Network Intrusion Detection Using Extreme Machine Learning Algorithm with Extreme Gradient Boosting for Feature Selection

Alex Ntwiga[1] & Eric Araka[2]

[1,2]Kenyatta University, Kenya

https://doi.org/10.62049/jkncu.v5i1.206

Abstract

This study addresses the challenge of improving the performance of the Extreme Learning Machine model, particularly in accurately identifying minority classes in unbalanced datasets like UNSW-NB15 and NSL-KDD. The research question guiding this study is: How can we improve the ELM model's performance for better accuracy and minority class recognition in network intrusion detection? The methodology includes balancing the dataset to address the issue of poor minority class identification, using XGBoost for feature selection to reduce the curse of high data dimensionality, Particle Swarm Optimization finally used to optimize the model. The results show that the proposed approach outperformed other models when tested on the NSL-KDD dataset, achieving accuracies of 94.29% for binary classification and 89.02% for multiclass classification. However, on the UNSW-NB15 dataset, the model achieved a binary accuracy of 90.79%, which was lower than the performance of Random Forest (93.02%) and Decision Tree (92.76 In the multiclass classification the accuracy achieved was 78.79%, indicating underperformance compared to the other state-of-the-art models. The study concludes that although the suggested approach performs well in binary classification, future studies need to focus on improving detection accuracies of datasets that are heavily unbalanced with multiple classes like UNSW-NB15 dataset.

Keywords: Network Intrusion Detection, Feature Selection, Optimization, Class Balancing, Extreme Machine Learning, Particle Swarm Optimization

Introduction

In the 21st century, we depend heavily on the Internet for almost everything we do. Although this has led in ease to carry our day-to-day activities, it poses a great threat as well. A network intrusion can result, for example, in the loss of confidential data, monetary loss, and even reputational harm. The worldwide expenditure on cybersecurity amenities for mitigating cybercrime is expected to reach $1 trillion additively within the course of five years spanning from 2017 to 2021. By 2025, the cost of cybercrime is anticipated to rise by 15% yearly to $10.5 trillion (Morgan, 2020).

An intrusion detection system (IDS) is an entity that watches both software and hardware aspects of the network while performing its vital function of cautioning the administrator of potential risks. An IDS could either be signature-based or anomaly-based. The patterns of attacks are kept in a repository in signature-based IDS. If the incoming network traffic resembles the previously saved pattern, an alert goes off, referencing the discovered attack. This technique only works for identifying previously reported assaults, hence the database needs to be updated periodically with new signatures. This causes the database to grow proportionately. A larger repository will consequently downtrend the network traffic evaluation and surveillance process. Another shortcoming of signature-based detection is that, any modification to the assault pattern, even minor ones made by the perpetrator, can easily evade detection (ElSayed et al., 2021). Conversely, anomaly-based techniques examine typical network and system activity and spot anomalies as deviations from that pattern. They are appealing because they can sense 0-day assaults. The other benefit is that the normal activity patterns of every machine, application, or network are distinct, making it more challenging for intruders to identify what operations they can perform out stealthily. A major drawback of anomaly-based technique is that, prior undiscovered system behavior could end up being regarded as anomalies leading to high false alarm rates (Xin et al., 2018).

Over the years numerous research have been done to improve intrusion detection systems using various machine learning algorithms. There has always been a tradeoff between accuracy, execution time, computational cost, and false positive rates between various algorithms, these could be attributed to high-data dimensionality and negative correlation in irrelevant features. Albulayhi et al. (2022) points out that applying all of the features in the IDS model has a number of disadvantages, including increased computational overhead, slower training and testing times, increased storage needs as a result of the high feature count, and increased error rates as a result of irrelevant features which reduce the ability of the pertinent features to discriminate. Feature selection aims to prioritize only pertinent and essential features while eliminating useless and redundant ones. On top of that, by reducing the intricate nature of constructing a model, the feature selection strategy often improves the model's general efficacy by reducing data dimensionality as well as the cost of prediction. Another problem facing ELM-based IDS models is randomization of initial weights and bias which increases computational time and lowers the overall accuracies. Tang and Li (2021) proposed an improved PSO online regularized ELM technique using NSL-KDD and UNSW-NB15 datasets. Although the research solved the randomization problem, the algorithm showed a lack of ability to recognize minority samples, hence the article suggests the next step of the research is to figure out how to increase the rate at which minority samples on uneven datasets are recognized. Similarly, Nixon et al. (2019) developed an IDS based on KDD Cup 1992 and UNSW-NB15 and observed that class imbalance degrades the performance of minority classes that represent legitimate attacks.

To address the problems stated, this paper introduces a novel approach in which undersampling and oversampling strategies such as RandomUnderSampler and SMOTE. Then to solve the issue of high data dimensionality XGBoost is utilized. And finally, the ELM model is optimized to improve its overall performance. The proposed method is then evaluated against various models.

