EURASIP Journal on Information Security

RESEARCH Open Access

Multi-layer intrusion detection system with ExtraTrees feature selection, extreme learning machine ensemble, and softmax aggregation

Jivitesh Sharma[1*], Charul Giri[1], Ole-Christoffer Granmo[1] and Morten Goodwin[1]

Abstract

Recent advances in intrusion detection systems based on machine learning have indeed outperformed other techniques, but struggle with detecting multiple classes of attacks with high accuracy. We propose a method that works in three stages. First, the ExtraTrees classifier is used to select relevant features for each type of attack individually for each (ELM). Then, an ensemble of ELMs is used to detect each type of attack separately. Finally, the results of all ELMs are combined using a softmax layer to refine the results and increase the accuracy further. The intuition behind our system is that multi-class classification is quite difficult compared to binary classification. So, we divide the multi-class problem into multiple binary classifications. We test our method on the UNSW and KDDcup99 datasets. The results clearly show that our proposed method is able to outperform all the other methods, with a high margin. Our system is able to achieve 98.24% and 99.76% accuracy for multi-class classification on the UNSW and KDDcup99 datasets, respectively. Additionally, we use the weighted extreme learning machine to alleviate the problem of imbalance in classification of attacks, which further boosts performance. Lastly, we implement the ensemble of ELMs in parallel using GPUs to perform intrusion detection in real time.

Keywords: Intrusion detection system, Machine learning, Artificial intelligence, Extreme learning machine, Ensemble methods, Feature selection, ExtraTrees, Softmax aggregation

1 Introduction

With the advancement of Internet technologies, the day-to-day life has become much easier and simpler. The emerging need of Internet has not only led to the rapid growth of web applications, data transfer devices, protocols, computer networks, and cloud computing but has also given rise to complex security threat environments, whether it be data security, identity theft, or social and engineering attacks. The complex nature and exponential growth of cyberattacks and their ability to deal with the current network security system highlight the necessity of more accurate and efficient security systems.

Network intrusion detection systems (abbreviated as IDS) have been developed over time to detect any unauthorized or suspicious action which can lead to data theft, breach in discretion, availability, and integrity of information resources. While IDS are good at monitoring traffic for malicious activities, they are prone to false positives, can lead to chasing ghosts because the attackers can fake the IP address, and encrypted packets which go unprocessed by IDS and could release malicious content when activated later. It results in high involvement and dependency on human analysts. Hence, the efficiency and accuracy of IDS have been a major concern in both research and industry.

The complexity in modern day attacks, and their highly intelligent and adaptive nature, has raised questions on the present adopted traditional network security measures. Signature-based IDS have attack signature database of known attacks. They can only detect attacks which matches with the stored signature and are unable to detect any other novel attack. Anomaly-based IDS create a model of the normal traffic by monitoring it and classify any

*Correspondence: jivitesh.sharma@uia.no
[1]University of Agder, Grimstad, Norway

other activity as abnormal or anomaly. Anomaly-based IDS can more effectively detect an unknown type of attack. One issue with anomaly detection is how to classify normal traffic from anomalies in efficient way. Several researches have been made, and various approaches have been proposed for the development of accurate and real-time IDS which can outperform the current intrusion detection system.

Machine learning has caught the attention of a lot of researchers to provide solutions for especially wide ranging big data problems. It can handle multi-dimensional data in dynamic environments giving real-time predictions. Presently, the advances in machine learning has extended its application for implementation of effective IDS. Learning-based approaches like neural networks have been outperforming traditional approaches in various applications. Machine learning-based IDS can keep up with varying types of attacks due to their learning and adaptive nature.

Intrusion detection can fall under various application-based categories using neural networks and statistical methods. In [1], probability features and fast parallel processing with Hadoop, consisting of a cluster of 19 nodes, were used to detect the authenticity of images. Also, in [2], false data injection attacks were detected in power system state estimation using the non-linear autoregressive exogenous (NARX) configuration of the neural network.

In this paper, we propose a novel approach based on neural networks for the problem of general purpose network intrusion detection. All previous approaches for intrusion detection either distinguish between normal traffic and attacks or can only detect one type of attack at a time. We propose to use an ensemble of ELMs [3] for detecting all types of attacks simultaneously. Each ELM is trained for a specific type of attack, and each ELM is fed a different feature set consisting of features selected by an ExtraTree classifier [4] for that specific attack. Training these ELMs on each and every type of attack takes less than 18 s. Our system is tested on the UNSW [5] and NSL-KDD [6] datasets and is able to outperform all previous machine learning-based intrusion detection systems.

To the best of our knowledge, the following novel techniques have never been applied to the intrusion detection problem:

1. This paper proposes to use attack-based feature selection. This decreases the number of features significantly.
2. Also, in this paper, an ensemble of extreme learning machine (ELM) is used for multi-class classification.
3. Each ELM distinguishes between one type of attack and all the rest of the categories. This is similar to one vs all SVM, but this is the first time it has been applied to a neural network-based ensemble for intrusion detection.
4. For such a unique kind of ensemble technique, voting-based aggregation cannot be applied. So we use softmax aggregation.
5. We report new state-of-the-art accuracy for multi-class intrusion detection on two of the benchmark datasets: UNSW and NSL-KDD.
6. Even better performance is achieved by replacing ELM with WELM at the cost of negligible computational overhead.
7. We implement the ensemble of ELMs in parallel using GPUs and map-reduce type of implementation to perform real-time intrusion detection.

The rest of the paper is laid out in the following manner: Section 2 enlists and briefly explains some key papers on intrusion detection systems that have used machine learning, Section 3 and its subsections give a detailed explanation of our proposed system, Section 4 shows our findings and results of experiments on the two datasets, and finally, the paper is concluded in Section 5.

2 Related work

There have been some innovative techniques proposed in the past and key research ideas implemented for intrusion detection using machine learning models that have transcended previous networking-based techniques and opened new avenues for future researchers to build their work on. Some of these ideas have been briefly described here.

In [7], the main contribution is to propose a traffic monitoring functionality for network intrusion detection and process monitoring functionality to detect modern malware attacks. The main novelty lies in the improvement over existing results benchmarked in [5]. In this paper, random forest (abbreviated as RF) is used to detect network attacks at cloud networking server (CNS) and virtual machine monitor (VMM) of cloud compute server (CCoS). Then, logistic regression (abbreviated as LR) is used as meta-classifier to reduce its overfitting problem. The dataset used is UNSW-NB and CAIDA for intrusion detection. The accuracy on UNSW-NB dataset is 94.54%, and false-positive rate is 2.81% and 98.90% on CAIDA dataset for DoS attack detection only. In contrast, our proposed method is able to deal with 10 types of attacks including DoS attacks. Also, the random forest requires hundreds of decision trees and is not as powerful as neural networks. That is why we use the ExtraTrees classifier (an updated version of random forest) for feature selection purposes only and leave the classification task to ELMs.

The paper [8] has proposed a new fitness function for genetic algorithm which uses three parameters: true-positive rate, false-positive rate, and number of selected

features to evaluate each subset of features. This is used for feature selection. Then, combination of genetic algorithm and SVM is used to detect intrusion. The method has been applied to classify each type of attack separately on KDD CUP 99 and UNSW-NB15 datasets with mostly achieving accuracy of above 90%. However, our method is able to classify each type of attack separately, as well as together in a multi-class fashion with a higher accuracy. Also, the ExtraTrees classifier used for feature selection in our model is able to prune more features than the method proposed in [8]. Performance numbers of these systems are shown in Table 1.

Developing a single neural network to discover DoS traffic in various network protocols is proposed by [9]. Feature selection is done by correlation-based feature selection. The accuracy of the network on UNSW-NB dataset is 97.1% with a false-positive rate of 0.06% and 99.2% on NSL-KDD with false-positive rate of 0.02%. Again, our proposed method detects many other types of attacks other than DoS. Also, correlation-based feature selection works only to reduce redundancy and is not data efficient as correlation between large feature matrices needs to be calculated. On the other hand, the decision tree-based ExtraTrees classifier, that we use for feature selection, is faster and much more data efficient and more importantly calculates feature importance score rather than evaluating features for redundancy. Our proposed model deploys feature selection in a novel way by selecting important features for detecting each type of attack separately.

This paper [10] proposes an intrusion detection model which is based on two stages and on a reduced error pruning tree algorithm for classification and identification of intrusion. In the first stage, the traffic is classified on the basis of its protocol. Then, a binary classification is done to define the traffic as normal or as an attack. In the second stage, a pre-trained multi-class classifier is launched which classifies the type of attack whenever an attack is identified. Even though our method has three stages, they are still less complex than the two stage method in [10]. Firstly, reduced error pruning tree adds more overhead to a decision tree by repeatedly building the tree and pruning it. The model is divided into two stages: attack detection and type of attack classification, while we do it in a single stage. Also, their method achieves 81.28% and 83.59% accuracy on UNSW-NB15 and NSL-KDD, respectively, which is much lower compared to the accuracies achieved by our proposed system.

