		2		2	

following experiment was conducted. The training sets of NSL-KDD, IDS 2017 and IDS 2018 datasets were split in a stratified fashion (preserving class proportions) into 10%, 20%, …100% subsets. ANN models (shallow: 64 nodes in hidden layer) were trained on these subsets (10%–100%) of the original training set and evaluated on the test sets.

4.2.4. Compute environment

The experiments were carried out on several PCs with and without using GPU acceleration. The time measurements (training and testing time) were obtained on a machine with the following specifications.

- Processors: AMD Ryzen 9 3900X CPU @ 3.80 GHz, 12-cores (24 threads)
- Memory (RAM): 64 GB
- GPU: 2 x Nvidia GeForce RTX 2080 Ti (11 GB) GPUs
- Operating system: Ubuntu 18.04

4.3. Results and analysis

Validating the benchmark implementation. The evaluation results of the 36 model-dataset combinations (Table 10) closely approximate the best results reported in the recent literature. The full set of results is given in Tables B.14, B.15, B.16 and B.17 and a comparison of results with prior works is given in Table 11. The deep feed-forward neural networks proposed by Vinayakumar et al. (2019) give similar results to our work (±3% difference in some metrics) on KDD 99, NSL-KDD and IDS 2017 datasets. The deep neural network suit (ANN, AE, DBN) by Ferrag et al. (2020) on the IDS 2018 dataset also yield similar results (±2.5% difference). In other model-dataset combinations, evaluation metrics from our experiments are in the same range as the values reported in the literature. This comparison validates the correctness of our benchmark implementation and verifies that the machine learning models are well-trained on the datasets. It must be noted that direct comparison is often difficult as many papers do not provide adequate details on the pre-processing steps performed on datasets (eg: removing new attack types in the test set that are not present in the training set, which gives more optimistic results), as well as the exact type of evaluation metric (eg: micro-averaged vs. weighted-macro-averaged precision, F1-score etc). Since the pre-processing steps and evaluation metrics in our study are uniform across all model evaluations in our study, the compilation of results allows for fair comparison of the selected deep learning models at the intrusion detection task. Models trained on IDS 2017 and IDS 2018 datasets give accuracy values in the range of 98.5%–99.8%, while those trained on NSL-KDD give accuracy figures in the 75.5%–77.5% range. This large difference is due to the fact that the test set of NSL-KDD is created with samples that were consistently misclassified by a set of 21 machine learning models (samples with high difficulty level: Tavallaee et al. (2009)).

Comparing model performance. The accuracy, precision and F1-score of the models on each dataset are illustrated in Fig. 3, and the false positive and false negative rates are illustrated in Fig. 4. The result set indicates that all the trained models are generally capable of the intru-

Table 11
Comparison of model performance with prior works. (Acc = accuracy, Prec = precision, Rec = recall, FAR = false alarm rate, F1 = F1-score).

Model	NSL-KDD	KDD 99	IDS 2017	IDS 2018
ANN	Vinayakumar et al. Acc = 78.5%, Prec = 81.0%, F1 = 76.5%	Vinayakumar et al. Acc = 92.5%, Prec = 93.4%, F1 = 92.1%	Vinayakumar et al. Acc = 95.6%, Prec = 96.2%, F1 = 95.7%	Ferrag et al. Acc = 97.28%
	Ours: Acc = 77.26%, Prec = 81.31%, F1 = 73.32%	Ours: Acc = 92.40%, Prec = 93.85%, F1 = 90.51%	Ours: Acc = 99.58%, Prec = 99.56%, F1 = 99.54%	Ours: Acc = 98.38%
AE	Javaid et al. Acc = 75%, Prec = 86%, F1 = 72%	Shone et al. Acc = 97.85%, Prec = 99.99%, F1 = 98.15%	Ours: Acc = 98.55%, Prec = 98.53%, F1 = 98.51%	Ferrag et al. Acc = 97.37%
	Ours: Acc = 76.94%, Prec = 80.51%, F1 = 73.62%	Ours: Acc = 92.25%, Prec = 92.74%, F1 = 90.40%		Ours: Acc = 98.22%
DBN	Alom et al. (40% subset of NSL-KDD) Acc = 97.45%	Gao et al. Acc = 93.49%, FAR = 0.76%	Ours: Acc = 99.56%, Prec = 99.54%, F1 = 99.52%	Ferrag et al. Acc = 97.3%
	Ours: Acc = 76.93%, Prec = 80.97%, F1 = 73.12%	Ours: Acc = 92.31%, Prec = 93.81%, FAR = 1.99%		Ours: Acc = 98.31%
LSTM	Yin et al. Acc = 81.29%	Le et al. Acc = 97.54%, Prec = 97.69%, FAR = 9.98%	Ours: Acc = 99.57%, Prec = 99.55%, F1 = 99.54%	Ferrag et al. Acc = 96%
	Ours: Acc = 77.26%, Prec = 80.40%, F1 = 73.71%	Ours: Acc = 92.32%, Prec = 93.12%, FAR = 2.17%		Ours: Acc = 98.88%
RF	Vinayakumar et al. Acc = 75.3%, Prec = 81.4%, F1 = 71.5%	Vinayakumar et al. Acc = 92.5%, Prec = 94.4%, F1 = 91.8%	Vinayakumar et al. Acc = 94.4%, Prec = 97.0%, F1 = 95.3%	Ferrag et al. Acc = 92%
	Ours: Acc = 75.80%, Prec = 80.18%, F1 = 71.21%	Ours: Acc = 92.34%, Prec = 94.05%, F1 = 90.34%	Ours: Acc = 99.86%, Prec = 99.86%, F1 = 99.85%	Ours: Acc = 98.34%

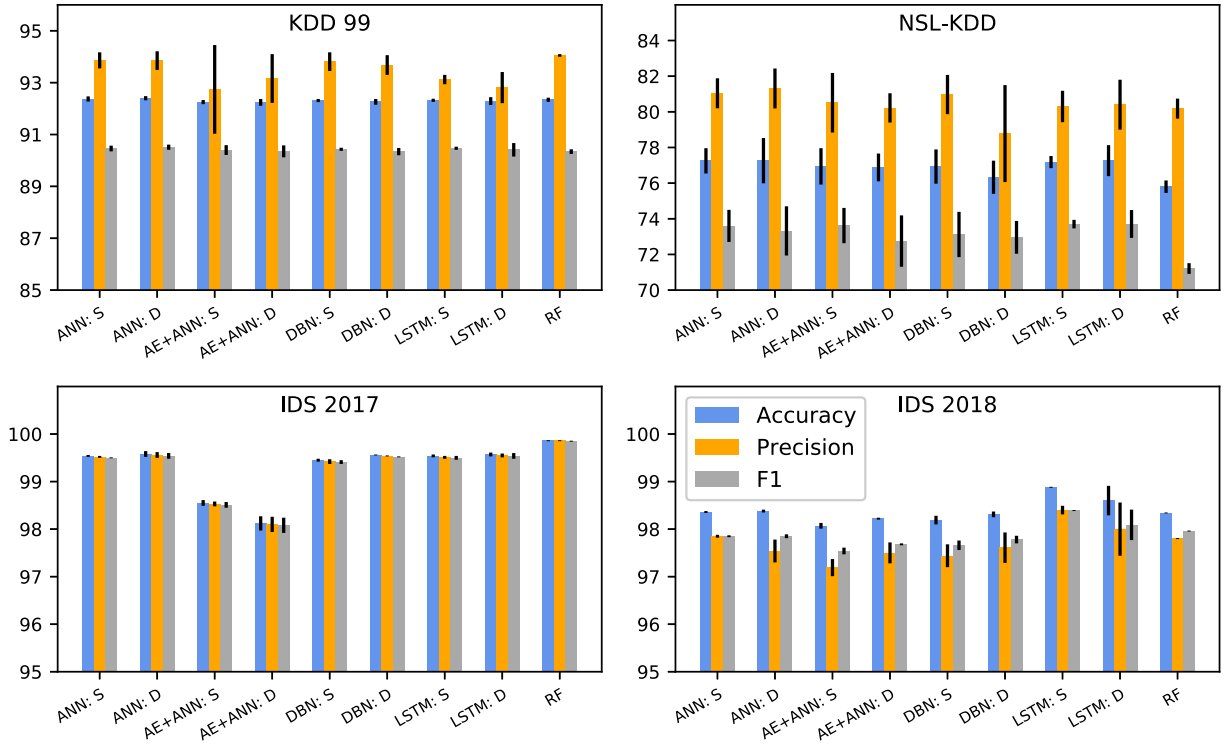

Fig. 3. Accuracy, precision and F1-score of the models on each dataset. 'S' and 'D' suffixes stand for 'Shallow' and 'Deep' respectively. The error bars indicate±one standard deviation of the error distribution obtained from 5 repetitions of training and evaluation.

sion detection classification task on these datasets. On each dataset, there is minimal variation of evaluation metrics between the models. In order to check the statistical significance of the variations in the metrics, a Friedman's test followed by a Nemenyi test was conducted on the accuracy figures of all models on the four datasets. These are two recommended statistical tests for comparing the performance multiple classifiers on multiple datasets (Demar, 2006). Friedman's test yielded a p-value of 0.0120, which results in a rejection of the null hypothesis that all models have equal accuracy (with a significance threshold of 0.05). Therefore, a post-hoc pairwise multiple comparison Nemenyi test was done, which has a null hypothesis that performance of each pair of models is equal (p-values are given in Table B.18). The null hypothesis is rejected (p-value below 0.05 significance threshold) in only one case, when the deep ANN model is compared with the deep AE model. This implies that the deep ANN model performs better than the deep AE model in a statistically significant sense. In all other model pair comparisons, the null hypothesis cannot be rejected, which suggests that their performance differences cannot be asserted with statistical significance. Similar results were observed when the statistical tests were performed for false negative rate and F1-score. It is worth noting that on any of the datasets, the semi-supervised models do not outperform other models along any of the metrics (a separate experiment was carried out to compare ANNs with semi-supervised models and its results are discussed later). The random forest (RF) gives high accuracy and