Related Works

A lot of research has been done to create IDS that is robust in terms of handling class imbalance, data dimensionality, and huge datasets while still maintaining large accuracies. To solve the problem of increasing levels of required human interaction and decreasing levels of detection accuracy, Shone et al. (2018) proposed the use non-symmetric deep auto-encoder (NDAE) for unsupervised feature learning. Although an accuracy of 85.42% was achieved the research points out the inability to recognize the underrepresented classes R2L and U2R in the NSL-KDD datasets. To reduce the data dimensionality, Mahfouz et al. (2020) proposed the use of InfoGainAttributeEval algorithm to select 14 relevant features from NSL-KDD dataset. The overall performance of the algorithms increased for instance; the best-performing algorithm was IBk whose performance improved from 79.35% to 84.35%. To solve the issue of class imbalance in KDD Cup 99 dataset Tan et al. (2019) used SMOTE to oversample the minority classes R2L and U2R and so an increase in performance across all the algorithms, the best performing one was Random Forest, where the accuracy moved from 92.39% to 92.57%. Tang and Li (2021) proposed regulation and optimization of ELM by creating IRELM-IPSO, the model saw an improvement compared to the normal ELM, that is from 78.29% to 85.58% in multiclassification problem and from 76.14% to 91.13% in binary problem. The research also worked on the UNSW-NB15 dataset where it was observed that neither of the models tested could identify the minority classes: Analysis, Backdoor, Shellcode and Worm attacks. On the other hand, a portion of Exploits data is mistaken for Dos, Fuzzers for Exploits, Normal for Fuzzers. An accuracy of 88.53 is achieved for the remaining 6 classes. Disha and Waheed (2021) performed binary classification on UNSW-NB15 dataset. Chi-Square test was used for feature selection by removing features that were independent of response. All algorithms tested improved after feature selection apart from RF. Decision trees performed best whose accuracies moved from 89.66% to 92.76%. Kasongo and Sun (2020) proposed use of a filter-based feature reduction technique using the XGBoost algorithm on UNSW-NB15 dataset. Under binary classification, DT performed best with an increase in its test accuracy from 88.13 to 90.85%, while in multiclassification 66.03 to 67.57%.

Proposed Approach

A systematic approach was taken to design, develop, and evaluate the proposed IDS. These entail datasets, data preprocessing, class balancing, feature extraction, model training, testing, and evaluation.

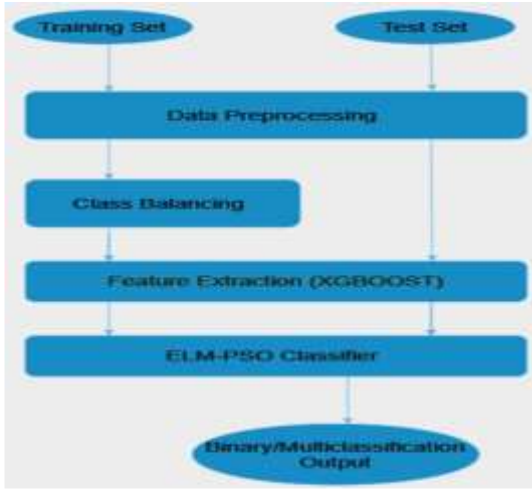

Figure 1: Proposed Model

Datasets

This stage is concerned with acquiring data from various sources. NSL-KDD dataset by Tavallaee et al. (2009) and UNSW-NB15 dataset by Moustafa (2019) are utilized in this research. Both datasets are quite unbalanced here is their distributions:

NSL-KDD Distribution

This dataset is very unbalanced with R2L and U2R occupying less than 1% while normal traffic occupies more than 50% of the dataset.

Table 1: NSL-KDD Distribution

	Training Set		**Test Set**	
Class	Count	Percentage Distribution	Count	Percentage Distribution
Normal	67,343	53.46	9,711	43.08
DoS	45,927	36.46	7,460	33.08
Probe	11,656	9.25	2,421	10.74
R2L	995	0.79	2,885	12.22
U2R	52	0.04	67	0.89

For NSL-KDD dataset, the training data consists of 22 attack types while test data contains 37 types of attack. 21 of these attacks are similar to that of training data while the remaining 16 are considered novel (Mohammadpour et al., 2018). Similar observations are made by (Saia et al., 2020) where they pointed out that the training and test portions of the dataset contain different numbers of distinct events due to occurrences of unique events in training and test set.

Table 2: Comparison of Training and Test Sets (Ding & Zhai, 2018).

Category	Training Set	Test Set
DoS	back, land, Neptune, pod, smurf, tear drop	apache2, back, land, mailbomb, Neptune, pod,smurf, teardrop, worm,process table, udpstrorm,
Probe	ipsweep, nmap, portsweep, satan	ipsweep, mscan, nmap, portsweep, saint, satan
R2L	spy, warezclient, ftpwrite, guesspasswd, imap, multihop, phf, warezmaster	ftpwrite, guesspasswd, httptunnel, imap, multihop, named, phf, sendmail, snmpguess, wxlock, warezmaster, xsnoop
U2R	bufferoverflow, ps, loadmodule, rootkit	bufferoverflow, ps, perl, loadmodule, sqlattack, xterm
Normal	normal	normal

Due to this distribution of attack in the training set and test it's important to use them separately as intended. Since we can consider the unique events in the test set as zero-day attacks. If the training set is split high accuracies of above 99% and 97% are achieved across various research in both binary and multiclass respectively.

UNSW-NB15 Distribution

This data set is also unevenly distributed. The first three groups occupy about three-quarters of the entire set and the last four groups formed less than 10% of the set.

Table 3: UNSW-NB15 Distribution

	Training Set		**Test Set**	
Class	Count	Percentage Distribution	Count	Percentage Distribution
Normal	56,000	31.94	37,000	44.93
Generic	40,000	22.81	18,871	22.92
Exploits	33,393	19.04	11,132	13.52
Fuzzers	18,184	10.37	6,062	7.36
DoS	12,264	6.99	4,089	4.97
Reconnaissance	10,491	5.98	3,496	4.25
Analysis	2,000	1.14	677	0.82
Backdoor	1,746	1.00	583	0.71
Shellcode	1,133	0.65	378	0.46
Worms	130	0.07	44	0.05
Total	175,341		82,332	

Data Preprocessing

Preprocessing stage basically deals with data preparation to make sure it's in the most suitable format for the models to learn from. The datasets were taken through the following stages:

Label encoding – This is the process of converting any categorical data to a numeric value.

Feature scaling – The data values are converted to a similar scale. This helps in speeding up the calculations. It also improves accuracy by reducing bias towards data with large values. MaxAbsScaler sets the values in the range of (-1,1)

Class Balancing

Oversampling of the minority class and/or under sampling if the majority class ensures that enough data is available for the model to learn from without bias towards the majority classes. Oversampling technique used is SMOTE, while under sampling technique used is RandomUnderSampler.