The paper [11] proposes a hybrid model by combining SVM and simulated annealing for intrusion detection. Random unique combinations of three features were created by simulated annealing at a time, and SVM was applied on those feature combination. This is repeated until low false-positive and false-negative rates and highest detection accuracy are achieved by the model. Accuracy achieved by the model is 98.76% for binary classification. The random selection of three features and repeating the process until highest performance is achieved takes a lot of computational power and time.

The authors in [12] performed random forest binary classification on various stages to classify eight kinds of attacks. They compared logistic regression and random forest for anomaly detection and showed that RF outperformed LR. They proposed that by reducing the number of features using best first feature selection technique on the criterion of minimizing the testing error resulted in reduced model complexity and enhanced the accuracy. The results showed that overall accuracy achieved was 99% for anomaly detection (binary classification) but categorization accuracy is 93.35% which is comparatively lower than our method. Also, best first feature selection is a naive feature selection technique and takes a lot of time to select the relevant features. On the other hand, we employ the ExtraTrees classifier as a feature selection algorithm which is more sophisticated and applied in a novel individual attack detection way.

In [13], a standard neural network (abbreviated as NN) has been used to classify attacks and genetic algorithm for feature selection. The similar features are classified into feature sets, and accuracy of each set is measured by multi-modal neural network (abbreviated as MNN). The genetic algorithm is applied to each feature set in ascending order of accuracy, eliminating the irrelevant features from overall feature set and evaluating remaining features with MNN. The accuracy achieved by the ANN on UNSW-NB15 and NSL-KDD is 91.98% and 95.46%, respectively, for binary classification between normal and attack traffic. We use ExtraTrees classifier for feature selection which is much faster and less complex than genetic algorithms (slow execution is the main problem of GA). Also, we perform ensemble ELM and softmax aggregation learning for the multi-class problem to detect all types of attacks.

In [14], a multi-scale Hebbian NN for threat detection is introduced. The learning algorithm follows Hebbian rule in which weights are updated as a function of nearby neuronal activity. The paper uses four features for detection and has shown improved true negative rate as 95% and true positive rate as 73%. Improvement is shown in comparison between multi-scale Hebbian-based NN and gradient descent-based NN. The mean accuracy of the proposed method is 93.56% for detecting attacks (binary classification). The Hebbian rule makes the learning much slower for neural networks, and introducing multi-scale approach makes it even slower. On the other hand, we deploy parallel ELMs which are many times faster than backpropagation.

The paper [15] employs a multi-agent-based cognitive approach to detect network intrusion and feature detection

Table 1 Previous state-of-the-art ML based intrusion detection systems and the proposed method

Models	UNSW (in %)	KDD (in %)	Binary classification	Detecting single type of attack	Multi-class classification
NvCloudIDS [7]	94.54	-	✓	-	-
ADDM [9]	97.1	99.2	✓	-	-
GF+SVM [8]	Normal 97.45				
	Generic 91.51	Normal 99.05			
	Exploits 79.19	DoS 99.95			
	DoS 91.24	Probe 99.06	-	✓	-
	Fuzzers 96.39	R2L 98.25			
	Reconnaissance 91.51	U2R 100			
	Shellcode 99.45				
RepTree [10]	88.95(Binary)	89.85(Binary)	✓	-	✓
	81.28(Multi-class)	83.59(Multi-class)			
Simulated annealing+SVM [11]	98.76	-	✓	-	-
Step-wise RF [12]	Normal 99.50				
	Exploits 99.50				
	DoS 20.00				
	Analysis 2.00	-		✓	-
	Backdoor 5.00				
	Reconnaissance 86.00				
	Shellcode 80.00				
	Worm 70.00				
ANN+GF [13]	91.98	95.46	✓	-	-
Multi-scale Hebbian [14]	93.56		✓	-	-
Unsupervised FE+classification [15]	89.00	-	✓	-	-
Semi-supervised ML [16]	93.74	98.23	✓	-	-
MLP [17]	93.29	-	✓	-	-
ICVAE-DNN [18]	89.08	85.97	-	-	✓
BMM+outlier detection [19]	Normal 93.40				
	Generic 80.50				
	Exploits 79.40				
	DoS 89.60				
	Analysis 83.40				
	Backdoor 63.80	-	-	✓	✓
	Reconnaissance 55.60				
	Shellcode 48.70				
	Worm 47.80				
	Overall 92.70				
GRU-RNN [20]	-	89.00	-	-	✓
Proposed method: ExtraTrees+ELM ensemble+softmax	*98.24*	*99.76*	-	✓	✓
Proposed method: ExtraTrees+WELM ensemble+softmax	*98.69*	*99.83*	-	✓	✓

through unsupervised learning. The authors modified the UNSW-NB15 dataset for the unsupervised problem. The paper proposed to divide datasets into time steps and then finding features from their statistical analysis resulting in a reduced dataset, hence reducing computation time for learning and then deploying agents with each host to compute and analyze the traffic flow. This technique yields accuracy of 89%. The accuracy with basic unsupervised *k*-means is 29% on AWIDR dataset, and with this approach, it is increased by 60%. Even though computation time for learning by the multi-agent system is reduced, the total time is increase due to the statistical analysis of the dataset since the dataset is quite large.

In this paper [16], the authors have proposed a semi-supervised learning approach to detect DDoS attack. The unsupervised part reduces the irrelevant and noisy data through entropy estimation, co-clustering, and information gain ratio, resulting in reduced false-positive rate. The supervised part uses ExtraTrees ensemble classifiers to classify the traffic. The best accuracy achieved on NSL-KDD, UNB ISCX 12, and UNSW-NB15 datasets are 98.23%, 99.88%, and 93.74%, respectively, for binary classification. Again, the unsupervised overhead of the system is too complex and requires lots of data and time. Our method is much less complex and can detect intrusions in real time.

A multi-layer perceptron-based feedforward neural network for detecting intrusion with sigmoid activation function and backpropagation learning algorithm has been proposed in [17]. The approach has achieved on the 93.29% UNSW-NB15 dataset. The approach is compared to J48 decision tree, and it turned out that J48 has comparatively lower accuracy of 88.53% for binary classification. The approach is too simple, and the backpropagation algorithm takes a lot of training time. Also, the accuracy reported on the UNSW-NB15 dataset for binary classification is on the lower side.

One of the most recently proposed methods to improve multi-class classification performance of intrusion detection [18] proposed to use a combination of improved conditional variational autoencoder with a deep neural network. Their approach is abbreviated as ICVAE-DNN. The autoencoder is used as a generative model to provide more attack samples according to the specified categories in the datasets, to the DNN classifier. This innovative approach was able to achieve 89.08% and 85.97% accuracy on the UNSW and NSL-KDD datasets, respectively. However, the model complexity is quite high since it uses two deep neural networks trained by backpropagation.

Another recent intrusion detection method that used machine learning and statistical techniques such as beta mixture models (BMM) and outlier detection was proposed in [19]. The method was tested on the UNSW dataset achieving an overall score of 92.7%. Individual attack detection accuracies were also reported.

In [20], a recurrent neural network (RNN) with gated recurrent unit (GRU) was proposed for intrusion detection. The model was specifically built for SDN networks (software-defined networking). The most striking feature of the model is that it uses only six features to achieve 89% classification accuracy on the NSL-KDD dataset.

In our approach, we use the extreme learning machine (ELM) [3, 21] for intrusion detection. The ELM algorithm has been used in a diverse set of applications including water quality forecasting [22], optimization of industrial chemical productions [23], big data processing [24], speech enhancement [25], heart disease diagnosis [26], medical image segmentation [27], and fault detection [28, 29].

The ELM has also been used before in IDS [30–33]. In [30], ELM is tested on the KDcup99 dataset [6] in a big data environment using a MapReduce-based variant (MR-ELM). This is completely different from our work, because [30] use MR-ELM specifically for big data environments and achieve maximum accuracy around 97% on the KDDcup99 dataset [6] whereas our proposed model achieves 99.6% accuracy on the same dataset with some modifications and fine-tuning.

Another paper that used ELM for intrusion detection is [31]. In this paper, a simple implementation of ELM and Kernel ELM is used to detect four types of attacks on a relatively simpler dataset, DARPA 1998. The Kernel ELM model is compared with SVM for intrusion detection and achieves nearly 98% accuracy for multiple attack detection.

A variant of ELM called the online sequential ELM (OS-ELM), which is used for online learning, is employed in [32] for intrusion detection. It is tested on the KDDcup99 dataset. Although it is not able to achieve very high accuracy, 90%, it is a more realistic approach towards intrusion detection, because learning online, especially in the field of computer networks and communication, is extremely important since the byte patterns can change over time.