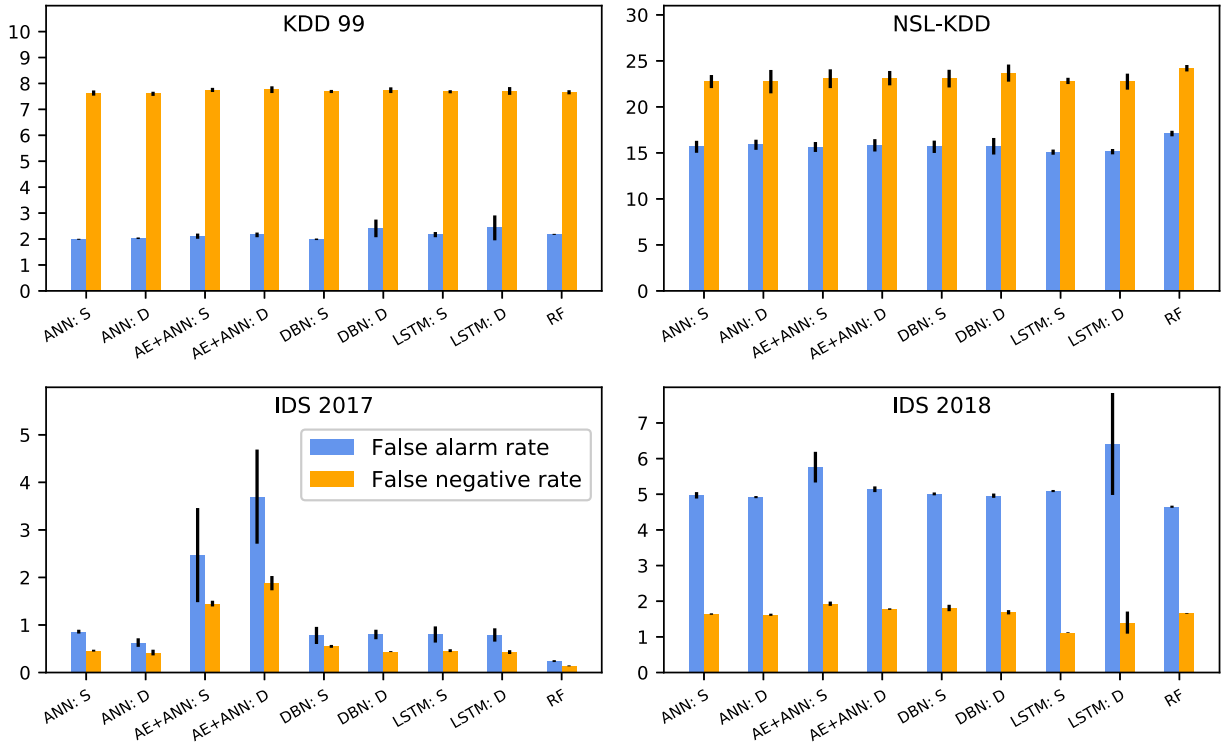

Fig. 4. False positive and false negative rates of the models on each dataset. 'S' and 'D' suffixes stand for 'Shallow' and 'Deep' respectively. The error bars indicate±one standard deviation of the error distribution obtained from 5 repetitions of training and evaluation.

F1-score on IDS 2017 and IDS 2018 datasets. RFs have been a popular classical machine learning model in the intrusion detection literature. The results from our study suggest that RFs perform well on recent intrusion detection datasets, and it is a suitable model to consider along with other machine learning models in future evaluations.

Performance of LSTM models. The purpose of evaluating an LSTM network for the intrusion detection task was to study the ability of sequence models to capture temporal patterns in the flows that may improve the classification. The experiment results (Figs. 3 and 4 and significance test results in Table B.18) indicate that the LSTM models trained on these datasets do not outperform the other models. They yield high false negative rates for attack classes that are time-correlated by their nature, such as denial of service and brute force style attacks. Increasing the sequence length (number of look-back time steps) of an LSTM did not improve the evaluation metrics. A reason for this could be that a single flow, which is a window of related packets, captures the significant packet-level time dependencies in flow features such as arrival rate, no. of packets or bytes in a flow etc. Inter-flow relationships may not contain useful dependencies learn-able by a sequence model like an LSTM network. The capability of sequence models at the intrusion detection task needs to be further investigated.

Computational cost of training and inference. The number of training epochs and time in minutes taken to achieve the average goal accuracy of a model, and test (inference) time to classify a batch of 1 million instances are given in the last three columns of Tables B.14, B.15, B.16 and B.17, and illustrated in Fig. 5 for the NSL-KDD and IDS 2017 datasets. Feed-forward neural networks (ANN) have the fastest training times across the four datasets. The semi-supervised models (AE + ANN and DBN + ANN) are generally slow to train (1.5x to 10x slower than ANNs) due to the computationally expensive unsupervised pre-training phase (10 epochs on the IDS 2018 dataset and 50 epochs on others). In comparison, the ANNs converge to their average accuracy with a small number of training epochs compared to the other neural network models. For inference (classifying instances), ANNs are fast across all datasets, and AE + ANN, LSTM and RF models are the slowest. This is likely due to the fact that features must be transformed by the encoder in the AE + ANN model, and model complexity of unrolled LSTM networks is high. Random forests have slow inference time (5x to 10x slower than ANNs), which make them undesirable for deployment in a real-time intrusion detection system.

This general comparison of the five models indicate that feed-forward neural networks (ANNs: shallow and deep variants) yield desirable evaluation metrics (accuracy, F1-score, training and inference time etc.) across all four intrusion detection datasets. Random forests show comparable performance and since they are easy to implement and fast to train, they are a suitable model to consider in the intrusion detection model building process (for initial results on new datasets and as a potential baseline for more complex models). LSTM networks and semi-supervised models (AE + ANN, DBN + ANN) do not show improvement over the ANNs.

Comparing semi-supervised methods with the supervised feed-forward neural network (ANN). To verify the finding from the initial evaluations that semi-supervised models (AE + ANN and DBN + ANN) do not show improvement over the ANNs, an additional set of experiments was carried out where the semi-supervised models were pre-trained with a data advantage (unlabeled) of 75% the size of the original dataset. (see section 4.2.3). The results of the experiments are given in Table 12 and illustrated in Fig. 6. In all cases, ANNs achieve higher or similar performance to the semi-supervised models. 10-fold cross validation *t*-tests were performed to verify any statistical significance in differences of accuracy (null hypothesis: two compared models have equal accuracy) and the results are given in Table 13. The results show that either the ANN model performs better (null hypothesis is rejected in favor of ANNs) or the accuracy differences are not significant. Another consideration is that the semi-supervised models are slow to train due to the expensive pre-training on a large chunk of the dataset (1.5x to 3.5x slower than ANN). Fig. 7 shows the validation error during the fine-tuning stage of the three models (after the pre-training stage). The DBN + ANN model error at the start of the fine-tuning process is very high in comparison to the other two methods, and it has a slower rate

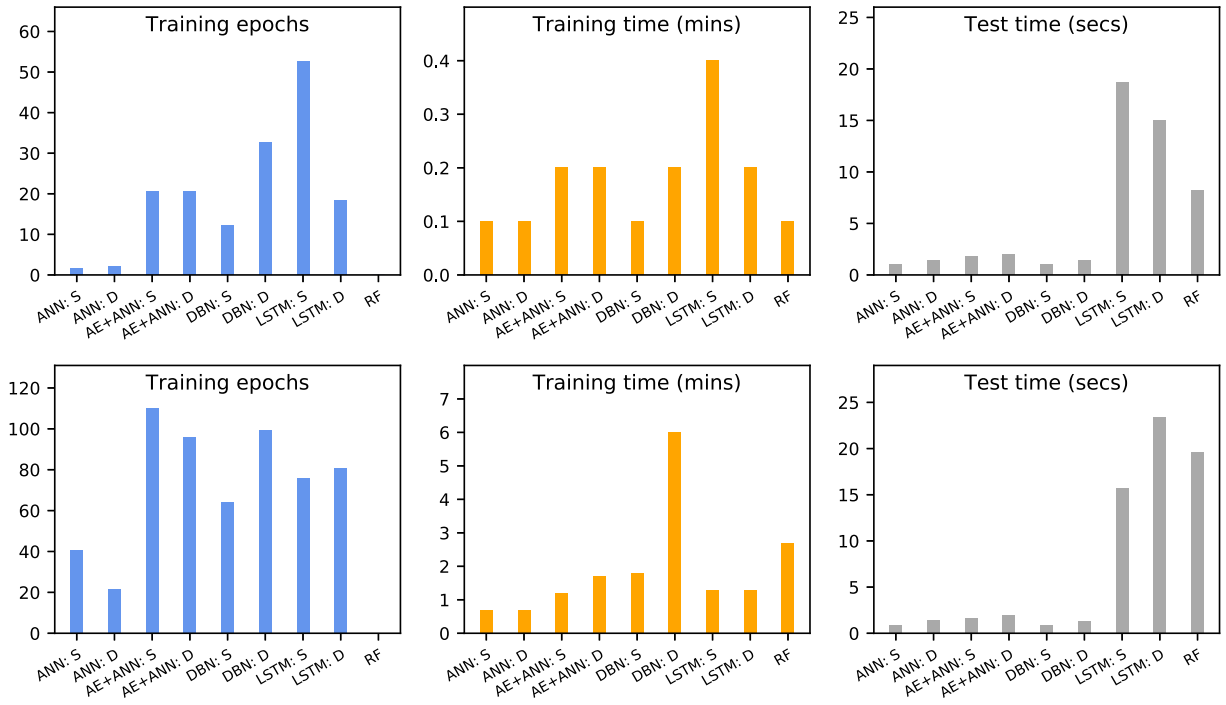

Fig. 5. Training epochs (left) and training time in minutes (middle) to achieve the average goal accuracy of a model, and test (inference) time to classify a batch of 1 million instances (right). Top row: NSL-KDD dataset, bottom row: IDS 2017 dataset.