Feature Selection

Feature selection endeavors to eliminate irrelevant and redundant features by choosing the most pertinent and important features. This is a vital step in machine learning process since it helps to minimize fitting issues, decrease adaption effectiveness on test data and shortens training timeframes (Talukder et al., 2023).

XGBoost provides built-in feature importance scores based on various scales such as gain, cover, and weight. These metrics quantify how useful each feature is for constructing the boosted decision tree within the model. From this ranking, one can then select n number of features to train other models.

Extreme Learning Machine (ELM)

The ELM has been extensively utilized in cx-image processing, medical diagnosis, fault inspection, traffic sign identification, and other domains and has some advantages in solving data fitting, regression, classification, pattern recognition, and other related problems. The ELM model has three layers as shown below. The training process for the ELM algorithm consists of three steps: providing random weights to the input-hidden layer, computing the output hidden layer matrix, and calculating the output layer weights using the Moore-Penrose equation.

Table 4

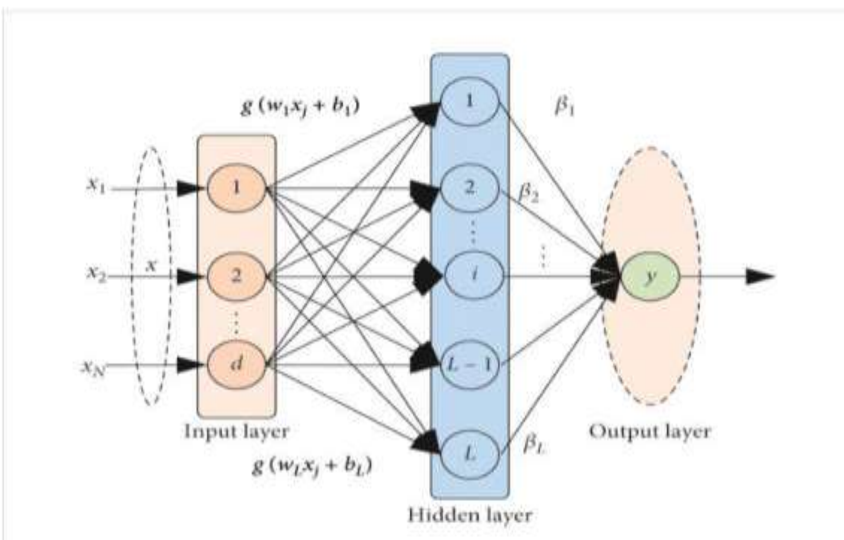

Figure 2: Extreme Learning Machine by (Zang et al., 2021)

The algorithm is based on Moore Penrose matrix theory. (Jingming et al., 2020; Xu et al., 2019) Suppose the number of neurons in the hidden layer of ELM is L and the number of training samples is N,

there are N arbitrary samples

$$(Xi, ti) \text{ where } Xi = (Xi1, Xi2, Xi3, \ldots \ldots \ldots Xin) \quad (1)$$

$$T \in Rn, ti = (ti1, ti2, ti3 \ldots \ldots \ldots \ldots tin)T \in Rm,$$

A single hidden layer neural network with L hidden layer nodes can be expressed as:

$$FL(x) = \sum \beta iLi = lgi(x) \quad (2)$$

$$= \sum \beta iLi = lgi(Wi * Xj + bi)$$

$$j = 1,2,3,4 \ldots . N$$

Where:
L is a number of hidden units
N is a number of training samples
β is weight vector between the hidden layer and output
w is a weight vector between input and hidden layer
g is an activation function
b is a bias vector
x in an input vector.

Particle Swarm Optimization (PSO)

Wang et al, (2021) observe that PSO algorithm models social behavior in a variety of animals, including insects, herds, birds, and fish. These swarms follow a cooperative method of locating food, and each member of the swarms continuously modifies the search pattern in response to its own and other members' learning experiences. According to Jain et al, (2018) the initial particles are created by the PSO, and their initial velocities are allocated to them. It determines the optimal function value and the best placement by evaluating the objective function at each particle position. Based on the present velocity, each particle's optimal location, and the optimal locations of its neighbors, it selects new velocities. The particle locations, velocities, and neighbors are then iteratively updated (the new position is the old location plus the velocity). The algorithm iterates until it encounters a stopping requirement.

The advantage of the PSO over evolution algorithms like the Genetic Algorithm (GA) is that it is simpler to construct and only requires a few parameters. At each iteration, the position and velocity are modified in an effort to find the best possible solution using the following algorithms: (Ali et al., 2018)

$$v_i(k+1) = wv_ik + c1r1\ (xbest, local - x_i) + \quad c2r2(xbest, global - x_i) \quad (3)$$

Each particle's position and velocity are represented as
$x_i = (x_{i1}, x_{i2}, \ldots\ldots, x_{id})$ and $v_k = (v_{k1}, v_{k2}, \ldots\ldots, v_{kd})$ respectively.
c1 and c2 are acceleration factors known as cognitive and social parameters. r1 and r2 are random numbers between 0 and 1. k is the iteration index. w is the inertia weight parameter.
The PSO updates the particle position using the following equation.

$$x_i(k+1) = xk + v(k+1) \quad (4)$$

Experiment And Discussion

This section presents the experiments of the study to show the performance of the proposed approach in binary and multiclassification problems. After balancing the training set, only 5% is used. This is because of the bulkiness of the data and the need to reduce the training time. The test set will be used whole as provided. All experiments are implemented on a Jupyter Notebook platform on a personal computer with the following specification: Operating system - Microsoft Windows 10 Enterprise, System Model - HP 15 Notebook PC, Processor - Intel(R) Core (TM) i5-3230M CPU @ 2.60GHz, 2601 Mhz, 2 Core(s), 4 Logical Processor(s), RAM - 8.00 GB

NSL-KDD

Binary

In binary classification, the classes are classified as either attack or normal.

