American Journal of Agricultural Science, Engineering, and Technology (AJASET)

ISSN: 2158-8104 (ONLINE), 2164-0920 (PRIN

VOLUME 10 ISSUE 3 (2026)

PUBLISHED BY
E-PALLI PUBLISHERS, DELAWARE, USA

American Journal of Agricultural Science, Engineering, and Technology (AJASET)

Volume 10 Issue 3, Year 2026
ISSN: 2158-8104 (Online), 2164-0920 (Print)
DOI: https://doi.org/10.54536/ajaset.v10i3.8373
https://journals.e-palli.com/home/index.php/ajaset

A Stacking Ensemble with Heterogeneous Base Classifiers for DDoS and Multi-Class Network Attack Detection on the CICIDS2017 Dataset

Segun Omotayo Olorunyomi[1], Joshephine Olamatanmi Mebawondu[1], Oluwatoyin Bunmi Abiola[1], Saidat Adebukola Onashoga[2], Ayodeji Ireti Fasiku[3*], Adeyemi Folasade[4]

Article Information

Received: June 03, 2026

Accepted: August 16, 2026

Published: September 29, 2026

Keywords

CICIDS2017 Dataset, Ddos, Heterogeneous Classifiers, Intrusion Detection, Stacking Ensemble

ABSTRACT

Volumetric and protocol floods are two most dangerous types of DDoS attacks, which affect network availability. The overall attack picture has become more complicated with advancement in technology, however, attacks can be of all shapes and sizes and many are lurking at the fringes of the everyday. One classifier will tend to overfit towards the majority class and the loss of this bias will cost a lot when the traffic comes from different families of attacks and the traffic is imbalanced. This problem can be address using stacking ensemble, Random Forest, an XGBoost model, k-Nearest Neighbours (kNN) and multilayer perceptron (MLP) classifiers. They are used as base classifiers to input into a logistic regression meta-learner; the design aims at both binary DDoS detection and finer multi-class classification. The model was carefully designed and tested using the CICIDS2017 dataset, and was developed by implementing and applying a systematic preprocessing pipeline that involves dealing with infinite and missing values, reducing correlated features, standardising input variables, and directly tackling class imbalance. The ensemble had an accuracy of 99.31%, macro-averaged F1 of 97.84% and 0.42% false positive based on the experimental simulation and outperform the existing base learners including a majority-voting baseline. Hence, the diversity of the base learners is more important than the number of base learners, and that stacking makes the diversity into a detector that is both accurate, and realistically deployable.

INTRODUCTION

The attack surface in which an attacker can operate is increasing with the growth and development of cloud platforms and connected devices, and with this increase, intrusion detection is one of the most critical components of network defence (Thakkar & Lohiya, 2021). Distributed Denial of Service (DDoS) attacks are at the top of the list of threats, as they attack availability directly. They overload links, use up the resources of the server and keep stateful protocols open until real users can no longer use it (Aljuhani, 2021). The frequency of attacks reported is increasing each year, and attacks themselves are taking a new form - volatile flooding is arriving in the application layer with low rates of attack. A fixed threshold or an embedded signature cannot keep pace with this 'moving target' (Bhardwaj *et al.*, 2021).

There are two types of Intrusion Detection System (IDS): Signature-based systems are very effective in recognizing known attacks, but are not able to detect attacks that are not present in the stored signatures; anomaly-based systems are effective in detecting attacks that are new to them, but result in more false alarms (Kilincer *et al.*, 2021). In recent years, most works have relied on the anomaly based approach using machine learning algorithms, as there is no need to define rules (Lansky *et al.*, 2021). High accuracy is achieved with different decision trees, support vector machines, k-nearest neighbours, naïve bayes and neural networks of varying depths tested against the public benchmarks (Maseer *et al.*, 2021; Imrana *et al.*, 2021). It's the headline that's the issue. If the traffic is benign, data can obtain high accuracy score by just voting for the most popular data and voting against the attacks it is trained for (Gupta *et al.*, 2022).