Table 12
Comparing the performance of semi-supervised models (AE + ANN, DBN + ANN) with supervised ANN. The results are the mean of weighted macro average of metrics over 3 runs.

Dataset	Model	Precision	Detection rate	False alarm rate	False negative rate	F1	Time to train (min)
NSL-KDD	ANN	0.811	0.7814	0.1502	0.2186	0.7438	0.2
	AE + ANN	0.797	0.7693	0.151	0.2307	0.7339	0.5
	DBN + ANN	0.7894	0.7686	0.1577	0.2314	0.7336	1.2
IDS 2017	ANN	0.9943	0.9945	0.0107	0.0055	0.9941	3
	AE + ANN	0.9941	0.9943	0.0105	0.0057	0.9939	5.8
	DBN + ANN	0.9914	0.992	0.0198	0.008	0.9915	14.9
IDS 2018	ANN	0.9779	0.9827	0.0495	0.0173	0.9776	8.9
	AE + ANN	0.9734	0.9815	0.052	0.0185	0.9763	7.6
	DBN + ANN	0.9782	0.9833	0.0502	0.0167	0.9782	10.5

of convergence (in the IDS 2017 case). This indicates that taking DBN-pre-trained weights as the initial point for the fine-tuning stage does not help it achieve better performance or converge faster. In contrast, the random-initialized ANN model starts with a lower validation loss, and converges faster to its minimum error. Similar results were obtained with a 50% unlabeled, 50% labeled split of the datasets. The observation that semi-supervised models bring no improvement over feed-forward neural networks aligns with research trends in the broader machine learning community. The following paragraph gives a summary of relevant works.

The greedy layer-wise pre-training of a deep belief network on unlabeled data is expected to learn weights for all layers that will provide a good initial point in the parameter space for the supervised fine-tuning step (Erhan et al., 2010). An autoencoder learns a non-linear feature transformation from unlabeled data, which can then be input to a supervised learner (Baldi, 2012). These models were intended to address issues in the difficult optimization problem of training deep architectures, utilizing abundantly available unlabeled data. However, techniques including activation functions such as the rectified linear unit (ReLU: Glorot et al., 2011) and scaled exponential linear units (SELU: Klambauer et al., 2017), better random weight initialization schemes (He et al., 2015; Glorot and Bengio, 2010), batch normalization (Ioffe and Szegedy, 2015), regularization methods like dropout (Srivastava et al., 2014), and efficient GPU implementations that allow longer training have successfully addressed the difficulties in training deep neural networks. Researchers report unsupervised pre-training brings no improvement over pure supervised training of deep feed-forward neural networks (DNN) (Glorot et al., 2011; Bengio et al., 2013). Therefore, computationally expensive unsupervised pre-training methods like deep belief networks and the autoencoders have largely been replaced by supervised training of deep feed forward neural networks with labeled datasets, helped by transfer learning. The results from our experiments on the intrusion detection datasets agree with the above observations made by the broader machine learning community on other tasks.

Neural network architecture search. The results of the network architecture search experiments are shown in Tables B.19, B.20 and B.21. In deep neural networks with the same number of nodes (64) in each layer (Table B.20), the depth of the network showed no correlation with the F1-score or other metrics on any of the three datasets. However, in deep networks with a narrowing architecture (Table B.21), accuracy and F1-scores improved with depth on the IDS 2017 and IDS 2018 datasets, as

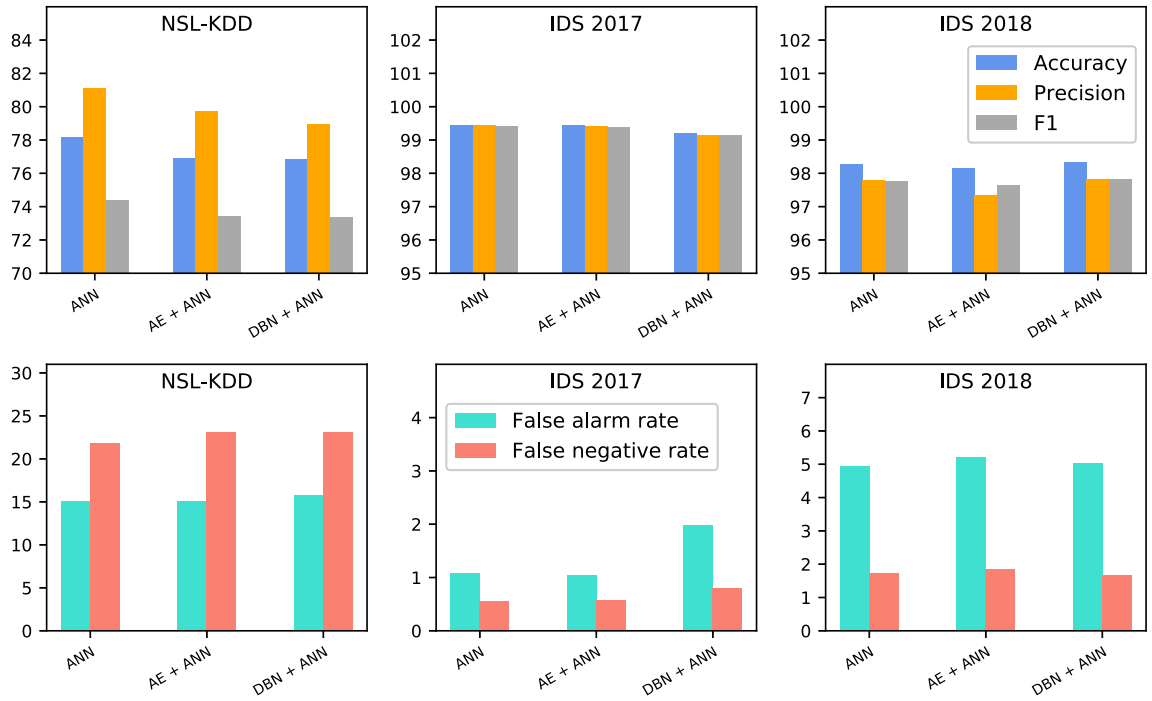

Fig. 6. Results of training semi-supervised models AE + ANN and DBN + ANN (unsupervised pre-training on 75% of original training set followed by supervised fine-tuning on the other 25%) and ANN (supervised training on 25% of original training set).

Table 13
p-values of the 10-fold cross validation *t*-tests comparing ANNs with semi-supervised models.

	NSL-KDD	IDS 2017	IDS 2018
ANN vs. AE + ANN	2.7 × 10(-5)	0.45	0.8405
ANN vs. DBN + ANN	1.8 × 10(-5)	1.8 × 10(-17)	0.9911

illustrated in Fig. 8. For example, the F1-score on the IDS 2017 dataset improves from 99.15% with a single-hidden-layer network to 99.61% with a 6-hidden-layer-network. The statistical significance of the performance improvement was verified with a paired *t*-test of a 10-fold cross validation (rejection of the null hypothesis that the two models have equal accuracy with p-value = 6.5×10^{-20}). This improvement corresponds to a drop in false negatives from 2248 to 820 (a 63% reduction), which can be significant in a critical task like intrusion detection. Shallow and wide networks (Table B.19) also produced improved evaluation metrics very similar to the deep and narrow architectures. The accuracy and F1-score of shallow and wide models are shown as the last 3 data points on the graphs in Fig. 8. This improvement is likely due to the large number of parameters in these networks, which increases their learning capacity. However, it also means that training these large networks is computationally very expensive. For example, training the 10-layer network in Table B.21 takes 15x the time it takes to train the single-layer network on the IDS 2017 dataset. Adding one layer in the narrowing architectures increases training time by 16% on average. Similarly, training shallow but wide architectures take long training times due to the large number of parameters. Therefore, researchers and practitioners must consider the trade-offs between accuracy, model complexity and training and inference time when selecting the models that go into intrusion detection systems.

Measuring the effect of training set size on ANN performance. The evaluation results of training a shallow ANN model on increasingly large datasets are given in Table B.22. Evaluation metrics on the NSL-KDD and IDS 2018 test set do not show a correlation with the training set size. This suggests that smaller, stratified subsets of the NSL-KDD and IDS 2018 training sets contain the information required for training models that can reach the accuracy of models trained with the full dataset. It also indicates that practitioners can use stratified small subsets of large intrusion detection datasets for fast experimentation during the model building process. Results on the IDS 2017 test set show that increasing the training set size improves the evaluation metrics of the ANN models (Fig. 9). The number of false negatives drop by 59.98% when the ANN model is trained with the full IDS 2017 training set as opposed to when it is trained with 10% of the training set. The statistical significance of the performance improvement was verified with a paired *t*-test of a 10-fold cross validation (rejection of the null hypothesis that the two models have equal accuracy with p-value = 8.9×10^{-10}). Assuming that the IDS 2017 dataset is representative of normal and attack traffic in a modern network, this result suggests that it is possible to build more accurate neural network models for the intrusion detection task by collecting large amounts of representative network data.

Results on the IDS 2018 dataset. All models trained on the IDS 2018 dataset yield very similar evaluation metrics (over 98.3% accuracy and 97.8% F1-score). Along with results reported by Ferrag et al. (2020), these evaluations can be used as a basis for further study of this large and realistic dataset.

A summary of the conclusions that can be drawn from the results of our empirical study is listed below.

- All five evaluated models perform generally well on all four intrusion detection datasets. Considering performance (accuracy, F1-score, false negatives), training time and inference time, deep feed-forward neural networks (ANN) are the more capable model in the benchmark.
- Semi-supervised models (AE + ANN, DBN + ANN) do not perform better than other purely supervised ANNs even when they have a substantial data advantage (unlabeled). They are also slow to train due to unsupervised pre-training required.
- LSTM networks operating on flow sequences do not show improved performance for highly time-correlated attacks like denial of service or brute force. Additionally, they have inference times due to the network complexity.