A sample selected, lightweight, ELM intrusion detection system for fog computing and mobile edge computing is proposed in [33]. The main contributions of the paper lie in the fast training and accurate detection (99%) of attacks on the KDDcup99 dataset. Our model is entirely different from the abovementioned techniques, as explained in the next section.

The difference between our ELM-based IDS and the previous ELM-based IDS is that our approach is a general purpose intrusion detection system that consists of an ensemble of ELMs, where each ELM is dedicated to detect a single type of attack. Also, we introduce a novel type of aggregating ensemble results by using a softmax layer which proves to improve accuracy of the system.

Furthermore, the ExtraTrees classifier is used to select relevant features for each type of attack separately to improve data efficiency and speed of the system. Overall, this general purpose IDS can be implemented in a parallel processing or distributed manner due to its multi-threaded structure as shown in Fig. 1, where each thread is independent of the other.

All the abovementioned papers and more have greatly improved intrusion detection systems in the past. However, almost all of them have their own shortcomings which we try to overcome in this paper. Some of the drawbacks that we try to solve are as follows: computation and training time, efficiency, accuracy, feature selection, classification between all types of attacks (not just binary or detecting a specific type of attack only), etc. Most of the previous state-of-the-art methods deal with binary classification, i.e., distinguishing between normal traffic and attacks.

Table 1 displays the results of the previous state-of-the-art methods on the NSL-KDD and UNSW benchmark datasets. The table also shows the type of classification performed. We propose a system that can achieve the highest accuracy for the multi-class problem of detecting all types of attacks on which the system is trained on. We explain our IDS and the intuition behind it in the next section.

3 The multi-layer intrusion detection system

The main contributions of this paper lie in the following: using ELM [3, 21] for classification of attacks, even though ELM has been used for intrusion detection systems before [30–33], but not quite in this manner and not on the UNSW dataset which is the most realistic and difficult dataset for intrusion detection; using the ExtraTrees classifier [4] to calculate feature importance and select relevant features for detecting each specific type of attack; and using an ensemble of ELMs (one for each type of attack) and combining their results using a softmax layer to fabricate an interpretable probabilistic output of very high accuracy. The flowchart of our deep multi-layered model is shown in Fig. 1.

We breakdown the problem of multi-class classification into a set of binary classifications. This is done in order to decrease the load on the classifiers in the ensemble. Since,

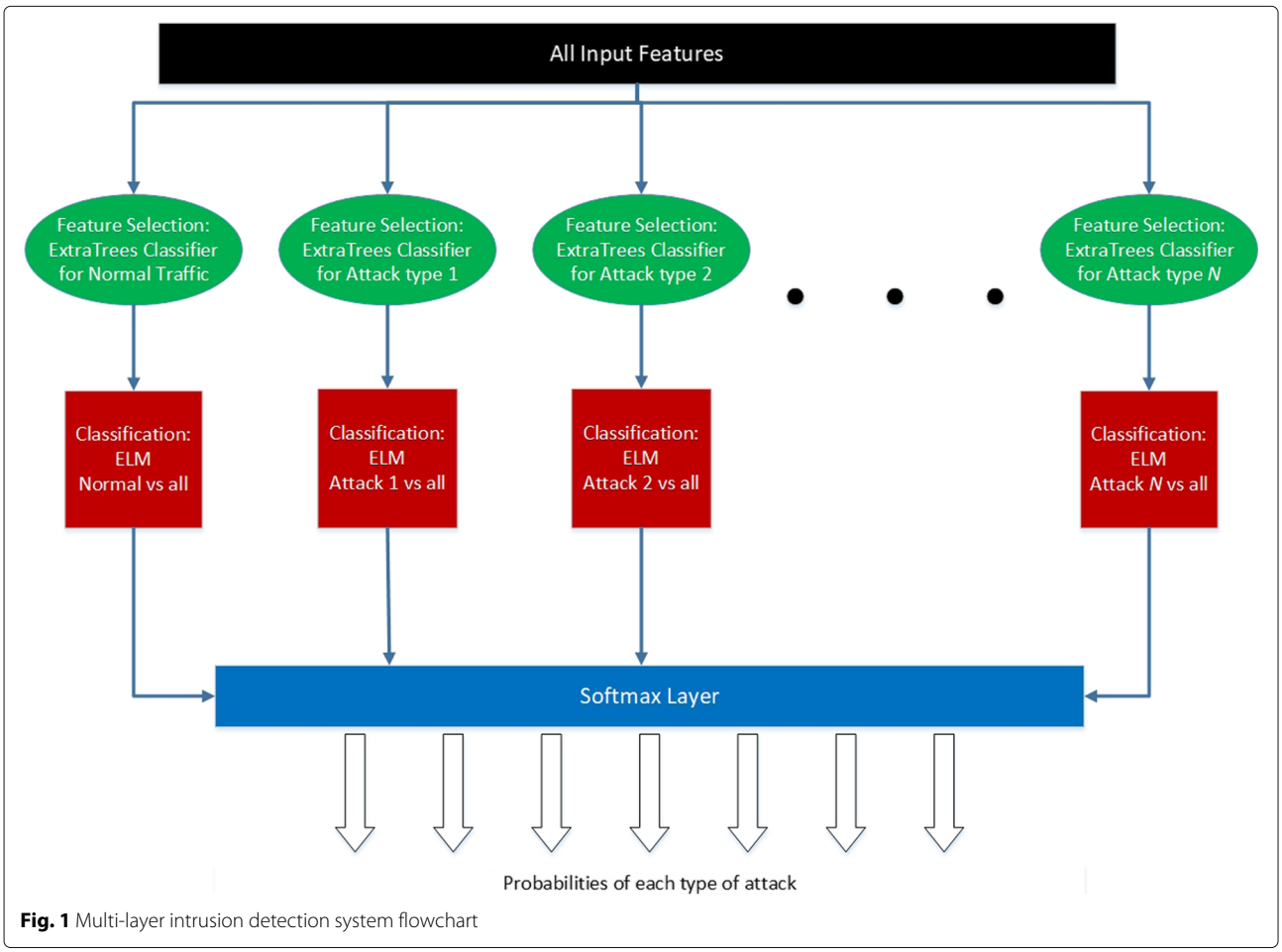

Fig. 1 Multi-layer intrusion detection system flowchart

multi-class classification is more complex than binary classification. This is because binary classification consists of two decision variables, i.e., the two classes, whereas the multi-class problem can consist of n decision variables representing the n classes. So, it is easier to learn a function that can map the set of input features to two decision variables rather than n decision variables. Also, the complexity of the function would be less for binary classification compared to multi-class classification, as the complexity of a function is directly proportional to the number of decision variables.

Each thread shown in Fig. 1 runs in parallel using GPUs and map-reduce type of implementation which enables real-time intrusion detection. New attacks can be detected by the system, since one ELM in the ensemble distinguishes normal network traffic from potential attacks. However, the type of the new attack cannot be determined if it does not fall under any of the attack categories on which the system is trained.

3.1 Layer 1: Feature selection with ExtraTrees classifier

The extremely randomized trees, abbreviated as ExtraTrees [4] are a variant of the random forests with more randomization at each step for picking an optimal cut/split or decision boundary. Unlike random forests where features are split based on a score (like entropy) and instances of the training set are bootstrapped, the split criteria of the ExtraTrees are random and the entire training set is considered. The resulting trees have more leaf nodes and are more computationally efficient. It also alleviates the problem of high variance in random forests due to its randomization and hence provides a better bias-variance trade-off.

Also, one of the advantages of using tree-based classifiers is their ability to perform feature selection. The advantage of using tree-based classifiers as a feature selection mechanism is that they require much less memory (as tree structures are more memory efficient), they are faster, and they give the most important features at the beginning itself starting from the root node and the first split. At each split, the most important feature is selected at that stage. As the tree grows and reaches the leaf nodes that give the end result, the path from the root node to the leaf node gives the most important features. An additional characteristic of tree-based methods is that features are given a score during each split which enables them to perform feature ranking. This characteristic is used in our model for feature selection. Features are ranked according to split score by the ExtraTrees classifier. The split score for sample S, split s, and class c is given by [4]:

$$\text{Score}_c(s,S) = \frac{2I_c^s(S)}{H_s(S) + H_c(S)} \tag{1}$$

where, $H_c(S)$ is the (log) entropy of the class c in sample S, $H_s(S)$ is the split entropy, and $I_s^c(S)$ is the mutual information of the split outcome and the class c. We select all the features above a threshold score. This is done for distinguishing each attack versus the rest. So, we get a different optimal feature subset for detecting each type of attack. Feature selection reduces redundancy, and emphasis is given to important features which leads to higher accuracy and faster training.

Most of the previous research applies feature selection for detecting all attacks. We use the ExtraTrees classifier for feature selection to detect each type of attack separately (as shown in Fig. 1) because a particular feature could be important for detecting a specific type of attack and it could be considered redundant for another type of attack. Each feature used for intrusion detection receives a score. We use a threshold score to discard irrelevant or redundant features that do not contribute enough to the benefit the performance of the intrusion detection system. This approach of individual class feature selection works better as shown in the Section 4.