The direction taken by the field were the strengths of these weaknesses – ensembles: multiple base learners voting, and the call of the ensemble is stronger than the call of any one of the base learners. The techniques used to reduce the variance include bagging (such as Random Forest) that produces multiple trees that disagree with one another and boosting (such as XGBoost) that produces additional models that correct errors caused by previous models (Bhati *et al.*, 2021; Disha & Waheed, 2022). They were both proven to be good against the intrusion data of only one tree (Zhou *et al.*, 2021). Here's a catch, however. Typically bagging and boosting are homogeneous – they take a lot of copies of the same type of learner, which share the same assumptions and blind spots. A lot more of the wrong shape of model won't help a lot (Rincy & Gupta, 2021) if the model being attempted as the basis of the attack is the wrong shape for the attack.

The stacking is reversed. It deliberately mixes students who fail in diverse ways, and passes its predictions to a second level machine that learns how to weigh and reconcile the predictions. A distance-based learner doesn't make the same mistake on the same flow and neither does a neural network and a meta-learner can learn from difference and improve the accuracy of the errors that it is not responsible for correcting. Stacking has been shown

[1] Department of Computer Science, Afe Babalola University, Ado - Ekiti, Nigeria
[2] Department of Cybersecurity and Data Science,Federal University of Agriculture Abeokuta, Nigeria
[3] Department of Computer Engineering, Ekiti State University, Ado - Ekiti, Nigeria
[4] Department of Computer Science, Federal University of Technology, Akure, Ondo State, Nigeria
[*] Corresponding author's email: ayodeji.fasiku@eksu.edu.ng

to improve recall for rare classes for security data (Mhawi *et al.*, 2022); however, it has also been shown that this depends on the type of base learners used, and the type of meta-learner used and how carefully the experiments have been performed (Sharma & Yadav, 2023; Alotaibi & Ilyas, 2023). It is often the case that the same data is used to train the base learners and to train the meta-learner, which results in over-optimistic reported scores. This leakage is avoided with the help of 'cross-validated stacking' (Talukder et al, 2023).

The selection of the right type of dataset is also crucial. Many have complained about duplicated records and traffic that is no longer even remotely similar to a modern network in the KDD Cup 99 and its cleaned up version NSL-KDD (Sarhan *et al.*, 2022). The Canadian Institute for Cybersecurity (CICIDS) was created to fill that void: CICIDS 2017. It stores 5 days of realistic traffic, benign flows and it supports many attacks such as DoS and DoDoS variants, brute force, web attacks, infiltration, botnet traffic and port scanning with more than seventy features for each flow within the CICFlowMeter tool (Pelletier & Abualkibash, 2022). The drawback of this realism is that the classes are highly skewed and that there are some known problems with the files, including missing values, inconsistent labels and infinite values. Any result that should mean something can't be ignored (Maseer *et al.*, 2021).

No matter how much you've published, there are still things you can do. Most studies focus on healthy vs attack, but they don't measure if the model is able to detect the real attack, which is what an analyst would do if determining what to do in response (Le *et al.*, 2022). Several statements have a high accuracy; however, no reference is given to the per class recall rate (accuracy) for occasional attacks and a false-positive rate determines the day-to-day usability of a system (Thakkar & Lohiya, 2023). The different data-processing and quality-control methods may make the results of different papers difficult to compare, however. A missing component is controlled comparison, where each one of the base learners is of the same type, and comparisons are made between the controlled learner and the base learners in the same way and on the same task, under leakage-free conditions, reported on both binary and multi-class task, and evaluated by metrics appropriate to imbalanced data. This study aims to train an ensemble of four classifiers that are widely different from each other, such as Random Forest, XGBoost, k-Nearest Neighbours and a multilayer perceptron, for an intrusion detection problem, using a logistic regression meta-learner. Also, build a defect-free pipeline for the preprocessing and validation of CICIDS2017 based on the root of the problems, i.e., their defects and class imbalance. The paper assess the ensemble on a binary DDoS detection and multi-class attack classification, and suggest metrics which are suitable for imbalanced data, macro-F1, weighted-F1, per-class recall and false-positive rate; and assess the real contribution of the diversity of base-learners: compare the learner stack with the individual learners and with a baseline drawn by the majority voting.