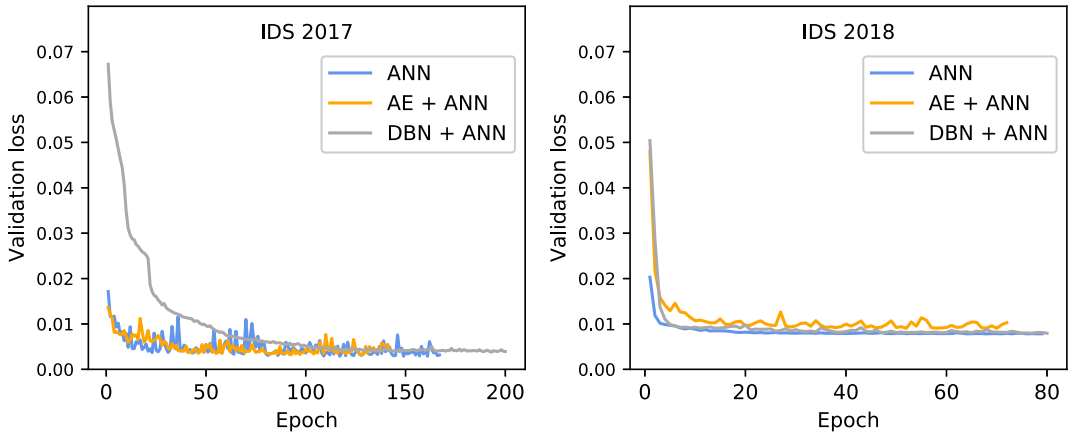

Fig. 7. Validation loss history during the supervised training stage of feed-forward neural network (ANN), autoencoder (AE + ANN) and deep belief network (DBN + ANN) models.

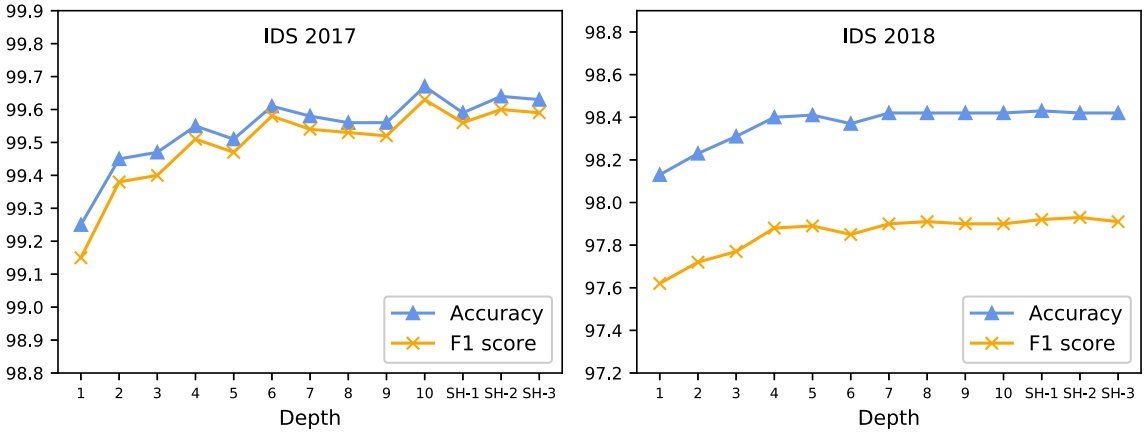

Fig. 8. Accuracy and F1-score on IDS 2017 and IDS 2018 datasets for ANNs with deep and narrowing architectures (first 10 data points in the graphs), and shallow and wide architectures (last 3 data points in graphs labeled SH-1, SH-2 and SH-3). The architectures are given in Tables B.19 and B.21.

- Random forests perform well, especially on the IDS 2017 and IDS 2018 datasets. However, they are slow during inference. Since they are easy to implement and fast to train, random forests can be used as a potential baseline model when building machine learning models for intrusion detection.

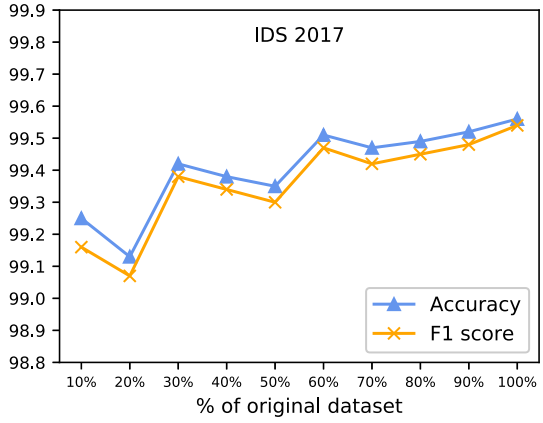

Fig. 9. Evaluation metrics from shallow ANN models trained on increasingly large subsets of the IDS 2017 dataset. Models trained with larger training sets show high accuracy and F1-score.

- Narrowing deep neural network architectures (large number of nodes in the first hidden layer, narrowing down to a small number of nodes in the final hidden layer) show improved performance than small networks (shallow and narrow) on IDS 2017 and IDS 2018 datasets.
- ANN performance can be improved by training it with more network flow data, which suggests that efforts to collect more data for network intrusion detection will be worthwhile.

5. Issues in current research and future directions

The extensive literature survey that we conducted revealed several common weaknesses in research papers. This section discusses these issues, and highlights a number of potential future research avenues based on our observations.

- *Issues in evaluation of novel methods.* Many papers on the topic of machine learning methods for intrusion detection use only older datasets (mostly, KDD 99 and NSL-KDD) for training and evaluating the models. These datasets are not representative of modern network traffic and intrusions, and only using these datasets for validating novel methods is not adequate. This issue has been discussed in detail by Sommer and Paxson (2010). We recommend that researchers build and validate their models on recent intrusion detection datasets (see Ring et al. (2019) for a survey of intrusion detection datasets). Comprehensive analyses of datasets, their

differences and understanding how machine learning models may generalize across datasets will provide useful information to make better decisions on new research. The literature also lacks reports on evaluating intrusion detection methods in real-life attack scenarios (eg: advanced persistent threats and data exfiltration), and comparisons of traditional signature-based intrusion detection and machine learning methods in these cases. New results in this area can be valuable to the intrusion detection and machine learning research communities.

- *Issue of reproducibility*. The inability to reproduce results of published research hinders building upon those ideas. In our survey, we found that many papers on deep learning for intrusion detection do not report adequate information on their methodology to fully understand their work (model details, operations on data etc). The machine learning research community has identified this problem, and now places a high importance on reproducibility (eg: a guide on reproducibility of machine learning research by Pineau (2019)). In our empirical analysis, we found that adhering to such a guide helps with conducting and reporting reproducible experiments. We have also open-sourced our code, which promotes transparency and allows researchers to use it as a reference point for their work. We encourage researchers in intrusion detection to consider the aspect of reproducibility when designing and reporting their systems and methods.
- *Semi-supervised machine learning methods for intrusion detection.* Since large quantities of unlabeled network flow data can be acquired from operating networks with relatively little effort, semi-supervised machine learning models that make use of this unlabeled data for the intrusion detection problem have been actively researched. The most frequent models in the literature are autoencoders and deep belief networks. In our empirical study, we show that these models do not perform better than feed-forward neural networks trained in a supervised fashion on labeled data. This observation aligns with the general trends in the machine learning community. Therefore, researchers in the intrusion detection domain may find it more fruitful to explore new methods in unsupervised and semi-supervised machine learning.
- *Exploring the use of novel machine learning paradigms and methods.* The field of machine learning has seen many advances in the recent years, some of which may be used to address problems in the intrusion detection domain. For example, transfer learning methods will allow practitioners to download pre-trained neural network intrusion detectors and adapt them to a new network environment with minimal supervised training on new data. Techniques such as the attention mechanism may be effective in detecting long-term attacks. Neural Architecture Search methods can be employed to find neural networks that are optimal under a set of metrics (eg: minimum false negatives, low model complexity and high inference throughput) for different intrusion detection settings. Research on interpretable machine learning could be used to explain the classifications of an intrusion detection system. For example, identifying which features in a flow record or set of packets were significant (Montavon et al., 2018) for an attack detection may provide useful information to a security analyst for investigating a threat incident (eg: root cause analysis and triage). In the case of misclassifications or overfitting, such explanations can be useful to the system developer to identify issues or weaknesses of machine learning models (eg: reliance on spurious feature correlations: Lapuschkin et al. (2019)) and improve them.
- *Open-source machine learning based intrusion detection systems (IDS).* There are several popular rule-based open-source IDS projects (such as Zeek and Snort) that have contributed to research and development in the field of rule-based intrusion detection. However, no such projects exist for machine learning based IDS. An open-source IDS with machine learning detectors will allow researchers to quickly deploy an IDS in a network environment and benchmark their models against a repository of machine learning based intrusion detectors. It may also help in fostering collaboration and attracting new researchers to the field.

6. Conclusion

Attacks against computer networks are a growing threat and intrusion detection systems perform the critical task of detecting them and alerting the security teams. Machine learning algorithms have become a viable option for network intrusion detection, as they are capable of learning intrusion patterns from large datasets. In this study, we present the results of a broad literature survey on the topic of deep learning models for network intrusion detection, and a set of experiments comparing the performance of four key deep learning models on four intrusion detection datasets. We have made the implementation and the complete set of results publicly available, so that researchers may build upon it.

The results show that supervised deep feed-forward neural networks (ANN) perform well on all four datasets across all metrics (accuracy, F1-score, false negatives, training and inference time). Two popular semi-supervised learning models, autoencoders and deep belief networks do not perform better than supervised feed-forward neural networks. The accuracy of ANNs increase as they are trained on larger datasets, which suggests that efforts to build large datasets with labeled flows of benign and attack traffic will be a worthwhile investment.

Our survey revealed several weaknesses in current research such as model evaluation performed only on legacy datasets and inadequate details of models leading to experiments and results that cannot be replicated. Several potential research avenues in the field of machine learning for intrusion detection have been suggested. This survey aims to provide a bird's-eye view of the field of deep learning methods for intrusion detection, along with reproducible empirical results that researchers can rely on.