Figure 3: Unnormalized data

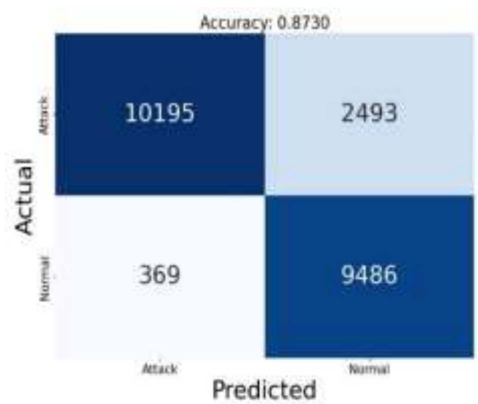

Figure 4: Normalized Data

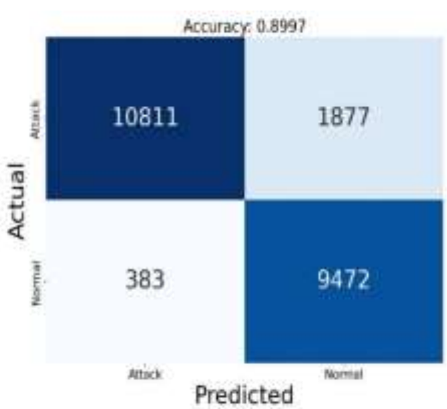

Figure 5: Balanced data with all features

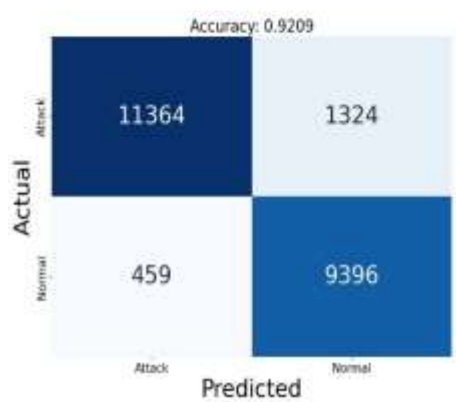

Figure 6: Balanced data with selected features

Figure 7: Results after optimization

Algorithm	Acc	Prec	Recall	F1
Random Forest	0.819	0.8564	0.819	0.8183
SVM	0.8226	0.841	0.8226	0.823
Decision Trees	0.8785	0.8932	0.8785	0.8789
Naive Bayes	0.5101	0.5518	0.5101	0.4922
XGBoost	0.8655	0.8815	0.8655	0.8659
MLP	0.807	0.8313	0.807	0.8071
LR	0.8128	0.8348	0.8128	0.813
GB	0.8612	0.8768	0.8612	0.8616
KNN	0.7964	0.8273	0.7964	0.796
ELM	0.873	0.8894	0.873	0.8734

Table 5: Comparison all features unbalanced

Algorithm	Acc	Prec	Recall	F1
Random Forest	0.8983	0.906	0.8983	0.8987
SVM	0.8919	0.8992	0.8919	0.8923
Decision Trees	0.8827	0.8912	0.8827	0.8831
Naive Bayes	0.5682	0.5624	0.5682	0.5634
XGBoost	0.8982	0.9038	0.8982	0.8986
MLP	0.894	0.9032	0.894	0.8944
LR	0.9	0.9047	0.9	0.9003
GB	0.9113	0.9155	0.9113	0.9116
KNN	0.8609	0.88	0.8609	0.8612
ELM	0.904	0.9095	0.904	0.9043
Optimized ELM	0.943	0.9434	0.943	0.943

Table 6: Comparison all features balanced

Without normalization, it's clear that the model is not learning. The model is biased towards the normal class. But after normalization it's visible that the model is now learning, the accuracy moves from 44.83% to 87.30%. Note that the normalized data is also unbalanced with all features. An improvement is observed by balancing the dataset from 87.30% to 89.97%. Further, an improvement is observed after applying feature selection, from 89.97% to 92.09%. And finally, after optimization, the accuracy improves further from 92.09% to 94.30%.

After balancing the dataset and selecting only the necessary features from the dataset, it's seen the accuracies of all models increase. Also, it's clear that the proposed model outperformed the rest with an accuracy of 94.3

Multiclassification

In multiclassification, the classification are made into either Normal, Access, Probe DoS or Privilege