The main motivation behind using a feature selection technique is to reduce the dimensionality of the problem in order to improve execution time, memory usage, and data efficiency, especially when redundant features are removed which helps to deal with overfitting and improve performance. Feature selection with decision tree-based methods is much simpler and faster compared to other techniques such as Fisher's score and F-score. The major disadvantage of using Fisher's score and F-score is that they calculate feature scores independently of other features, i.e., they do not include mutual information. On the other hand, ExtraTrees classifier uses all features together to categorize data. Some feature combinations might be better than high scoring independent features, which is why we employ ExtraTrees classifier as the feature selector.

3.2 Layer 2: Extreme learning machine ensemble

The extreme learning machine is a supervised learning algorithm originally for a single hidden layer feedforward neural network [3, 21]. But after extensive research in the past few years, it has been modified and updated to work for deep neural networks as well, details can be found here [34–37]. We use the original form of the ELM, to keep things simple and fast.

The inputs to the ELM, in this case, are the features selected by the ExtraTrees classifier [4]. Let it be represented as x_i, t_i , where x_i is the input feature instance and t_i is the corresponding label. The input features are fed to the hidden layer neurons by randomly weighted connections w. The sum of the product of the inputs and their corresponding weights act as inputs to the hidden layer activation function. The hidden layer activation

function is a non-linear non-constant bounded continuous infinitely differentiable function that maps the input data to the feature space. There is a catalog of activation functions from which we can choose according to the problem at hand. We ran experiments for all activation functions, and the best performance was achieved with the smooth approximation of the ReLU function [38], which is called the SoftPlus function [39]:

ReLU:

$$f(x) = \max(0, x) \tag{2}$$

SoftPlus:

$$f(x) = \log(1 + e^x) \tag{3}$$

The hidden layer and the output layer are connected by weights β, which are to be analytically determined. The mapping from the feature space to the output space is linear. Now, with the inputs, hidden neurons, their activation functions, the weights connecting the inputs to the hidden layer, and the output weights produce the final output function:

$$\sum_{i=1}^{L} \beta_i g(w_i.x_j + b_i) = o_j \tag{4}$$

The output in matrix form is:

$$H\beta = T \tag{5}$$

The error function used in extreme learning machine is the mean squared error function, written as:

$$E = \frac{1}{2} \sum_{j=1}^{N} \left(\sum_{i=1}^{L} \beta_i g(w_i.x_j + b_i) - t_j \right)^2 \tag{6}$$

The MSE with L_2 regularization and C as regularization parameter is:

$$E = \frac{1}{2} \sum_{j=1}^{N} \left(\sum_{i=1}^{L} \beta_i g(w_i.x_j + b_i) - t_j \right)^2 + C\frac{1}{2}||\beta||^2 \tag{7}$$

To minimize the error, we need to get the least-squares solution of the above linear system:

$$\|H\beta^* - T\| = \min_\beta \|H\beta - T\| \tag{8}$$

The minimum norm least-squares solution to the above linear system is given by:

$$\hat{\beta} = H^\dagger T \tag{9}$$

where, $H^\dagger$ is the Moore-Penrose pseudo inverse of H, which is given by [40, 41]:

$$H^\dagger = \left(\frac{I}{C} + H^T H \right)^{-1} H^T \tag{10}$$

However, the product of $H^T H$ may not always be a non-singular matrix or it may tend to be singular under certain conditions, and thus, this method of computing the pseudo inverse may not work for all cases. The singular value decomposition (SVD) can be used to calculate the Moore–Penrose pseudo inverse of H in all cases.

Properties of the above solution are as follows:

1. *Minimum training error:* The following equation provides the least-squares solution, which means the solution for $\|H\beta - T\|$, i.e., the error is minimum: $\|H\beta^* - T\| = \min_\beta \|H\beta - T\|$.
2. *Smallest norm of weights:* The minimum norm of least-squares solution is given by the Moore-Penrose pseudo inverse of the hidden layer output matrix, H: $\hat{\beta} = H^\dagger T$.
3. *Unique solution:* The minimum norm least-squares solution of $H\beta = T$ is unique, which is $\hat{\beta} = H^\dagger T$.

Detailed mathematical proofs of these properties and the ELM algorithm can be found in [3]. We use an ensemble of $N+1$ ELMs, where N is the number of types of attacks. One additional ELM, apart from N ELMs, is for detecting normal traffic. Each ELM is trained with an X vs all strategy, where X is the type of attack/normal traffic. Each ELM outputs a "1" when it detects a type of attack for which it is trained, or "0" otherwise. This approach breaks down the multi-class problem to a two-class problem with several ELM classifiers, each having to detect only one type of attack instead of several.

Even though this ensembling approach requires many ELMs, it gives a much better performance in terms of accuracy and training time and is much less computationally complex compared to single deep and wide neural networks that have a much more demanding multi-class problem at hand. This is because as the number of decision variables increases (number of classes), the network size has to be increased as well. Additionally, deep neural networks require backpropagation for training which is more computationally complex than ELM. Since each ELM in the ensemble has to detect one type of attack, they have a perspective of the data unique to that particular type of attack which makes them more efficient, accurate, and faster. This unique perspective of data with selected features is provided by the ExtraTrees classifier.

Also, convergence is guaranteed by the solution to the Moore-Penrose pseudo inverse of H, as long as sufficient number of hidden neurons are provided. We use 512 neurons for guaranteed convergence.

3.3 Layer 3: The softmax layer

The output of an ensemble of classifiers can be combined in several ways like averaging, voting, and max operation. But that is when all the classifiers have the same goal and the same perspective of the problem. Such techniques cannot work when the global view of the problem is multi-class and the local view of the classifiers is binary class.

The ELMs in the ensemble return a single output which is either "0" or "1." To amalgamate these outputs to get the final actual output becomes a challenge because the abovementioned techniques for combining ensemble results do not work here. If only one of the ELMs output is "1," then there is no problem. But let us assume that we get a difficult input stream to classify. For instance, let us consider that two ELMs output a "1," in this case which one should we consider? We cannot apply averaging or voting or max operation here for obvious reasons. To alleviate this ambiguity, we use a softmax layer at the end to integrate the outputs of the ELM ensemble and produce a probability vector which displays the probabilities of each type of attack.

The softmax layer employs the softmax function [42] which is a generalized form of the logistic function:

$$f(y)_j = \frac{e^{y_j}}{\sum_{k=1}^{N} e^{y_k}} \tag{11}$$

In order to further increase performance and accuracy of the system, the softmax layer is fine-tuned using the Adam optimizer [43]. The true classes encoded as one-hot vectors are fed as labels, and the input is the output of the ELM ensemble. This behaves as a single layer softmax classifier. The categorical cross-entropy loss works best with softmax layer which is used here as well [44]:

$$H(t, y) = -\sum_{i} t_i \log y_i \tag{12}$$

The fine-tuning is run for 10 epochs only which is enough since a large portion of the classification task is done in the ELM ensemble stage. The softmax layer acts as a module that dispenses ambiguity and makes the output interpretable. Also, it adds an additional layer of abstraction to the model. The output of the final stage is a refined probability vector that displays the probabilities of each type of attack and normal traffic associated with each input instance stream.

4 Experimental setup and results

We test our proposed intrusion detection model on the well-known benchmark datasets: UNSW-NB15 dataset [5] and KDDcup99 dataset [6]. A comparison of our methods with previous state-of-the-art machine learning techniques for intrusion detection can be seen in Table 1. The system is implemented to perform in real time. Hence, we implement the ensemble of ELMs in parallel. Each ELM is provided with a set of all features. The training and testing of the ELMs are performed in parallel using a map-reduce type of implementation [45]. Finally, the results are aggregated by a softmax layer.

Table 1 shows the results and type of classification for the recently proposed state-of-the-art machine learning-based intrusion detection systems on the UNSW and NSL-KDD datasets. Most of these methods perform binary classification, i.e., either distinguishing between normal traffic and malicious traffic or distinguishing between a single type of attack and normal traffic. Some other methods like [8, 12] classify between attacks individually, which is a similar strategy that we use, but are not able to classify all attacks simultaneously as proved in [12]. Our proposed method outperforms the rest as explained empirically in the next few subsections. All the results displayed in Table 1 are the results as stated in the respective papers of the state-of-the-art methods for the benchmark datasets of UNSW and NSL-KDD, hence making for an unbiased and credible comparison.

We use the basic extreme learning machine for all our experiments. This is because, with the basic ELM, we achieve state-of-the-art performance as well as real-time intrusion detection. There are many other variants of the ELM, but all of them are more complex versions of the basic ELM. We stick to the basic version due to the following reasons: to avoid unnecessary overhead of other versions, to keep the system simple and fast, and lastly, since the performance of our system is able to outperform previous intrusion detection systems and achieve very high accuracies (98.24% and 99.76% on the UNSW and NSL-KDD, respectively), we believe that using a more complex version of the ELM would have a negligible increase in performance while increasing the complexity of the system.