The set of experiments in this study were designed and conducted to identify the capability of different machine learning models in the intrusion detection task and to give researchers a bird's-eye view of the field. However, some limitations exist in the study due to resource constraints. We have evaluated the models on only four datasets, which lack variety in terms of network traffic patterns and the attacks included in them. Furthermore, all models in the benchmark classify flow records, and no analysis is performed at the packet-level. For future work, we plan to extend the benchmark by training and evaluating these models on more large intrusion detection datasets. Other varieties of recurrent neural networks will also be added to the model repository.

CRediT authorship contribution statement

Sunanda Gamage: Conceptualization, Methodology, Software, Validation, Formal analysis, Writing - original draft, Visualization. **Jagath Samarabandu:** Conceptualization, Resources, Writing - review & editing, Supervision, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors are grateful to Darshil Desai (darshildesai99@gmail.com) for his assistance in categorizing, reading and summarizing a number of research papers for the survey. This work was supported in part by The Natural Sciences and Engineering Research Council of Canada (NSERC) [grant number RGPIN-2017-04755].

Appendix A. Evaluation metrics

The following evaluation metrics are calculated based on the values in the confusion matrix (Table 3).

- Overall accuracy: the proportion of correctly classified instances out of a given set of instances. When the classes are heavily imbalanced, this metric is less useful.

$$accuracy = \frac{TP + TN}{all\ instances}$$

- Precision or positive predictive value (PPV): the proportion of correct classifications out of a set of predictions classified as attacks.

$$precision = \frac{TP}{TP + FP}$$

- Recall, sensitivity, true positive rate or detection rate: the proportion of correct attack classifications out of a given set of attack instances. This metric represents the attack detection capability of the model.

$$recall = \frac{TP}{TP + FN}$$

- Specificity or true negative rate: the proportion of correct benign classifications out of a given set of benign instances.

$$specificity = \frac{TN}{TN + FP}$$

- False positive rate (FPR) or Fall out: the proportion of instances classified incorrectly as attacks out of a given set of benign instances. An intrusion detection system with a high false positive rate is practically less useful, as it generates a flood of false alarms, overwhelming security operators.

$$FPR = \frac{FP}{TN + FP} = 1 - specificity$$

- False negative rate (FNR) or miss rate: the proportion instances classified incorrectly as benign out of a given set of attack instances. Note that in an IDS, a false negative corresponds to an attack that was not detected by the system; an event that can have severe consequences.

$$FNR = \frac{FN}{TP + FN} = 1 - recall$$

- F1 score: the harmonic mean of precision and recall. For class-imbalanced problems, which is usually the case with intrusion datasets, this is a useful metric to evaluate models.

$$F1 = 2.\frac{precision\ .\ recall}{precision + recall}$$

Appendix B. Results of experiments

Table B.14
Overall results (weighted macro average of metrics) of the models on the KDD 99 test set in the format: mean (std_dev) from 5 repetitions.

Model	Accuracy	Precision	Detection Rate	False alarm rate	False negative rate	F1	Train epochs	Train time (min)	Test time (sec)
ANN: shallow	0.9237	0.9386	0.9237	0.0199	0.0763	0.9046	2.0	0.1	1.0
	(0.0010)	(0.0031)	(0.0010)	(0.0002)	(0.0010)	(0.0011)	(1.00)	(0.05)	(0.01)
ANN: deep	0.9240	0.9385	0.9240	0.0203	0.0760	0.9051	1.7	0.1	1.4
	(0.0008)	(0.0036)	(0.0008)	(0.0003)	(0.0008)	(0.0010)	(0.58)	(0.03)	(0.04)
AE + ANN: shallow	0.9225	0.9274	0.9225	0.0211	0.0775	0.9040	11.0	1.0	1.7
	(0.0008)	(0.0171)	(0.0008)	(0.0010)	(0.0008)	(0.0019)	(0.00)	(0.00)	(0.06)
AE + ANN: deep	0.9224	0.9316	0.9224	0.0216	0.0776	0.9035	15.7	0.7	1.7
	(0.0013)	(0.0094)	(0.0013)	(0.0009)	(0.0013)	(0.0023)	(0.00)	(0.00)	(0.91)
DBN + ANN: shallow	0.9231	0.9381	0.9231	0.0199	0.0769	0.9043	12.7	2.6	1.0
	(0.0006)	(0.0036)	(0.0006)	(0.0003)	(0.0006)	(0.0006)	(0.58)	(0.02)	(0.04)
DBN + ANN: deep	0.9226	0.9368	0.9226	0.0241	0.0774	0.9034	39.0	6.2	1.5
	(0.0011)	(0.0038)	(0.0011)	(0.0034)	(0.0011)	(0.0014)	(4.58)	(0.38)	(0.04)
LSTM: shallow	0.9232	0.9312	0.9232	0.0217	0.0768	0.9047	7.0	0.4	18.8
	(0.0006)	(0.0018)	(0.0006)	(0.0010)	(0.0006)	(0.0006)	(4.36)	(0.22)	(0.00)
LSTM: deep	0.9229	0.9281	0.9229	0.0243	0.0771	0.9041	17.3	0.9	26.0
	(0.0015)	(0.0060)	(0.0015)	(0.0048)	(0.0015)	(0.0026)	(16.65)	(0.83)	(0.72)
RF	0.9234	0.9405	0.9234	0.0218	0.0766	0.9034		2.0	7.7
	(0.0008)	(0.0005)	(0.0008)	(0.0002)	(0.0008)	(0.0009)		(0.00)	(0.00)

Table B.15
Overall results (weighted macro average of metrics) of the models on the NSL KDD test set in the format: mean (std_dev) from 5 repetitions.

Model	Accuracy	Precision	Detection Rate	False alarm rate	False negative rate	F1	Train epochs	Train time (min)	Test time (sec)
ANN: shallow	0.7725	0.8104	0.7725	0.1566	0.2275	0.7360	1.7	0.1	1.0
	(0.0071)	(0.0084)	(0.0071)	(0.0065)	(0.0071)	(0.0090)	(0.58)	(0.00)	(0.02)
ANN: deep	0.7726	0.8131	0.7726	0.1588	0.2274	0.7332	2.0	0.1	1.4
	(0.0127)	(0.0112)	(0.0127)	(0.0055)	(0.0127)	(0.0138)	(0.00)	(0.00)	(0.03)
AE + ANN: shallow	0.7694	0.8051	0.7694	0.1564	0.2306	0.7362	20.7	0.2	1.8
	(0.0102)	(0.0167)	(0.0102)	(0.0054)	(0.0102)	(0.0099)	(4.62)	(0.04)	(0.03)
AE + ANN: deep	0.7688	0.8022	0.7688	0.1582	0.2312	0.7275	20.6	0.2	2.0
	(0.0078)	(0.0082)	(0.0078)	(0.0067)	(0.0078)	(0.0144)	(2.65)	(0.02)	(0.04)
DBN + ANN: shallow	0.7693	0.8097	0.7693	0.1566	0.2307	0.7312	12.3	0.1	1.0
	(0.0096)	(0.0110)	(0.0096)	(0.0067)	(0.0096)	(0.0127)	(0.58)	(0.00)	(0.01)
DBN + ANN: deep	0.7633	0.7878	0.7633	0.1572	0.2367	0.7296	32.7	0.2	1.4
	(0.0093)	(0.0272)	(0.0093)	(0.0090)	(0.0093)	(0.0092)	(0.58)	(0.00)	(0.04)
LSTM: shallow	0.7718	0.8030	0.7718	0.1508	0.2282	0.7370	52.7	0.4	18.7
	(0.0034)	(0.0088)	(0.0034)	(0.0028)	(0.0034)	(0.0024)	(22.85)	(0.19)	(0.93)
LSTM: deep	0.7726	0.8040	0.7726	0.1513	0.2274	0.7371	18.3	0.2	15.0
	(0.0087)	(0.0140)	(0.0087)	(0.0029)	(0.0087)	(0.0078)	(6.81)	(0.06)	(0.12)
RF	0.7580	0.8018	0.7580	0.1710	0.2420	0.7121		0.1	8.2
	(0.0035)	(0.0056)	(0.0035)	(0.0030)	(0.0035)	(0.0030)		(0.00)	(0.00)

Table B.16
Overall results (weighted macro average of metrics) of the models on the IDS 2017 test set in the format: mean (std_dev) from 5 repetitions.

Model	Accuracy	Precision	Detection Rate	False alarm rate	False negative rate	F1	Train epochs	Train time (min)	Test time (sec)
ANN: shallow	0.9954	0.9952	0.9954	0.0086	0.0046	0.9950	40.7	0.7	0.9
	(0.0002)	(0.0002)	(0.0002)	(0.0004)	(0.0002)	(0.0001)	(17.04)	(0.28)	(0.01)
ANN: deep	0.9958	0.9956	0.9958	0.0063	0.0042	0.9954	21.3	0.7	1.4
	(0.0006)	(0.0006)	(0.0006)	(0.0009)	(0.0006)	(0.0006)	(5.13)	(0.17)	(0.04)
AE + ANN: shallow	0.9855	0.9853	0.9855	0.0247	0.0145	0.9851	110.0	1.2	1.6
	(0.0006)	(0.0005)	(0.0006)	(0.0099)	(0.0006)	(0.0006)	(0.00)	(0.00)	(0.02)
AE + ANN: deep	0.9812	0.9810	0.9812	0.0370	0.0188	0.9808	96.0	1.7	1.9
	(0.0015)	(0.0016)	(0.0015)	(0.0099)	(0.0015)	(0.0016)	(28.29)	(0.47)	(0.05)
DBN + ANN: shallow	0.9945	0.9942	0.9945	0.0078	0.0055	0.9941	64.3	1.8	0.9
	(0.0003)	(0.0005)	(0.0003)	(0.0018)	(0.0003)	(0.0004)	(17.04)	(0.28)	(0.02)
DBN + ANN: deep	0.9956	0.9954	0.9956	0.0080	0.0044	0.9952	99.3	6.0	1.3
	(0.0001)	(0.0001)	(0.0001)	(0.0010)	(0.0001)	(0.0001)	(17.24)	(0.29)	(0.03)
LSTM: shallow	0.9954	0.9951	0.9954	0.0080	0.0046	0.9950	76.0	1.3	15.7
	(0.0003)	(0.0003)	(0.0003)	(0.0017)	(0.0003)	(0.0004)	(20.78)	(0.35)	(0.21)
LSTM: deep	0.9957	0.9955	0.9957	0.0079	0.0043	0.9954	80.7	1.3	23.4
	(0.0004)	(0.0004)	(0.0004)	(0.0014)	(0.0004)	(0.0006)	(30.09)	(0.50)	(0.19)
RF	0.9986	0.9986	0.9986	0.0024	0.0014	0.9985		2.7	19.6
	(0.0001)	(0.0001)	(0.0001)	(0.0002)	(0.0001)	(0.0001)		(0.00)	(0.08)

Table B.17
Overall results (weighted macro average of metrics) of the models on the IDS 2018 test set in the format: mean (std_dev) from 5 repetitions.