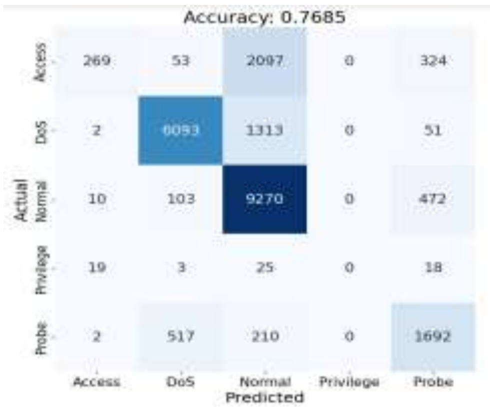

Figure 8: Unbalanced with all features

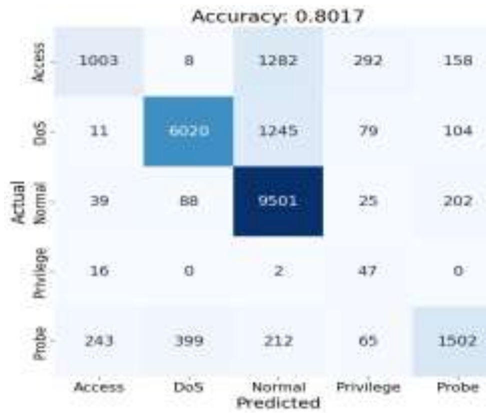

Figure 9: Balanced dataset and all features

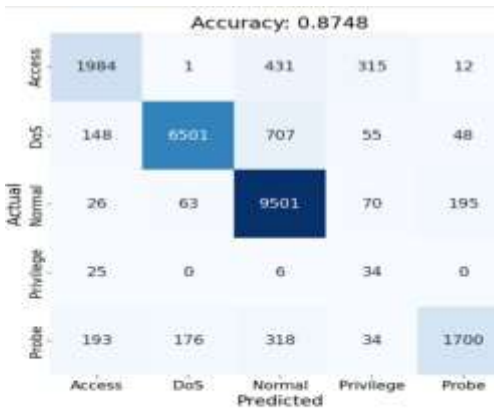

Figure 10: Balanced dataset with selected features

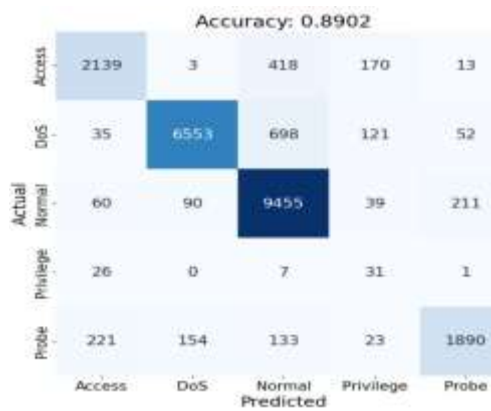

Figure 11: Results after Optimization

Algorithm	Acc	Prec	Recall	F1
Random Forest	0.7626	0.8129	0.7626	0.7228
SVM	0.7582	0.7785	0.7582	0.721
Decision Trees	0.7928	0.775	0.7928	0.769
Naive Bayes	0.5235	0.6914	0.5235	0.5339
XGBoost	0.7873	0.8154	0.7873	0.7577
MLP	0.7732	0.783	0.7732	0.7346
LR	0.777	0.7987	0.777	0.7463
GB	0.7852	0.8186	0.7852	0.7613
KNN	0.7668	0.7753	0.7668	0.7316
ELM	0.7685	0.7911	0.7685	0.7338

Table 7: Comparison of all features unbalanced

Algorithm	Acc	Prec	Recall	F1
Random Forest	0.8155	0.8443	0.8155	0.8059
SVM	0.8198	0.8499	0.8198	0.8213
Decision Trees	0.79	0.7943	0.79	0.7814
Naive Bayes	0.5883	0.7094	0.5883	0.6102
XGBoost	0.8176	0.827	0.8176	0.8123
MLP	0.8193	0.8433	0.8193	0.8108
LR	0.8067	0.8369	0.8067	0.8041
GB	0.8203	0.8318	0.8203	0.8143
KNN	0.844	0.8629	0.844	0.842
ELM	0.8748	0.8931	0.8748	0.8801
ELM_PSO	0.8902	0.9036	0.8902	0.8946

Table 8: Comparison of selected features balanced

Initially, it observed that privilege (U2R) attacks were barely recognized, while over 90% of Access (R2L) were being misclassified. But after balancing, it's seen that 72% of privilege and 36% of access attacks have been now correctly classified. Overall accuracy also improves from 76.89% to 80.17%. After feature selections accuracies of all categories improved apart from privilege attack which dropped, and normal which remained the same. These made the overall accuracy jump from 80.17% to 87%. Finally, after optimization, the accuracy moved to 89.02%. A huge overall improvement is seen in Access class from the initial 10% to 78% and privilege from the initial 0% to 48%

When comparing with other machine learning models, accuracies have increased for all models after selecting features and balancing the dataset. The proposed algorithm ELM-PSO outperformed the rest of the models with an accuracy of 89.02%.

Table 9: Comparison of proposed model against state of art models. (NSL-KDD dataset)

Author	Best Algorithms	Classification Type	Accuracy
(Tang & Li, 2021)	IRELM-IPSO	Multiclass Binary	85.58 91.13
(Su et al., 2020)	BAT-MC	Multiclass	84.25
(Ding & Zhai, 2018)	CNN	Multiclass	80.13
(Lopez-Martin et al., 2019)	Linear-Model	Multiclass Binary	80.7 88.7
(Ji et al., 2018)	MLP	Binary	73.8
(Yin et al., 2017)	RNN	Multiclass Binary	81.29 83.28
Proposed Model	ELM-PSO	Multiclass Binary	89.02 94.30

UNSW-NB15

Binary

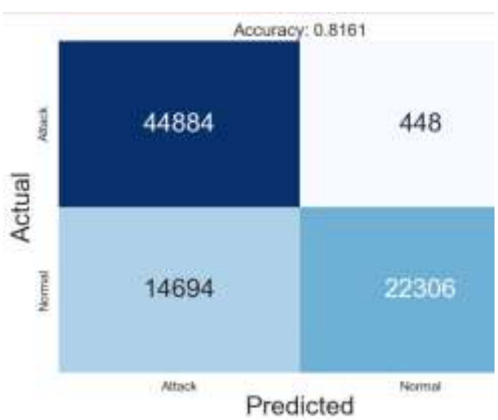

Figure 12: Unbalanced with all features

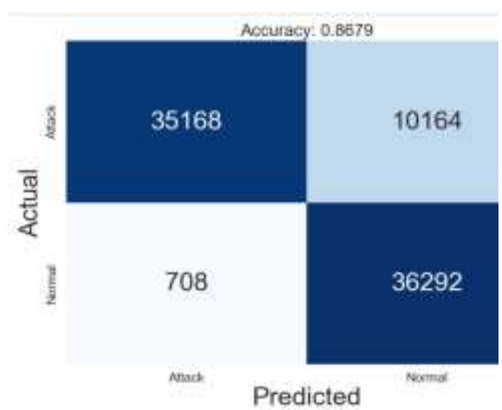

Figure 13: Balanced with all features

.