However, for some types of attacks, both UNSW and NSL-KDD datasets are highly imbalanced. This means that using the extreme learning machine version for imbalanced classification, called the weighted extreme learning machine (WELM) [46], could prove to be advantageous. Thus, we test our method by using an ensemble of WELM, replacing the basic ELM.

4.1 UNSW dataset

The UNSW dataset [5] consists of 49 features and 10 classes (9 attacks and 1 normal traffic). The UNSW dataset is by far the most realistic and difficult dataset for the problem of intrusion detection. It is one of the more difficult datasets for intrusion detection as it consists of 10 classes of attacks and the accuracies reported on this dataset are generally lower than on other datasets. It is the most widely used dataset by researchers, and we use it to empirically prove the capabilities of our system.

The individual attack detection accuracies are obtained from each ELM specifically trained to detect that particular type of attack by a one vs all strategy (one ELM for each class). All ELMs consist of a single hidden layer of 512 neurons with the ReLU activation function [38]. We also tried several other activations such as multi-quadratics, soft limit, hard limit, hyperbolic tangent, sigmoid, and linear, but we achieved best performance with ReLU.

Only one iteration of training is required to train the ELM since it is a one-shot learning algorithm. The training time for ELMs on the UNSW dataset was 18 s on each type of attack. After training the ELMs, the results are combined using the softmax layer which is trained for 10 epochs using the ADAM optimizer [43] with default settings. The softmax layer is as wide as the number of classes.

Training is performed on 175,342 instances, and the model is tested on 82,332 instances. To avoid overfitting, a small portion of the training set (20%) is used for validation.

Table 2 shows the results achieved by our model. The reason behind using the softmax layer is quite clear from Table 2 and Fig. 2 which shows the difference between the average accuracy and accuracy with the softmax layer. Figure 2 displays a bar graph representation of the results, highlighting the difference between average accuracy and softmax accuracy.

4.2 KDDcup99 dataset

The KDDcup99 dataset [6] presents with 41 features and 4 broad categories into which the attacks can be classified and 1 normal traffic category. It is one of the oldest benchmark datasets available for intrusion detection and has been widely used for the past decade. We test our system on this dataset to show that our model is generalizable to different datasets and does not overfit on a single dataset.

We train and test on the KDDcup99 dataset to show that our model is a general purpose intrusion detection system that has not been tailored for a single dataset. It works on the general problem of intrusion detection and not on a single dataset.

The system configuration used for KDDcup99 dataset is the same as the UNSW dataset (single hidden layer, 512 ReLU neurons, ensemble ELM with 1 ELM for each class, 1 training iteration, softmax layer as wide as the number of classes, trained for 10 epochs with ADAM optimizer with default settings).

Training was performed on 296,412 instances, and the model was tested on 197,608 instances. To avoid overfitting, a small portion of the training set (20%) is used for validation. Table 3 tabulates the results on this dataset, showing individual accuracies for each class of attack and the average accuracy. The softmax layer increases the accuracy slightly. The difference can be seen clearly in the bar graph in Fig. 3.

Table 2 Proposed system performance on the UNSW dataset

Attacks	Accuracy (in %)	No. of features
Normal	91.26	13
Generic	98.16	14
Exploits	89.13	25
Fuzzers	91.30	24
DoS	94.75	25
Reconnaissance	94.60	17
Analysis	98.96	16
Backdoor	99.11	18
Shellcode	99.40	12
Worms	99.92	15
Average	*95.66*	*18*
Softmax	*98.24*	-
No feature selection	*92.95*	*49*

4.3 Feature selection

We perform feature selection using the ExtraTrees classifier [4] on all features for each type of attack separately. Each feature is given a score and ranked on the basis of this score. An example of feature importance ranking is given in Fig. 4 (y-axis is the feature score and x-axis is the features), which displays the scores of all features in the UNSW dataset according to their importance in detecting the DoS attack (the features with blue bars are discarded and red ones are selected). The most important feature at the top has the highest score, and the score and importance decrease as we move down. To select the most relevant feature, instead of selecting a predefined number of features, we set a threshold on the importance score to 0.02. All features having scored below this are discarded as irrelevant for detecting a particular type of attack. So, we get different number of features for classifying different types of attacks.

From Tables 2 and 3, we can see that feature selection boosts performance of the system by $\sim 4-6\%$. When using all 49 features in UNSW and 41 features in NSL-KDD datasets, the accuracy is reduced. This means that there are some redundant features present in both the datasets, which the ExtraTrees classifier is able to remove.

Table 4 tabulates the feature scores generated by the ExtraTrees classifier for the DoS type of attack in the UNSW dataset, in ascending order. The most important characteristic of using an ensemble-based approach to feature selection is that the most important features for each attack are considered separately. This is an indispensable property because for detecting a particular type of attack, some features might be considered irrelevant and, as a result, will be discarded. But, for another type of attack, some of those discarded features might be considered important. So, it is highly desirable to have an ensemble approach to feature selection as well, so that features can be selected for the detection of different kinds of attacks separately.

Tables 2 and 3 have a column for the number of features selected for each type of attack. The number of features is decreased significantly, and at the same time,

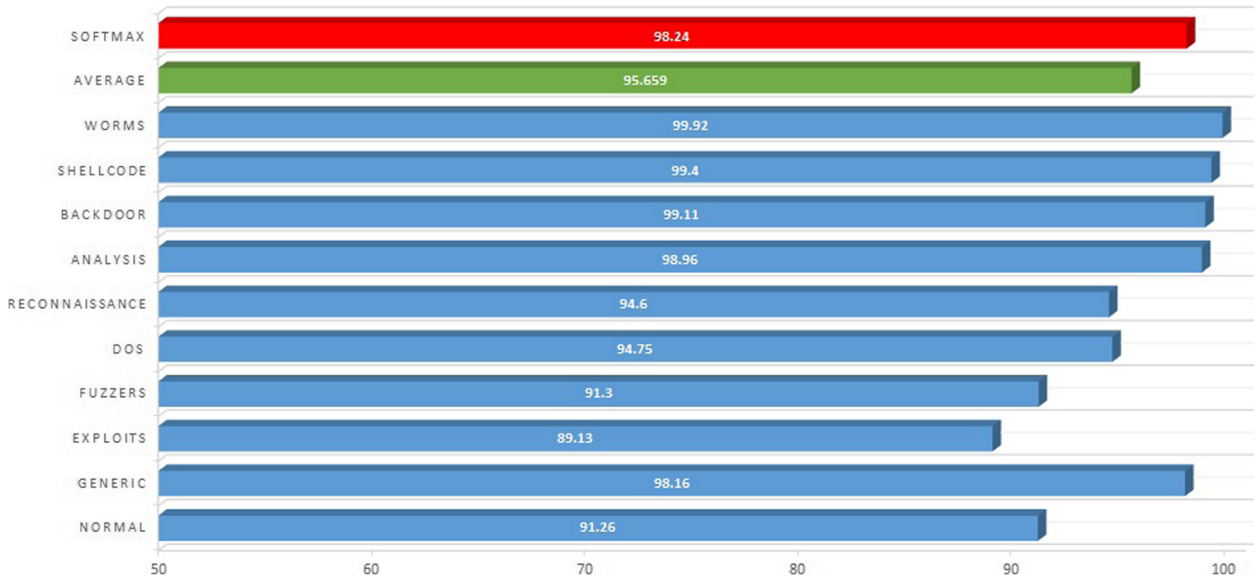

Fig. 2 Results on the UNSW dataset

detection problem for each attack gets its own feature set. The threshold limit which selects the number of features versus overall accuracy of the system is shown in Fig. 5. It is basically comparing the performance of the model with respect to the number of features, since increasing the threshold score means considering less features and vice versa.

As we can see from Fig. 5, the optimal feature selection threshold is 0.02, at which the accuracy reaches its maximum, but after which the accuracy starts to decrease. The feature selection threshold is a hyperparameter whose optimal was found by running experiments with different thresholds.

4.4 Weighted extreme learning machine

We apply the WELM algorithm in place of the ELM in our multi-layer intrusion detection system. The intuition behind using WELM is that UNSW and NSL-KDD have imbalanced class instances for some of the attack classes. Adding the class weights on top of ELM is a small overhead. The only change made to the ELM to get WELM is that Eq. 9 is changed to:

$$H^{\dagger} = \left(\frac{I}{C} + H^T W H\right)^{-1} H^T \tag{13}$$

And the MSE loss is calculated as:

$$E = \frac{1}{2}\sum_{j=1}^{N}\left(\sum_{i=1}^{L}\beta_i g(w_i.x_j + b_i) - t_j\right)^2 + CW\frac{1}{2}||\beta||^2 \tag{14}$$

where W is a $n \times n$ diagonal matrix, where each diagonal element is associated with the corresponding training sample. Each training instance is assigned a weight according to the class it belongs to and the number of instances belonging to that class. We use the weighting scheme W2 as used in [46]. We show the results of using WELM on the UNSW and NSL-KDD datasets and compare the results with ELM in Tables 5 and 6, respectively.