Model	Accuracy	Precision	Detection Rate	False alarm rate	False negative rate	F1	Train epochs	Train time (min)	Test time (sec)
ANN: shallow	0.9836	0.9785	0.9836	0.0497	0.0164	0.9785	43.0	5.0	0.9
	(0.0002)	(0.0003)	(0.0002)	(0.0009)	(0.0002)	(0.0002)	(5.20)	(0.61)	(0.02)
ANN: deep	0.9838	0.9754	0.9838	0.0492	0.0162	0.9785	19.7	3.0	1.4
	(0.0003)	(0.0024)	(0.0003)	(0.0003)	(0.0003)	(0.0004)	(7.64)	(1.15)	(0.11)
AE + ANN: shallow	0.9807	0.9719	0.9807	0.0576	0.0193	0.9754	40.3	7.7	1.6
	(0.0006)	(0.0018)	(0.0006)	(0.0043)	(0.0006)	(0.0007)	(27.23)	(2.27)	(0.04)
AE + ANN: deep	0.9822	0.9750	0.9822	0.0514	0.0178	0.9768	29.8	6.5	1.9
	(0.0002)	(0.0022)	(0.0002)	(0.0008)	(0.0002)	(0.0002)	(2.31)	(0.27)	(0.10)
DBN + ANN: shallow	0.9819	0.9744	0.9819	0.0501	0.0181	0.9766	19.0	5.1	0.9
	(0.0009)	(0.0024)	(0.0009)	(0.0004)	(0.0009)	(0.0010)	(5.00)	(0.50)	(0.02)
DBN + ANN: deep	0.9831	0.9761	0.9831	0.0496	0.0169	0.9778	44.0	25.1	1.3
	(0.0006)	(0.0032)	(0.0006)	(0.0006)	(0.0006)	(0.0008)	(8.89)	(0.59)	(0.05)
LSTM: shallow	0.9888	0.9840	0.9888	0.0509	0.0112	0.9839	66.0	5.5	16.1
	(0.0001)	(0.0009)	(0.0001)	(0.0003)	(0.0001)	(0.0001)	(15.72)	(1.31)	(0.20)
LSTM: deep	0.9860	0.9800	0.9860	0.0641	0.0140	0.9809	46.0	3.8	24.0
	(0.0031)	(0.0056)	(0.0031)	(0.0143)	(0.0031)	(0.0032)	(31.10)	(2.59)	(0.14)
RF	0.9834	0.9780	0.9834	0.0465	0.0166	0.9796		5.6	25.5
	(0.0001)	(0.0001)	(0.0001)	(0.0003)	(0.0001)	(0.0001)		(0.01)	(0.10)

Table B.18
p-values of the Nemenyi pairwise multiple comparison test on the accuracy figures of models with the null hypotheses: performance of each pair of models in the matrix is equal. p-values below a 0.05 significance threshold (null hypothesis is rejected) are shown in bold.

	ANN: shallow	ANN: deep	AE + ANN: shallow	AE + ANN: deep	DBN + ANN: shallow	DBN + ANN: deep	LSTM: shallow	LSTM: deep	RF
ANN: shallow	−1	0.9	0.4971	0.3274	0.8106	0.8889	0.9	0.9	0.9
ANN: deep	0.9	−1	0.0736	**0.0341**	0.2553	0.3274	0.9	0.9	0.8889
AE + ANN: shallow	0.4971	0.0736	−1	0.9	0.9	0.9	0.5756	0.4118	0.8106
AE + ANN: deep	0.3274	**0.0341**	0.9	−1	0.9	0.9	0.4118	0.2553	0.6539
DBN + ANN: shallow	0.8106	0.2553	0.9	0.9	−1	0.9	0.8889	0.7323	0.9
DBN + ANN: deep	0.8889	0.3274	0.9	0.9	0.9	−1	0.9	0.8106	0.9
LSTM: shallow	0.9	0.9	0.5756	0.4118	0.8889	0.9	−1	0.9	0.9
LSTM: deep	0.9	0.9	0.4118	0.2553	0.7323	0.8106	0.9	−1	0.9
RF	0.9	0.8889	0.8106	0.6539	0.9	0.9	0.9	0.9	−1

Table B.19
Performance of shallow and wide neural network architectures. No. of parameters are calculated with 78 input units and 12 output units. (Acc = accuracy, Prec = precision, FNR = false negative rate, F1 = F1-score, T.T. = training time in minutes)

Name	No. of nodes in hidden layers	No. of params	IDS 2017				IDS 2018			
			Acc.	FNR	F1	T.T.	Acc.	FNR	F1	T.T.
SH-1	1000-500-200	680.4 K	0.9959	0.0041	0.9956	10.4	0.9843	0.0157	0.9792	20.9
SH-2	2000-1000-500	2.66 M	0.9964	0.0036	0.996	15.2	0.9842	0.0158	0.9793	32.1
SH-3	3000-1500-500	5.49 M	0.9963	0.0037	0.9959	30.4	0.9842	0.0158	0.9791	68.1

Table B.20
Performance of deep neural network architectures with the same no. of nodes in each layer. No. of parameters are calculated with 78 input units and 12 output units. (Acc = accuracy, Prec = precision, FNR = false negative rate, F1 = F1-score, T.T. = training time in minutes)

Depth	No. of nodes in hidden layers	No. of params	IDS 2017				IDS 2018			
			Acc.	FNR	F1	T.T.	Acc.	FNR	F1	T.T.
1	64	5,7 K	0.995	0.005	0.9946	4	0.9831	0.017	0.9777	7.7
2	64–64	9.9 K	0.9954	0.0046	0.9951	4.8	0.984	0.016	0.9789	9.8
3	64-64-64	14 K	0.9961	0.0039	0.9957	5.6	0.984	0.016	0.9788	8.5
4	64-64-64-64	18 K	0.9946	0.0054	0.9942	6	0.9836	0.0164	0.9782	8.9
5	64-64-64-64-64	22.1 K	0.9947	0.0053	0.9943	6.8	0.9832	0.0168	0.9778	11.5
6	64-64-64-64-64-64	26.2 K	0.9958	0.0042	0.9954	7.6	0.9841	0.0159	0.9789	17.1
7	64-64-64-64-64-64-64	30.3 K	0.9955	0.0045	0.9951	8.4	0.9829	0.0171	0.9775	7.1
8	64-64-64-64-64-64-64-64	34.4 K	0.9952	0.0048	0.9948	9.2	0.9837	0.0163	0.9783	19.7
9	64-64-64-64-64-64-64-64-64	38.6 K	0.9955	0.0045	0.9951	9.6	0.9832	0.0168	0.9778	18.9
10	64-64-64-64-64-64-64-64-64-64	42.7 K	0.9952	0.0048	0.9948	10.4	0.9837	0.0163	0.9785	23.1

Table B.21
Performance of narrowing deep neural network architectures. No. of parameters are calculated with 78 input units and 12 output units. (Acc = accuracy, Prec = precision, FNR = false negative rate, F1 = F1-score, T.T. = training time in minutes)

Depth	No. of nodes in hidden layers	No. of params	IDS 2017				IDS 2018			
			Acc.	FNR	F1	T.T.	Acc.	FNR	F1	T.T.
1	16	1.4 K	0.9925	0.0075	0.9915	4	0.9813	0.0187	0.9762	3.7
2	32–16	2.7 K	0.9945	0.0055	0.9938	4.8	0.9823	0.0177	0.9772	5
3	64-32-16	7.7 K	0.9947	0.0053	0.994	5.6	0.9831	0.0169	0.9777	9.5
4	128-64-32-16	20.1 K	0.9955	0.0045	0.9951	6	0.984	0.016	0.9788	10.7
5	256-128-64-32-16	63.7 K	0.9951	0.0049	0.9947	7.2	0.9841	0.0159	0.9789	15.8
6	400-256-128-64-32-16	177.3 K	0.9961	0.0039	0.9958	8.4	0.9837	0.0163	0.9785	10.8
7	500-400-256-128-64-32-16	385.1 K	0.9958	0.0042	0.9954	10.4	0.9842	0.0158	0.979	24.1
8	600-500-400-256-128-64-32-16	693 K	0.9956	0.0044	0.9953	12.8	0.9842	0.0158	0.9791	24.5
9	700-600-500-400-256-128-64-32-16	1.12 M	0.9956	0.0044	0.9952	15.6	0.9842	0.0158	0.979	33.1
10	800-700-600-500-400-256-128-64-32-16	1.69 M	0.9967	0.0033	0.9963	19.6	0.9842	0.0158	0.979	40.2

Table B.22
Evaluation metrics from shallow ANN models trained on increasingly large subsets of the IDS 2017 and IDS 2018 datasets.