Figure 14: Balanced with selected features

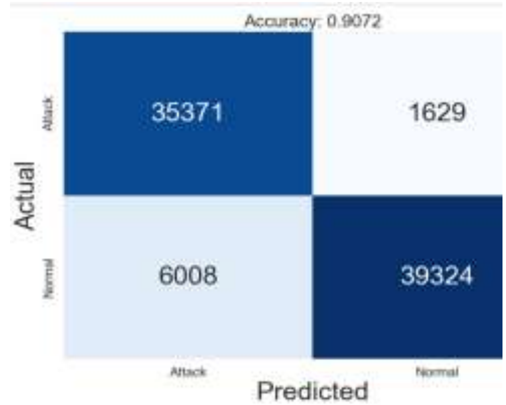

Figure 15: Results after optimization

Algorithm	Acc	Prec	Recall	F1
Random Forest	0.8594	0.8807	0.8594	0.8553
SVM	0.8117	0.8581	0.8117	0.8007
Decision Trees	0.859	0.8677	0.859	0.8566
Naive Bayes	0.551	0.7753	0.551	0.4712
XGBoost	0.865	0.8793	0.865	0.862
MLP	0.837	0.8608	0.837	0.8315
LR	0.8024	0.8364	0.8024	0.7927
GB	0.8626	0.881	0.8626	0.8589
KNN	0.8359	0.8535	0.8359	0.8313
ELM	0.8161	0.8554	0.8161	0.8066

Table 10: Comparison with all features unbalanced

Algorithm	Acc	Prec	Recall	F1
Random Forest	0.9302	0.9332	0.9302	0.9304
SVM	0.8217	0.865	0.8217	0.8199
Decision Trees	0.8935	0.8966	0.8935	0.8937
Naive Bayes	0.722	0.7566	0.722	0.7028
XGBoost	0.9173	0.921	0.9173	0.9176
MLP	0.8855	0.8961	0.8855	0.8857
LR	0.7092	0.7795	0.7092	0.6997
GB	0.9029	0.9097	0.9029	0.9031
KNN	0.8902	0.9029	0.8902	0.8903
ELM	0.9013	0.9073	0.9013	0.9016
Optimized ELM	0.9072	0.9128	0.9072	0.9075

Table 11: Comparison selected feature balanced set

An increase in accuracy is observed in all models after balancing and selecting the important features. In this case, the proposed model had an accuracy of 90.72% which underperformed XGBoost (91.73%) and RF (93.02%).

Multiclassification

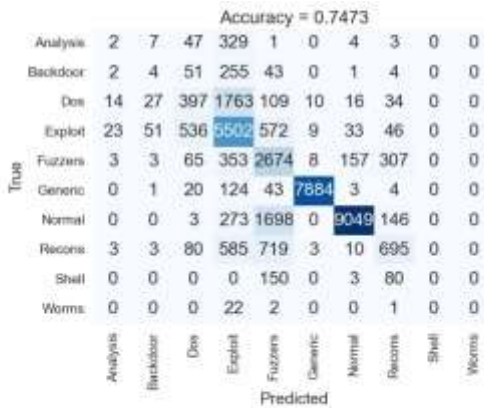

Figure 16: All features and unbalanced

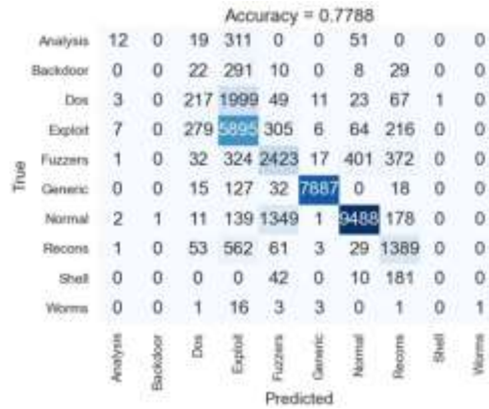

Figure 17: Selected feature and unbalanced

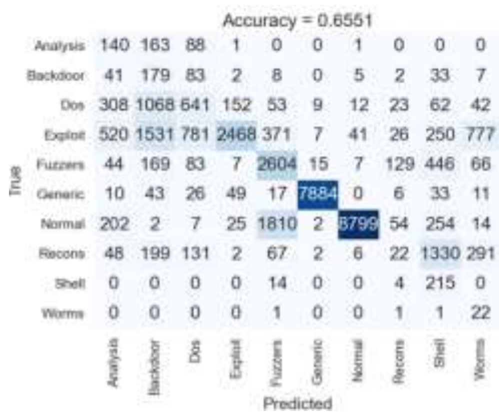

Figure 18: Selected feature and balanced and unbalanced

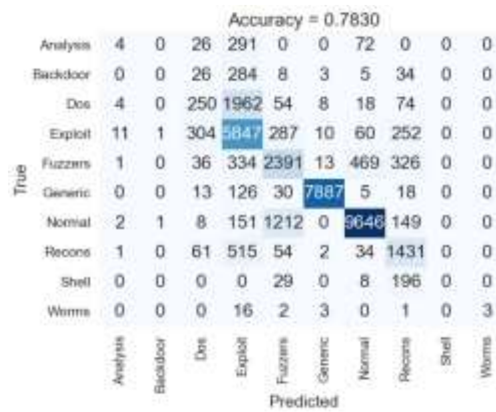

Figure 19: Optimized selected and unbalanced

In order to gauge the model's performance, it was tested against other models as shown below:

Algorithm	Acc	Prec	Recall	F1
Random Forest	0.7927	0.7884	0.7927	0.7888
SVM	0.7502	0.7641	0.7502	0.727
Decision Trees	0.7634	0.7715	0.7634	0.7669
Naive Bayes	0.5358	0.6196	0.5358	0.5053
XGBoost	0.8004	0.794	0.8004	0.7945
MLP	0.7744	0.7693	0.7744	0.7635
LR	0.7494	0.7529	0.7494	0.7394
GB	0.7944	0.7838	0.7944	0.7844
KNN	0.7417	0.7465	0.7417	0.7419
ELM	0.7473	0.7557	0.7473	0.7385

Table 12: Comparison of all features unbalanced

Algorithm	Acc	Prec	Recall	F1
Random Forest	0.7989	0.7947	0.7989	0.7951
SVM	0.7465	0.7435	0.7465	0.7224
Decision Trees	0.7704	0.7765	0.7704	0.773
Naive Bayes	0.5218	0.6802	0.5218	0.5155
XGBoost	0.8021	0.7953	0.8021	0.7957
MLP	0.7784	0.7633	0.7784	0.7559
LR	0.7222	0.725	0.7222	0.7151
GB	0.8018	0.791	0.8018	0.7897
KNN	0.7801	0.7795	0.7801	0.7784
ELM	0.7794	0.7724	0.7794	0.7636
ELM_PSO	0.783	0.7659	0.783	0.764

Table 13: Comparison of selected and unbalanced

In UNSW_NB15, the model did not perform as well as expected when compared with the existing research. Attempt to balance the classes led to a drop in the overall accuracies from 77.88% to 65.51%, hence an approach of using the unbalanced dataset was used. Although optimizing the model's performance improved its accuracy to 78.3%, the model still underperformed some of the models it was compared to. The best-performing model was XGBoost at 80.21%.

Table 12: Comparison of proposed model against state of art models. (UNSW-NB15 dataset)

Author	Best Algorithims	Classification Type	Accuracy
(Tang & Li, 2021)	IRELM-IPSO	Multiclass	88.53
(Lopez-Martin et al., 2019)	CNN	Multiclass Binary	78.2 89.8
(Jing & Chen, 2019)	SVM	Multiclass Binary	75.77 85.99
(Meftah et al., 2019)	DT SVM	Multiclass Binary	86 82.11
(Disha & Waheed, 2021)	DT	Binary	92.76
(Kasongo & Sun, 2020)	ANN	Multiclass Binary	77.51 86.71
(Almomani, 2020)	J48	Binary	90.48
Proposed Model	ELM-PSO	Multiclass Binary	78.3 90.72
	DT RF	Multiclass Binary	80.21 93.06

Conclusions and Recommendations.

The ELM-PSO model proposed for NSL-KDD model outperformed all other models used within this research and those used by other researchers. While the UNSW-NB15 binary outperformed other research apart from one where an accuracy of 92.76% was achieved using DT. Although it is important to note that RF tested in this research performed better than that with an accuracy of 93.06%. Under multiclassification of UNSW-NB15, all models tested in this research performed poorly against the state of art models. Hence a different approach needs to be undertaken where many classes are underrepresented in a dataset.

This research recommends that future studies need to focus on improving the accuracy of the UNSW-NB15 dataset without dropping the minority classes, as observed in various research works. Also, datasets produced in the future should have a well reasonable distribution of the attack classes.

References

Albulayhi, K., Abu Al-Haija, Q., Alsuhibany, S. A., Jillepalli, A. A., Ashrafuzzaman, M., & Sheldon, F. T. (2022). IoT Intrusion Detection Using Machine Learning with a Novel High Performing Feature Selection Method. *Applied Sciences*, *12*(10), Article 10. https://doi.org/10.3390/app12105015

Ali, M. H., Fadlizolkipi, M., Firdaus, A., & Khidzir, N. Z. (2018). A hybrid Particle swarm optimization - Extreme Learning Machine approach for Intrusion Detection System. *2018 IEEE Student Conference on Research and Development (SCOReD)*, 1–4. https://doi.org/10.1109/SCORED.2018.8711287

Almomani, O. (2020). A Feature Selection Model for Network Intrusion Detection System Based on PSO, GWO, FFA and GA Algorithms. *Symmetry*, *12*(6), Article 6. https://doi.org/10.3390/sym12061046

Ding, Y., & Zhai, Y. (2018). Intrusion Detection System for NSL-KDD Dataset Using Convolutional Neural Networks. *Proceedings of the 2018 2nd International Conference on Computer Science and Artificial Intelligence*, 81–85. https://doi.org/10.1145/3297156.3297230

Disha, R. A., & Waheed, S. (2021). A Comparative study of machine learning models for Network Intrusion Detection System using UNSW-NB 15 dataset. *2021 International Conference on Electronics, Communications and Information Technology (ICECIT)*, 1–5. https://doi.org/10.1109/ICECIT54077.2021.9641471

ElSayed, M. S., Le-Khac, N.-A., Albahar, M. A., & Jurcut, A. (2021). A novel hybrid model for intrusion detection systems in SDNs based on CNN and a new regularization technique. *Journal of Network and Computer Applications*, *191*, 103160. https://doi.org/10.1016/j.jnca.2021.103160

Jain, N. K., Nangia, U., & Jain, J. (2018). A Review of Particle Swarm Optimization. *Journal of The Institution of Engineers (India): Series B*, *99*(4), 407–411. https://doi.org/10.1007/s40031-018-0323-y

Ji, H., Kim, D., Shin, D., & Shin, D. (2018). A Study on Comparison of KDD CUP 99 and NSL-KDD Using Artificial Neural Network. In J. J. Park, V. Loia, G. Yi, & Y. Sung (Eds.), *Advances in Computer Science and Ubiquitous Computing* (Vol. 474, pp. 452–457). Springer Singapore. https://doi.org/10.1007/978-981-10-7605-3_74