As shown in the tables, we get a very small performance boost by using WELM in place of ELM. Also, WELM plays its critical role in the imbalance classification problem by increasing the detection accuracy of attacks with comparatively fewer number of instances. But, there is no increase in performance for detecting backdoor, shellcode, and worms in Table 5. This might be because their

Table 3 Proposed system performance on the KDDcup99 dataset

Attacks	Accuracy (in %)	No. of features
Normal	100	13
DoS	100	14
Probe	99.25	15
R2L	98.50	14
U2R	100	14
Average	*99.55*	*14*
Softmax	*99.76*	-
No feature selection	*95.43*	*41*

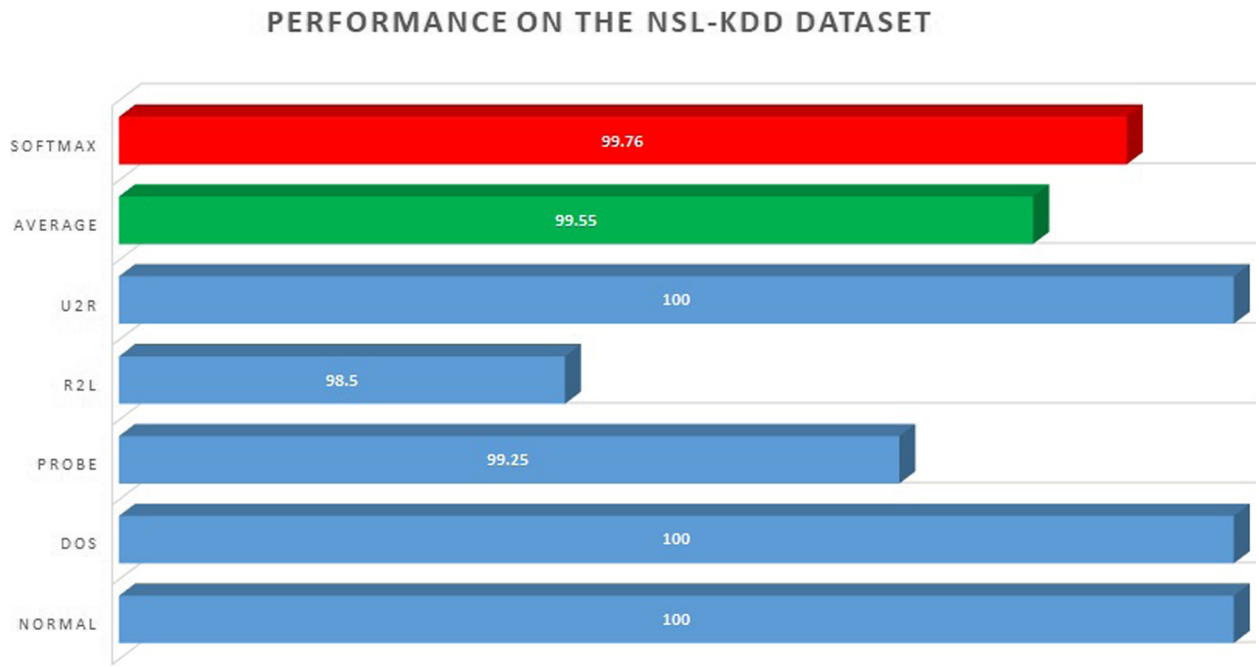

Fig. 3 Results on the KDDcup99 dataset

detection accuracy is already close to 100%, and since the dataset is quite large, some instances are bound to be misclassified. Also, note that the rest of system remains the same. So, the number of features selected by the Extra-Trees classifier is the same for both the systems (with ELM or WELM).

Overall, there is a small noticeable boost in performance of the multi-layer intrusion detection system with WELM. Furthermore, the increase in overhead is negligible since the class instance weight matrix W is easy to calculate and can be determined before training begins.

5 Discussion

The proposed method performs very well on the standard benchmark intrusion detection datasets. One advantage of the method lies in reducing the number of

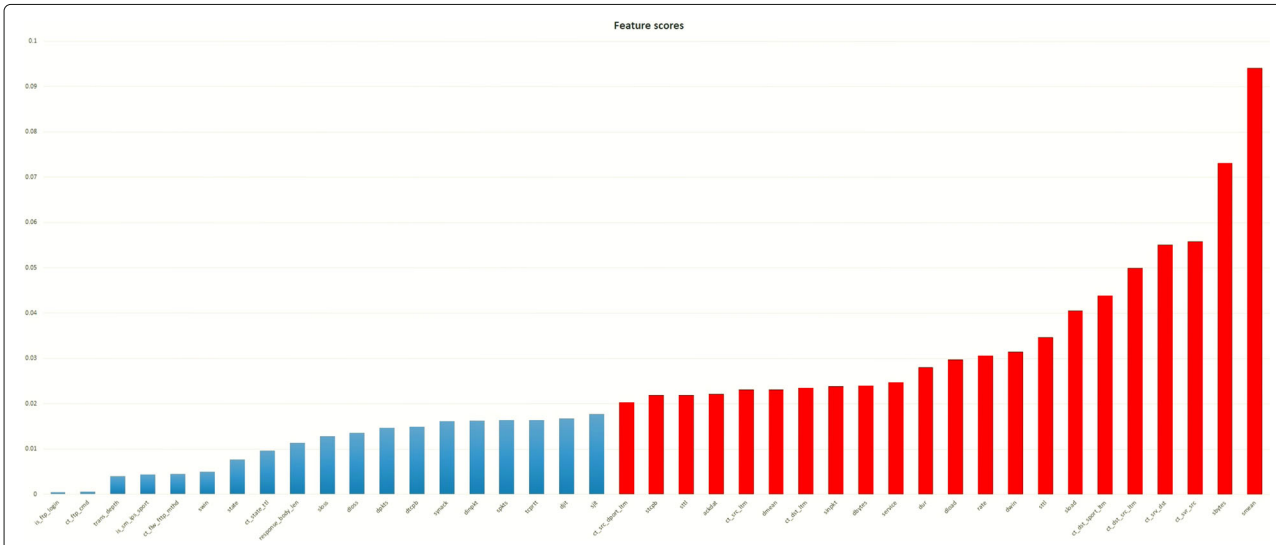

Fig. 4 Feature scores by ExtraTrees for DoS attack in the UNSW dataset

Table 4 Feature scores by ExtraTrees classifier for DoS attack in the UNSW dataset

Features	Scores
is_ftp_login	0.000429706
ct_ftp_cmd	0.000555861
trans_depth	0.003995418
is_sm_ips_sport	0.004405755
ct_flw_fttp_mthd	0.004472606
swin	0.005013446
state	0.007730736
ct_state_ttl	0.009660607
response_body_len	0.011350468
sloss	0.012895271
dloss	0.013546359
dpkts	0.014662184
dtcpb	0.014982101
synack	0.016172927
dinpkt	0.016299402
spkts	0.016379687
tcprtt	0.016397195
djit	0.016837784
sjit	0.017782288
ct_src_dport_ltm	0.020360274
stcpb	0.021925168
sttl	0.021994751
ackdat	0.02219905
ct_src_ltm	0.023103935
dmean	0.0231134
ct_dst_ltm	0.02347893
sinpkt	0.02396061
dbytes	0.024039132
service	0.024705258
dur	0.02809861
dload	0.029781236
rate	0.030608379
dwin	0.031516652
sttl	0.034695868
sload	0.040628565
ct_dst_sport_ltm	0.043927692
ct_dst_src_ltm	0.049993128
ct_srv_dst	0.055067287
ct_svr_src	0.055864165
sbytes	0.073172393
smean	0.094195716

features required to train a model by using a suitable feature selection technique such as the ExtraTrees classifier. To train a DL model, a lot of features are required since the number of parameters to be trained is more. However, in the case of ELM, a single hidden layer has fewer parameters which requires less features. Hence, feature selection can be beneficial in this case.

Also, DL methods require large amounts of data for training. But, our model is more data and memory efficient and makes use of less training data due to parallel processing and using single layer feedforward neural networks. However, if given enough data to train, some DL methods could give better performance.

Furthermore, DL methods are not able to generalize well to change in datasets. They need to be modified and trained differently if another dataset does not belong to the same distribution as the dataset it has been trained on. On the other hand, our model can be applied to another dataset without any modifications as shown above.

Overall, the proposed model uses considerably less number of features, with real-time detection due to parallel implementation, and gives state-of-the-art performance for intrusion detection on the UNSW and NSL-KDD benchmark datasets with an ensemble of ELM/WELM aggregated by a softmax layer.