Subset size	IDS 2017				IDS 2018			
	Acc.	Prec.	FNR	F1	Acc.	Prec.	FNR	F1
10%	0.9925	0.991	0.0075	0.9916	0.9832	0.9781	0.0168	0.9779
20%	0.9913	0.9905	0.0087	0.9907	0.983	0.978	0.017	0.9779
30%	0.9942	0.994	0.0058	0.9938	0.9834	0.9779	0.0166	0.9782
40%	0.9938	0.9936	0.0062	0.9934	0.9831	0.9757	0.0169	0.9779
50%	0.9935	0.9932	0.0065	0.993	0.983	0.9776	0.017	0.9777
60%	0.9951	0.9949	0.0049	0.9947	0.9834	0.9791	0.0166	0.9782
70%	0.9947	0.9941	0.0053	0.9942	0.9831	0.9784	0.0169	0.9779
80%	0.9949	0.9947	0.0051	0.9945	0.9835	0.9792	0.0166	0.9783
90%	0.9952	0.995	0.0048	0.9948	0.9831	0.9784	0.0169	0.9779
100%	0.9956	0.9955	0.0044	0.9954	0.9825	0.9775	0.0175	0.9771

References

Abeshu, A., Chilamkurti, N., Feb. 2018. Deep learning: the frontier for distributed attack detection in fog-to-things computing. IEEE Commun. Mag. 56 (2), 169–175.

Al-Garadi, M.A., Mohamed, A., Al-Ali, A., Du, X., Guizani, M., 2018. A Survey of Machine and Deep Learning Methods for Internet of Things (IoT) Security. arXiv:1807.11023 [cs] http://arxiv.org/abs/1807.11023.

Alom, M.Z., Bontupalli, V., Taha, T.M., Jun. 2015. Intrusion detection using deep belief networks. In: 2015 National Aerospace and Electronics Conference (NAECON), pp. 339–344.

Alrawashdeh, K., Purdy, C., Dec. 2016. Toward an online anomaly intrusion detection system based on deep learning. In: 2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA), pp. 195–200.

Amodei, D., Ananthanarayanan, S., Anubhai, R., et al., Jun. 2016. Deep speech 2: end-to-end speech recognition in English and Mandarin. In: International Conference on Machine Learning, pp. 173–182. http://proceedings.mlr.press/v48/amodei16.html.

Baldi, P., Jun. 2012. Autoencoders, unsupervised learning, and deep architectures. In: Proceedings of ICML Workshop on Unsupervised and Transfer Learning, pp. 37–49. http://proceedings.mlr.press/v27/baldi12a.html.

Bengio, Y., 2009. Learning deep architectures for AI. Foundations and Trends{\textregistered} in Machine Learning, 2, pp. 1–127 1.

Bengio, Y., Courville, A., Vincent, P., Aug. 2013. Representation learning: a review and new perspectives. IEEE Trans. Pattern Anal. Mach. Intell. 35 (8), 1798–1828.

Berman, D.S., Buczak, A.L., Chavis, J.S., Corbett, C.L., Apr. 2019. A survey of deep learning methods for cyber security. Information 10 (4), 122.

Bontemps, L., Cao, V.L., McDermott, J., Le-Khac, N.-A., Mar. 2017. Collective Anomaly Detection Based on Long Short Term Memory Recurrent Neural Network. arXiv:1703.09752 [cs] http://arxiv.org/abs/1703.09752.

Buczak, A.L., Guven, E., 2016. A survey of data mining and machine learning methods for cyber security intrusion detection. IEEE Commun. Surv. Tutor. 18 (2), 1153–1176.

Chaabouni, N., Mosbah, M., Zemmari, A., Sauvignac, C., Faruki, P., 2019. Network intrusion detection for IoT security based on learning techniques. IEEE Commun. Surv. Tutor. 21 (3), 2671–2701.

CIC, 2018. CSE-CIC-IDS2018 Dataset. https://www.unb.ca/cic/datasets/ids-2018.html.

Cisco, 2018. Cisco Security Analytics Whitepaper. Tech. rep. https://www.cisco.com/c/dam/en/us/products/collateral/security/stealthwatch/white-paper-c11-740605.pdf.

CyberEdge, 2019. 2019 Cyberthreat Defense Report. https://go.illusivenetworks.com/2019-cyberthreat-defense-report.

Demar, J., 2006. Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. 7 (Jan), 1–30. http://www.jmlr.org/papers/v7/demsar06a.html.

Diro, A., Chilamkurti, N., 2018a. Leveraging LSTM networks for attack detection in fog-to-things communications. IEEE Commun. Mag. 56 (9), 124–130.

Diro, A.A., Chilamkurti, N., 2018b. Distributed attack detection scheme using deep learning approach for Internet of Things. Future Generat. Comput. Syst. 82, 761–768.

Erhan, D., Bengio, Y., Courville, A., Manzagol, P.-A., Vincent, P., Bengio, S., 2010. Why does unsupervised pre-training help deep learning? J. Mach. Learn. Res. 11 (Feb), 625–660. http://www.jmlr.org/papers/v11/erhan10a.html.

Farah, N.H., Avishek, M., Muhammad, F., Rahman Onik, A., Rafni, M., Md, D., 2015. Application of machine learning approaches in intrusion detection system: a survey. Int. J. Adv. Res. Artif. Intell. 4 (3).

Ferrag, M.A., Maglaras, L., Moschoyiannis, S., Janicke, H., Feb. 2020. Deep learning for cyber security intrusion detection: approaches, datasets, and comparative study. J. Inform. Secur. Appl. 50, 102419.

Fiore, U., Palmieri, F., Castiglione, A., De Santis, A., Dec. 2013. Network anomaly detection with the restricted Boltzmann machine. Neurocomputing 122, 13–23.

FireEye, 2019. M-trends 2019 Cyber Security Report. FireEye, https://www.fireeye.com/current-threats/annual-threat-report/mtrends.html.

Gao, N., Gao, L., Gao, Q., Wang, H., Nov. 2014. An intrusion detection model based on deep belief networks. In: 2014 Second International Conference on Advanced Cloud and Big Data, pp. 247–252.

Glorot, X., Bengio, Y., Mar. 2010. Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, pp. 249–256. http://proceedings.mlr.press/v9/glorot10a.html.

Glorot, X., Bordes, A., Bengio, Y., Jun. 2011. Deep Sparse rectifier neural networks. In: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, pp. 315–323. http://proceedings.mlr.press/v15/glorot11a.html.

He, K., Zhang, X., Ren, S., Sun, J., 2015. Delving deep into rectifiers: surpassing human-level performance on ImageNet classification. In: Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV). ICCV 15. IEEE Computer Society, Washington, DC, USA, pp. 1026–1034.

He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778. http://openaccess.thecvf.com/content_cvpr_2016/html/He_Deep_Residual_Learning_CVPR_2016_paper.html.

He, Y., Mendis, G.J., Wei, J., Sep. 2017. Real-time detection of false data injection attacks in smart grid: a deep learning-based intelligent mechanism. IEEE Trans. Smart Grid 8 (5), 2505–2516.

Hinton, G., Deng, L., Yu, D., Dahl, G., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Kingsbury, B., Sainath, T., Nov. 2012. Deep neural networks for acoustic modeling in speech recognition. IEEE Signal Process. Mag. 29, https://www.microsoft.com/en-us/research/publication/deep-neural-networks-for-acoustic-modeling-in-speech-recognition/.

Hodo, E., Bellekens, X., Hamilton, A., Dubouilh, P., Iorkyase, E., Tachtatzis, C., Atkinson, R., May 2016. Threat analysis of IoT networks using artificial neural network intrusion detection system. In: 2016 International Symposium on Networks, Computers and Communications (ISNCC), pp. 1–6.

Hodo, E., Bellekens, X., Hamilton, A., Tachtatzis, C., Atkinson, R., Jan. 2017. Shallow and Deep Networks Intrusion Detection System: A Taxonomy and Survey. arXiv:1701.02145 [cs] http://arxiv.org/abs/1701.02145.

Ingre, B., Yadav, A., Jan. 2015. Performance analysis of NSL-KDD dataset using ANN. In: 2015 International Conference on Signal Processing and Communication Engineering Systems, pp. 92–96.

Ioffe, S., Szegedy, C., Mar. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. arXiv:1502.03167 [cs] http://arxiv.org/abs/1502.03167.

Javaid, A., Niyaz, Q., Sun, W., Alam, M., 2016. A deep learning approach for network intrusion detection system. In: Proceedings of the 9th EAI International Conference on Bio-Inspired Information and Communications Technologies (Formerly BIONETICS). BICT15. ICST. Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, New York City, United States, pp. 21–26.

Jiang, F., Fu, Y., Gupta, B.B., Lou, F., Rho, S., Meng, F., Tian, Z., 2019. Deep learning based multi-channel intelligent attack detection for data security. IEEE Trans. Sustain. Comput. 11.

Jin, Kim, Shin, Nara, Jo, S.Y., Kim, Sang Hyun, Feb. 2017. Method of intrusion detection using deep neural network. In: 2017 IEEE International Conference on Big Data and Smart Computing (BigComp), pp. 313–316.

Kang, M.-J., Kang, J.-W., Jun. 2016. Intrusion detection system using deep neural network for in-vehicle network security. PloS One 11 (6), e0155781.