Jing, D., & Chen, H.-B. (2019). SVM Based Network Intrusion Detection for the UNSW-NB15 Dataset. *2019 IEEE 13th International Conference on ASIC (ASICON)*, 1–4. https://doi.org/10.1109/ASICON47005.2019.8983598

Jingming, L., Xuhui, L., Daoming, D., Sumei, R., & Xuhui, Z. (2020). Research on Credit Risk Measurement of Small and Micro Enterprises Based on the Integrated Algorithm of Improved GSO and ELM. *Mathematical Problems in Engineering*, *2020*, e3490536. https://doi.org/10.1155/2020/3490536

Kasongo, S. M., & Sun, Y. (2020). Performance Analysis of Intrusion Detection Systems Using a Feature Selection Method on the UNSW-NB15 Dataset. *Journal of Big Data*, *7*(1), 105. https://doi.org/10.1186/s40537-020-00379-6

Lopez-Martin, M., Carro, B., Sanchez-Esguevillas, A., & Lloret, J. (2019). Shallow neural network with kernel approximation for prediction problems in highly demanding data networks. *Expert Systems with Applications*, *124*, 196–208. https://doi.org/10.1016/j.eswa.2019.01.063

Mahfouz, A. M., Venugopal, D., & Shiva, S. G. (2020). Comparative Analysis of ML Classifiers for Network Intrusion Detection. In X.-S. Yang, S. Sherratt, N. Dey, & A. Joshi (Eds.), *Fourth International Congress on Information and Communication Technology* (Vol. 1027, pp. 193–207). Springer Singapore. https://doi.org/10.1007/978-981-32-9343-4_16

Meftah, S., Rachidi, T., & Assem, N. (2019). Network Based Intrusion Detection Using the UNSW-NB15 Dataset. *International Journal of Computing and Digital Systems*, *8*(5), 477–487. https://doi.org/10.12785/ijcds/080505

Mohammadpour, L., Ling, T. C., Liew, C. S., & Chong, C. Y. (2018). *A Convolutional Neural Network for Network Intrusion Detection System*.

Morgan, S. (2020, October 26). Cybercrime To Cost the World $10.5 Trillion Annually By 2025. *Cybercrime Magazine*. https://cybersecurityventures.com/annual-cybercrime-report-2019/

Moustafa, N., Turnbull, B., & Choo, K.-K. R. (2019). An Ensemble Intrusion Detection Technique Based on Proposed Statistical Flow Features for Protecting Network Traffic of Internet of Things. *IEEE Internet of Things Journal*, *6*(3), 4815–4830. IEEE Internet of Things Journal. https://doi.org/10.1109/JIOT.2018.2871719

Nixon, C., Sedky, M., & Hassan, M. (2019). Practical Application of Machine Learning based Online Intrusion Detection to Internet of Things Networks. *2019 IEEE Global Conference on Internet of Things (GCIoT)*, 1–5. https://doi.org/10.1109/GCIoT47977.2019.9058410

Saia, R., Carta, S., Recupero, D., & Fenu, G. (2020). A Feature Space Transformation to Intrusion Detection Systems: *Proceedings of the 12th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management*, 137–144. https://doi.org/10.5220/0009982901370144

Shone, N., Ngoc, T. N., Phai, V. D., & Shi, Q. (2018). A Deep Learning Approach to Network Intrusion Detection. *IEEE Transactions on Emerging Topics in Computational Intelligence*, *2*(1), 41–50. IEEE Transactions on Emerging Topics in Computational Intelligence. https://doi.org/10.1109/TETCI.2017.2772792

Su, T., Sun, H., Zhu, J., Wang, S., & Li, Y. (2020). BAT: Deep Learning Methods on Network Intrusion Detection Using NSL-KDD Dataset. *IEEE Access*, *8*, 29575–29585. IEEE Access. https://doi.org/10.1109/ACCESS.2020.2972627

Talukder, Md. A., Hasan, K. F., Islam, Md. M., Uddin, Md. A., Akhter, A., Yousuf, M. A., Alharbi, F., & Moni, M. A. (2023). A dependable hybrid machine learning model for network intrusion detection. *Journal of Information Security and Applications*, *72*, 103405. https://doi.org/10.1016/j.jisa.2022.103405

Tan, X., Su, S., Huang, Z., Guo, X., Zuo, Z., Sun, X., & Li, L. (2019). Wireless Sensor Networks Intrusion Detection Based on SMOTE and the Random Forest Algorithm. *Sensors*, *19*(1), Article 1. https://doi.org/10.3390/s19010203

Tang, Y., & Li, C. (2021). An Online Network Intrusion Detection Model Based on Improved Regularized Extreme Learning Machine. *IEEE Access*, *9*, 94826–94844. IEEE Access. https://doi.org/10.1109/ACCESS.2021.3093313

Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009). A detailed analysis of the KDD CUP 99 data set. *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 1–6. https://doi.org/10.1109/CISDA.2009.5356528

Wang, Z., Zeng, Y., Liu, Y., & Li, D. (2021). Deep Belief Network Integrating Improved Kernel-Based Extreme Learning Machine for Network Intrusion Detection. *IEEE Access*, *9*, 16062–16091. IEEE Access. https://doi.org/10.1109/ACCESS.2021.3051074

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. IEEE Access. https://doi.org/10.1109/ACCESS.2018.2836950

Xu, X., Chen, Q., Ren, M., Cheng, L., & Xie, J. (2019). Combustion Optimization for Coal Fired Power Plant Boilers Based on Improved Distributed ELM and Distributed PSO. *Energies*, *12*(6), Article 6. https://doi.org/10.3390/en12061036

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. IEEE Access. https://doi.org/10.1109/ACCESS.2017.2762418

Zang, S., Cheng, Y., Wang, X., & Yan, Y. (2021). Transfer Extreme Learning Machine with Output Weight Alignment. *Computational Intelligence and Neuroscience*, *2021*, e6627765. https://doi.org/10.1155/2021/6627765