6 Conclusion

Our proposed intrusion detection system is able to outperform all recent machine learning-based intrusion detection systems in terms of detection accuracy. We tested our model exhaustively on two of the most widely used datasets, UNSW and KDDcup99. We also compare our method with machine learning-based intrusion detection systems. The comparisons are fair and unbiased since all models have been compared on the same benchmark datasets. The three stage pipeline of the model can be summarized as follows: ExtraTrees classifier for feature selection (for feature scores and ranking) for each type of attack separately, ensemble of ELMs (for fast and accurate training) where each ELM detects one type of attack with its own feature set, and finally, the softmax layer for combining and fine tuning the results (for dispensing with ambiguity and increasing precision) for achieving greater accuracy and generating a interpretable probabilistic output. Our system attains accuracies of 98.24% and 99.76% on the UNSW and KDDcup99 datasets, respectively. Performance is increased even further by using WELM in place of ELM for imbalance classification, by incurring a small overhead. Also, the ELM that distinguishes between normal traffic and potential attacks enables the system to detect new attacks, but the system cannot determine the new type of attack if it is not trained on it.

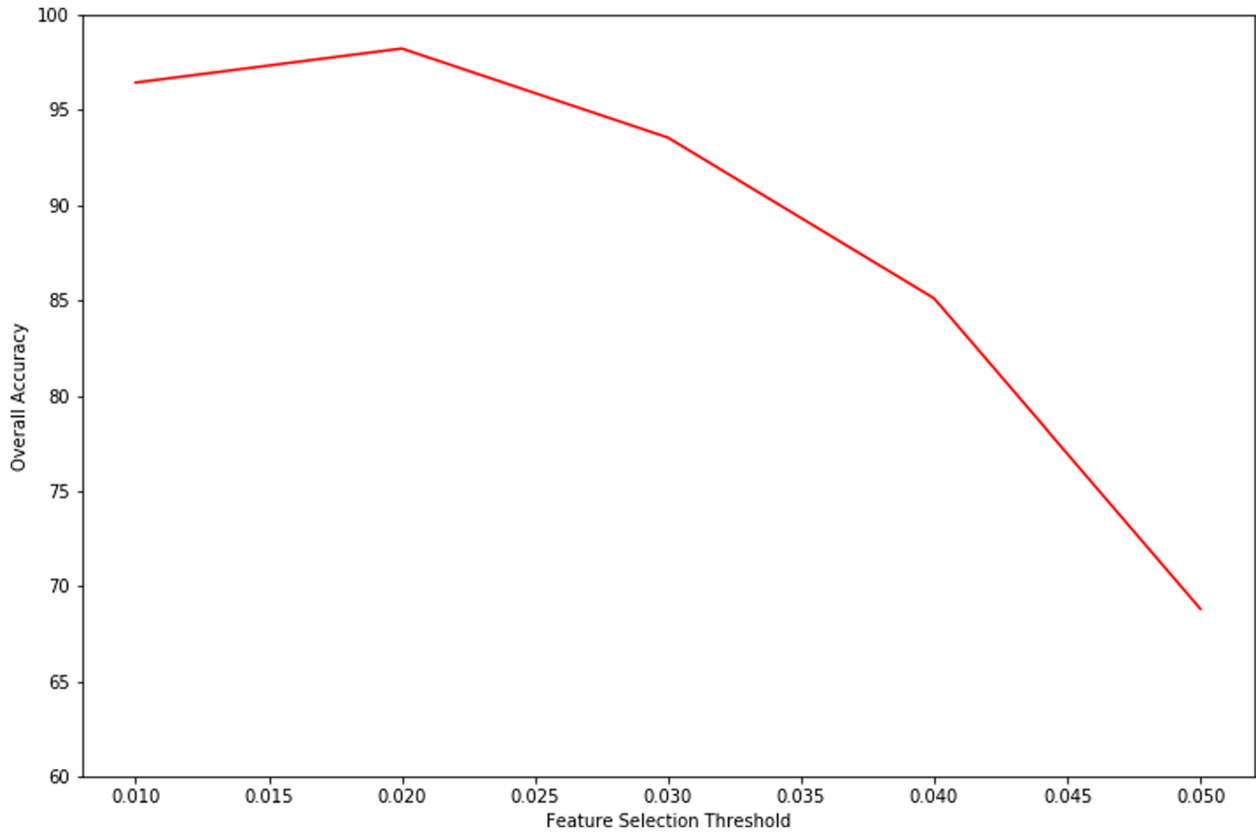

Fig. 5 Performance based on feature score threshold selection

Table 5 Proposed system based on WELM performance on the UNSW dataset

Attacks	Accuracy with ELM	Accuracy with WELM
Normal	91.26	93.54
Generic	98.16	98.23
Exploits	89.13	90.12
Fuzzers	91.30	91.47
DoS	94.75	94.90
Reconnaissance	94.60	95.33
Analysis	98.96	99.26
Backdoor	99.11	99.11
Shellcode	99.40	99.40
Worms	99.92	99.92
Average	*95.66*	*96.12*
Softmax	*98.24*	*98.69*

Table 6 Proposed system based on WELM performance on the KDDcup99 dataset

Attacks	Accuracy with ELM	Accuracy with WELM
Normal	100	100
DoS	100	100
Probe	99.25	99.25
R2L	98.50	99.33
U2R	100	100
Average	*99.55*	*99.71*
Softmax	*99.76*	*99.83*

Acknowledgements
Not applicable.

Authors' contributions
The authors' contributions are in the serial order of the names in which they appear. All authors read and approved the final manuscript.

Funding
Not applicable.

Availability of data and materials
The datasets used for our experiments are freely available online (UNSW and KDDcup99).
UNSW-NB15 dataset: https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/.
KDDcup99 dataset: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

Competing interests
The authors declare that they have no competing interests.

Received: 15 February 2019 Accepted: 29 August 2019
Published online: 22 October 2019