KDD, 1999. KDD Cup 1999 Data. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.
Kim, J., Kim, J., Thu, H.L.T., Kim, H., Feb. 2016. Long short term memory recurrent neural network classifier for intrusion detection. In: 2016 International Conference on Platform Technology and Service (PlatCon), pp. 1–5.
Kingma, D.P., Ba, J., Jan. 2017. Adam: A Method for Stochastic Optimization. arXiv:1412.6980 [cs] http://arxiv.org/abs/1412.6980.
Klambauer, G., Unterthiner, T., Mayr, A., Hochreiter, S., 2017. Self-normalizing neural networks. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (Eds.). Advances in Neural Information Processing Systems, vol. 30. Curran Associates, Inc., pp. 971–980. http://papers.nips.cc/paper/6698-self-normalizing-neural-networks.pdf.
Kneale, C., Sadeghi, K., Jul. 2019. Semisupervised Adversarial Neural Networks for Cyber Security Transfer Learning. https://arxiv.org/abs/1907.11129v1.
Krizhevsky, A., Sutskever, I., Hinton, G.E., 2012. ImageNet classification with deep convolutional neural networks. In: Pereira, F., Burges, C.J.C., Bottou, L., Weinberger, K.Q. (Eds.). Advances in Neural Information Processing Systems, vol. 25. Curran Associates, Inc., pp. 1097–1105. http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.
Kwon, D., Kim, H., Kim, J., Suh, S.C., Kim, I., Kim, K.J., 2017. A survey of deep learning-based network anomaly detection. Cluster Comput..
Kwon, D., Natarajan, K., Suh, S.C., Kim, H., Kim, J., Jul. 2018. An empirical study on network anomaly detection using convolutional neural networks. In: 2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS), pp. 1595–1598.
Lapuschkin, S., Wldchen, S., Binder, A., Montavon, G., Samek, W., Mller, K.-R., Mar. 2019. Unmasking Clever Hans predictors and assessing what machines really learn. Nat. Commun. 10 (1), 1096.
Le, T., Kim, J., Kim, H., Feb. 2017. An effective intrusion detection classifier using long short-term memory with gradient descent optimization. In: 2017 International Conference on Platform Technology and Service (PlatCon), pp. 1–6.
LeCun, Y., Bengio, Y., Hinton, G., May 2015. Deep learning. Nature 521 (7553), 436–444.
Li, Y., Ma, R., Jiao, R., 2015. A Hybrid Malicious Code Detection Method Based on Deep Learning.
Loukas, G., Vuong, T., Heartfield, R., Sakellari, G., Yoon, Y., Gan, D., 2018. Cloud-based cyber-physical intrusion detection for vehicles using deep learning. IEEE Access 6, 3491–3508.
Ma, T., Wang, F., Cheng, J., Yu, Y., Chen, X., Oct. 2016. A hybrid spectral clustering and deep neural network ensemble algorithm for intrusion detection in sensor networks. Sensors 16 (10).
Microsoft, 2019. Azure Sentinel. https://azure.microsoft.com/en-ca/services/azure-sentinel/.
Mishra, P., Varadharajan, V., Tupakula, U., Pilli, E.S., 2019. A detailed investigation and analysis of using machine learning techniques for intrusion detection. IEEE Commun. Surv. Tutor. 21 (1), 686–728.
Montavon, G., Samek, W., Mller, K.-R., Feb. 2018. Methods for interpreting and understanding deep neural networks. Digit. Signal Process. 73, 1–15.
Mukkamala, S., Janoski, G., Sung, A., May 2002. Intrusion detection using neural networks and support vector machines. In: Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN02 (Cat. No.02CH37290), vol. 2, pp. 1702–1707 vol. 2.
Pineau, J., 2019. ReproducibilityChecklist-v1.2. Tech. rep.. https://www.cs.mcgill.ca/jpineau/ReproducibilityChecklist.pdf.
Potluri, S., Diedrich, C., Sep. 2016. Accelerated deep neural networks for enhanced Intrusion Detection System. In: 2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA), pp. 1–8.
Radford, B.J., Apolonio, L.M., Trias, A.J., Simpson, J.A., Mar. 2018. Network Traffic Anomaly Detection Using Recurrent Neural Networks. arXiv:1803.10769 [cs] http://arxiv.org/abs/1803.10769.
Radiflow, 2018. Detection of a Crypto-Mining Malware Attack at a Water Utility. https://radiflow.com/detection-of-a-crypto-mining-malware-attack-at-a-water-utility/.
Ring, M., Wunderlich, S., Scheuring, D., Landes, D., Hotho, A., Sep. 2019. A survey of network-based intrusion detection data sets. Comput. Secur. 86, 147–167.
Sharafaldin, I., Habibi Lashkari, A., Ghorbani, A.A., 2018. Toward generating a new intrusion detection dataset and intrusion traffic characterization:. In: Proceedings of the 4th International Conference on Information Systems Security and Privacy. SCITEPRESS - Science and Technology Publications, Funchal, Madeira, Portugal, pp. 108–116.
Shimeall, T.J., Spring, J.M., Jan. 2014. Chapter 12 - recognition strategies: intrusion detection and prevention. In: Shimeall, T.J., Spring, J.M. (Eds.), Introduction to Information Security. Syngress, Boston, pp. 253–274.
Shone, N., Ngoc, T.N., Phai, V.D., Shi, Q., Feb. 2018. A deep learning approach to network intrusion detection. IEEE Trans. Emerg. Top. Comput. Intell. 2 (1), 41–50.
Snort, Snort, https://github.com/snort3/snort3.
Sommer, R., Paxson, V., 2010. Outside the closed world: on using machine learning for network intrusion detection. In: 2010 IEEE Symposium on Security and Privacy. IEEE, Oakland, CA, USA, pp. 305–316.
Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R., 2014. Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. 15, 1929–1958. http://jmlr.org/papers/v15/srivastava14a.html.
Staudemeyer, R.C., Jul. 2015. Applying long short-term memory recurrent neural networks to intrusion detection. S. Afr. Comput. J. 56 (1).
Sultana, N., Chilamkurti, N., Peng, W., Alhadad, R., Mar. 2019. Survey on SDN based network intrusion detection system using machine learning approaches. Peer-to-Peer Network. Appl. 12 (2), 493–501.
Sutskever, D., Cho, K., Bengio, Y., Sep. 2014. Neural Machine Translation by Jointly Learning to Align and Translate. arXiv:1409.0473 [cs, stat] http://arxiv.org/abs/1409.0473.
Taghiyarrenani, Z., Fanian, A., Mahdavi, E., Mirzaei, A., Farsi, H., Oct. 2018. Transfer learning based intrusion detection. In: 2018 8th International Conference on Computer and Knowledge Engineering (ICCKE), pp. 92–97.
Tang, T.A., Mhamdi, L., McLernon, D., Zaidi, S.A.R., Ghogho, M., Oct. 2016. Deep learning approach for network intrusion detection in software defined networking. In: 2016 International Conference on Wireless Networks and Mobile Communications (WINCOM), pp. 258–263.
Tavallaee, M., Bagheri, E., Lu, W., Ghorbani, A.A., Jul. 2009. A detailed analysis of the KDD CUP 99 data set. In: 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, pp. 1–6.
Taylor, A., Leblanc, S., Japkowicz, N., Oct. 2016. Anomaly detection in automobile control network data with long short-term memory networks. In: 2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA), pp. 130–139.
Tieleman, T., 2008. Training restricted Boltzmann machines using approximations to the likelihood gradient. In: Proceedings of the 25th International Conference on Machine Learning. ICML 08. ACM, Helsinki, Finland, pp. 1064–1071.
Tsai, C.-F., Hsu, Y.-F., Lin, C.-Y., Lin, W.-Y., 2009. Intrusion detection by machine learning: a review. Expert Syst. Appl. 36 (10), 11994–12000.
Vinayakumar, R., Alazab, M., Soman, K.P., Poornachandran, P., Al-Nemrat, A., Venkatraman, S., 2019. Deep learning approach for intelligent intrusion detection system. IEEE Access 7, 41525–41550.
Wang, Z., 2018. Deep learning-based intrusion detection with adversaries. IEEE Access 6, 38367–38384.
Wang, Wei, Zhu, Ming, Zeng, Xuewen, Ye, Xiaozhou, Sheng, Yiqiang, Jan. 2017. Malware traffic classification using convolutional neural network for representation learning. In: 2017 International Conference on Information Networking (ICOIN), pp. 712–717.
Wu, Y., Schuster, M., Chen, Z., et al., Sep. 2016. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. arXiv:1609.08144 [cs] http://arxiv.org/abs/1609.08144.
Xin, Y., Kong, L., Liu, Z., Chen, Y., Li, Y., Zhu, H., Gao, M., Hou, H., Wang, C., 2018. Machine learning and deep learning methods for cybersecurity. IEEE Access 6, 35365–35381.
Xu, C., Shen, J., Du, X., Zhang, F., 2018. An intrusion detection system using a deep neural network with gated recurrent units. IEEE Access 6, 48697–48707.
Yan, J., Jin, D., Lee, C.W., Liu, P., Jul. 2018. A comparative study of off-line deep learning based network intrusion detection. In: 2018 Tenth International Conference on Ubiquitous and Future Networks (ICUFN), pp. 299–304.
Yin, C., Zhu, Y., Fei, J., He, X., 2017. A deep learning approach for intrusion detection using recurrent neural networks. IEEE Access 5, 21954–21961.
Zeng, Y., Gu, H., Wei, W., Guo, Y., 2019. Deep-full-range: a deep learning based network encrypted traffic classification and intrusion detection framework. IEEE Access 7, 45182–45190.
Zhao, J., Shetty, S., Pan, J.W., Kamhoua, C., Kwiat, K., Feb. 2019. Transfer learning for detecting unknown network attacks. EURASIP J. Inf. Secur. 2019 (1), 1.

Sunanda Gamage Sunanda Gamage is currently pursuing a Ph.D. in electrical and computer engineering at the University of Western Ontario. He received his B.Sc. (Eng.) in Electronics and Telecommunication with first class honours from the University of Moratuwa, Sri Lanka in 2015. He worked as a software engineer on real-time distributed systems at London Stock Exchange Group Technology, Sri Lanka for 3 years. His research interests are in novel machine learning models and network intrusion detection.

Jagath Samarabandu Jagath Samarabandu is a full professor in electrical and computer engineering at the University of Western Ontario since 2000. He received his B.Sc. (Eng.) in Electronics and Telecommunication with first class honours from the University of Moratuwa, Sri Lanka in 1982. He was awarded the Fulbright scholarship in 1987 for postgraduate study at State University of New York at Buffalo. He received M.S and Ph.D. degrees in Electrical Engineering from SUNY-Buffalo in 1990 and 1994 respectively. His research interests are in machine learning applications and computer network security.