References

1. M. Faiz, N. B. Anuar, A. W. A. Wahab, S. Shamshirband, A. T. Chronopoulos, Source camera identification: a distributed computing approach using hadoop. J. Cloud Comput. **6**(1), 18 (2017). https://doi.org/10.1186/s13677-017-0088-x
2. M. Ganjkhani, S. N. Fallah, S. Badakhshan, S. Shamshirband, K.-w. Chau, A novel detection algorithm to identify false data injection attacks on power system state estimation. Energies. **12**(11) (2019). https://doi.org/10.3390/en12112209
3. G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: Theory and applications. Neurocomputing. **70**(1), 489–501 (2006). https://doi.org/10.1016/j.neucom.2005.12.126. Neural Networks
4. P. Geurts, D. Ernst, L. Wehenkel, Extremely randomized trees. Mach. Learn. **63**(1), 3–42 (2006). https://doi.org/10.1007/s10994-006-6226-1
5. N. Moustafa, J. Slay, in *2015 Military Communications and Information Systems Conference (MilCIS)*. Unsw-nb15: A comprehensive data set for network intrusion detection systems (unsw-nb15 network data set), (2015), pp. 1–6. https://doi.org/10.1109/MilCIS.2015.7348942
6. S. D. Bay, D. F. Kibler, M. J. Pazzani, P. Smyth, The uci kdd archive of large data sets for data mining research and experimentation. SIGKDD Explor. **2**, 81 (2000)
7. P. Mishra, E. S. Pilli, V. Varadharajant, U. Tupakula, in *2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. Nvcloudids: A security architecture to detect intrusions at network and virtualization layer in cloud environment, (2016), pp. 56–62. https://doi.org/10.1109/ICACCI.2016.7732025
8. H. Gharaee, H. Hosseinvand, in *2016 8th International Symposium on Telecommunications (IST)*. A new feature selection ids based on genetic algorithm and svm, (2016), pp. 139–144. https://doi.org/10.1109/ISTEL.2016.7881798
9. M. Idhammad, A. Karim, M. Belouch, Dos detection method based on artificial neural networks. **8** (2017). https://doi.org/10.14569/ijacsa.2017.080461
10. M. Belouch, S. E. Hadaj, M. Idhammad, A two-stage classifier approach using reptree algorithm for network intrusion detection. Int. J. Adv. Comput. Sci. Appl. **8**(6) (2017). https://doi.org/10.14569/IJACSA.2017.080651
11. N. Chowdhury, M. Ferens, K. Ferens, *Network intrusion detection using machine learning*, (2016), p. 6
12. T. Salman, D. Bhamare, A. Erbad, R. Jain, M. Samaka, in *2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud)*. Machine learning for anomaly detection and categorization in multi-cloud environments, (2017), pp. 97–103. https://doi.org/10.1109/CSCloud.2017.15
13. S. Guha, S. S. Yau, A. B. Buduru, in *2016 IEEE 14th Intl Conf on Dependable, Autonomic and Secure Computing, 14th Intl Conf on Pervasive Intelligence and Computing, 2nd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress(DASC/PiCom/DataCom/CyberSciTech)*. Attack detection in cloud infrastructures using artificial neural network with genetic feature selection, (2016), pp. 414–419. https://doi.org/10.1109/DASC-PICom-DataCom-CyberSciTec.2016.32
14. S. Siddiqui, M. S. Khan, K. Ferens, in *2017 International Joint Conference on Neural Networks (IJCNN)*. Multiscale hebbian neural network for cyber threat detection, (2017), pp. 1427–1434. https://doi.org/10.1109/IJCNN.2017.7966020
15. K. Nahiyan, K. Ferens, R. McLeod, S. Kaiser, A multi-agent based cognitive approach to unsupervised feature extraction and classification for network intrusion detection, 6 (2017)
16. M. Idhammad, K. Afdel, M. Belouch, Semi-supervised machine learning approach for ddos detection. Appl. Intell. **48**(10), 3193–3208 (2018). https://doi.org/10.1007/s10489-018-1141-2
17. L. V. Efferen, A. M. T. Ali-Eldin, in *2017 International Symposium on Networks, Computers and Communications (ISNCC)*. A multi-layer perceptron approach for flow-based anomaly detection, (2017), pp. 1–6. https://doi.org/10.1109/ISNCC.2017.8072036
18. Y. Yang, K. Zheng, C. Wu, Y. Yang, Improving the classification effectiveness of intrusion detection by using improved conditional variational autoencoder and deep neural network. Sensors. **19**, 2528 (2019). https://doi.org/10.3390/s19112528
19. N. Moustafa, G. Creech, J. Slay, in *Progress in Computing, Analytics and Networking*, ed. by P. K. Pattnaik, S. S. Rautaray, H. Das, and J. Nayak. Anomaly detection system using beta mixture models and outlier detection (Springer, Singapore, 2018), pp. 125–135
20. T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, M. Ghogho, in *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*. Deep recurrent neural network for intrusion detection in sdn-based networks, (2018), pp. 202–206. https://doi.org/10.1109/NETSOFT.2018.8460090
21. G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, in *2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No.04CH37541)*. Extreme learning machine: a new learning scheme of feedforward neural networks, vol. 2, (2004), pp. 985–9902. https://doi.org/10.1109/IJCNN.2004.1380068
22. M. J. Alizadeh, M. R. Kavianpour, M. Danesh, J. Adolf, S. Shamshirband, K.-W. Chau, Effect of river flow on the quality of estuarine and coastal waters using machine learning models. Eng. Appl. Comput. Fluid Mech. **12**(1), 810–823 (2018). https://doi.org/10.1080/19942060.2018.1528480
23. S. Faizollahzadeh Ardabili, B. Najafi, M. Alizamir, A. Mosavi, S. Shamshirband, T. Rabczuk, Using svm-rsm and elm-rsm approaches for optimizing the production process of methyl and ethyl esters. Energies. **11**(11) (2018). https://doi.org/10.3390/en11112889
24. C. Chen, K. Li, M. Duan, K. Li, in *Big Data Analytics for Sensor-Network Collected Intelligence. Intelligent Data-Centric Systems*, ed. by H.-H. Hsu, C.-Y. Chang, and C.-H. Hsu. Chapter 6 - extreme learning machine and its applications in big data processing (Academic Press, 2017), pp. 117–150. https://doi.org/10.1016/B978-0-12-809393-1.00006-4. http://www.sciencedirect.com/science/article/pii/B9780128093931000064
25. T. Hussain, S. M. Siniscalchi, C. Lee, S. Wang, Y. Tsao, W. Liao, Experimental study on extreme learning machine applications for speech enhancement. IEEE Access. **5**, 25542–25554 (2017). https://doi.org/10.1109/ACCESS.2017.2766675
26. S. Ismaeel, A. Miri, D. Chourishi, in *2015 IEEE Canada International Humanitarian Technology Conference (IHTC2015)*. Using the extreme learning machine (elm) technique for heart disease diagnosis, (2015), pp. 1–3. https://doi.org/10.1109/IHTC.2015.7238043
27. H. Lingappa, H. Suresh, S. Manvi, Medical image segmentation based on extreme learning machine algorithm in kernel fuzzy c-means using artificial bee colony method. Int. J. Intell. Eng. Syst. **11**, 128–136 (2018). https://doi.org/10.22266/ijies2018.1231.13
28. Y.-P. Zhao, G. Huang, Q.-K. Hu, J.-F. Tan, J.-J. Wang, Z. Yang, Soft extreme learning machine for fault detection of aircraft engine. Aerosp. Sci. Technol. **91**, 70–81 (2019). https://doi.org/10.1016/j.ast.2019.05.021
29. A. EL Bakri, M. Koumir, I. Boumhidi, in *2016 International Conference on Electrical and Information Technologies (ICEIT)*. Extreme learning machine for fault detection and isolation in wind turbine, (2016), pp. 174–179. https://doi.org/10.1109/EITech.2016.7519584
30. J. Xiang, M. Westerlund, D. Sovilj, G. Pulkkis, in *Proceedings of the 2014 Workshop on Artificial Intelligent and Security Workshop. AISec '14*. Using extreme learning machine for intrusion detection in a big data environment (ACM, New York, 2014), pp. 73–82. https://doi.org/10.1145/2666652.2666664

31. C. Cheng, W. P. Tay, G. Huang, in *The 2012 International Joint Conference on Neural Networks (IJCNN)*. Extreme learning machines for intrusion detection, (2012), pp. 1–8. https://doi.org/10.1109/IJCNN.2012.6252449
32. G. P. M. de Farias, A. L. I. de Oliveira, G. G. Cabral, in *Neural Information Processing*, ed. by T. Huang, Z. Zeng, C. Li, and C. S. Leung. Extreme learning machines for intrusion detection systems (Springer, Berlin, 2012), pp. 535–543
33. A. Xingshuo, L. Xing, L. Fuhong, Z. Xianwei, L. Yang, Sample selected extreme learning machine based intrusion detection in fog computing and mec. Wirel. Commun. Mob. Comput. **2018**, 1–10 (2018)
34. M. D. Tissera, M. D. McDonnell, Deep extreme learning machines: supervised autoencoding architecture for classification. Neurocomputing. **174**, 42–49 (2016). https://doi.org/10.1016/j.neucom.2015.03.110
35. M. D. Tissera, M. D. McDonnell, in *Proceedings of ELM-2014 Volume 1*, ed. by J. Cao, K. Mao, E. Cambria, Z. Man, and K.-A. Toh. Deep extreme learning machines for classification (Springer, 2015), pp. 345–354. https://doi.org/10.1007/978-3-319-14063-6_29
36. M. Uzair, F. Shafait, B. Ghanem, A. S. Mian, Representation learning with deep extreme learning machines for efficient image set classification. CoRR. **abs/1503.02445** (2015). 1503.02445
37. S. Ding, X. Xu, L. Guo, N. Zhang, J. Zhang, Deep extreme learning machines and its application to eeg classification. Math. Probl. Eng. **2015**, 1–11 (2015). https://doi.org/10.1155/2015/129021
38. X. Glorot, A. Bordes, Y. Bengio, in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics. Proceedings of Machine Learning Research, vol. 15*, ed. by G. Gordon, D. Dunson, and M. Dudik. Deep sparse rectifier neural networks (PMLR, Fort Lauderdale, 2011), pp. 315–323. http://proceedings.mlr.press/v15/glorot11a.html
39. C. Dugas, Y. Bengio, F. Bélisle, C. Nadeau, R. Garcia, in *Proceedings of the 13th International Conference on Neural Information Processing Systems. NIPS'00*. Incorporating second-order functional knowledge for better option pricing (MIT Press, Cambridge, 2000), pp. 451–457. http://dl.acm.org/citation.cfm?id=3008751.3008817
40. T. N. E. Greville, Some applications of the pseudoinverse of a matrix. **2**, 8 (1960)
41. P. Courrieu, *Fast computation of Moore-Penrose inverse matrices*, vol. 8, (2008)
42. C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. (Springer, Berlin, 2006)
43. D. P. Kingma, J. Ba, Adam: A method for stochastic optimization. CoRR. **abs/1412.6980** (2014). 1412.6980
44. S. Mannor, D. Peleg, R. Rubinstein, in *Proceedings of the 22nd International Conference on Machine Learning. ICML '05*. The cross entropy method for classification (ACM, New York, 2005), pp. 561–568. https://doi.org/10.1145/1102351.1102422
45. J. Dean, S. Ghemawat, Mapreduce: simplified data processing on large clusters. Commun. ACM. **51**(1), 107–113 (2008). https://doi.org/10.1145/1327452.1327492
46. W. Zong, G.-B. Huang, Y. Chen, Weighted extreme learning machine for imbalance learning. Neurocomputing. **101**, 229–242 (2013). https://doi.org/10.1016/j.neucom.2012.08.010

Publisher's Note