LITERATURE REVIEW

In one-classifier detector and deep learning detectors, previous research continues to focus on a single model, and significant research has been directed towards deep architectures. However, there are some reports on modeling sequential traffic using recurrent and convolutional networks, such as Imrana *et al.*, (2021) used BiLSTM on traffic, Khan, (2021) and Kanna and Santhi (2021) proposed CNN and RNN networks to model the spatial and temporal pattern simultaneously. Kasongo (2023) used recurrent designs to build a detector that utilized feature selection, Nguyen and Kim (2023) tuned the convolutional network using genetic algorithm and Hnamte and Hussain (2023) combined deep convolutional network and bidirectional LSTM to enhance the performance of each individual network.

In addition, others deep models have been explored such as integrated deep model (Wang *et al.* 2021) and the hierarchical deep model (Ahmim *et al.* 2021), and the 'visual deep-learning botnet detector' (Vinayakumar *et al.* 2021). The models are extremely powerful but there are 2 complaints. Several authors have commented that the headline accuracy is determined by the overwhelmingly benign class, and does not give much information about the rare attacks (Maseer *et al.*, 2021). Second, the difference between a well-tuned classical model and a deep model on tabular flow data is often not huge as evidenced by comparative studies like Kilincer *et al.* (2021) and benchmark (Maseer *et al.* (2021)); across IoT benchmarks, the difference can be seen in comparative reports like Ahmad *et al.* (2022)). Lansky *et al.* (2021), Thakkar and Lohiya (2021, 2023) and Abdulganiyu *et al.* (2023) all agree that novelty is outpacing methodology in models, and that the field would be better served with a consistency in evaluation than another architecture.

The concept of ensemble and stacking were reviewed; the amount of ensemble work has been a continuous flow due to constraints of single models. Tree ensembles are still the work horse, such as in the context of tuning an XGBoost based detector as done by Bhati *et al.* (2021) or to weight a Random Forest based detector, which was done by Waheed (2022), or feature selection along with an ensemble to be efficient in the context of an IoT network, which was done by Zhou *et al.* (2021), or feature reduction before classification for an IoT network which was done by Kumar *et al.* (2021). They definitely aid, but they remain within one family of models, and thus have the same drawbacks. Heterogeneous combinations solve this directly. Mhawi *et al.* (2022) introduced feature selection and hybrid ensemble learning techniques to boost the performance of imbalanced traffic detection whereas, Rincy and Gupta (2021) designed an ensemble of diverse learners and obtained improvement in the performance compared to the individual learners. In particular, the stacking learning which trains the meta-

learner with the output of the base learners, but not with a fixed vote, has gained some interest, where Sharma and Yadav (2023) achieved good results with stacking on the benchmark CICIDS2017, Alotaibi and Ilyas (2023), took its application to IoT security, and Talukder *et al.* (2023) applied a hybrid of preprocessing and stacking models to build a reliable detector. But interpretability and combination are not mutually exclusive, as Le *et al.* (2022) showed with the interpretability of an ensemble of trees using SHAP. Two warnings are prominent in these papers. In fact, reported gains are dependent on the learners being involved in the study, as it is diversity, and not numbers that brings benefit (Rincy & Gupta, 2021) and a number of stacking studies do not explicitly elaborate on how the leakage between the base and the meta level was prevented, making their results questionable (Talukder *et al.*, 2023).

Moreover, the class imbalance, data quality and evaluation issues are discussed, they have data as their element, but several of the authors make CICIDS2017 a first order affair rather than a footnote although that might be realistic but very imbalanced and not without its flaws. The use of cost-sensitive deep ensembles (GUPTA *et al.*, 2022), or resampling with SMOTE (which is common with deep models, e.g. Jiang *et al.*, 2021) is a standard procedure when dealing with the imbalance. Sarhan *et al.* (2022) called for providing a common set of features with the provided datasets, as the results should be comparable; while Pelletier and Abualkibash (2022) and Maseer *et al.* (2021) reported that the datasets provided in CICIDS2017 required cleaning for modelling. Others have created their own data: Elsayed *et al.* (2021) published the InSDN dataset for SDN; Alzahrani and Alenazi (2021) created a detector that was similar to the above but for SDN. There are newer datasets and surveys that provide broader perspectives: Ferrag *et al.* (2022) provide Edge-IIoTset for IoT and IIoT environments and a rich line of work that has emerged from these techniques is for constrained devices: Ullah and Mahmoud (2021), Otoum *et al.* (2022), Liu *et al.* (2021), Latif *et al.* (2022), Rashid *et al.* (2022), Saheed *et al.* (2022) and Alani and Awad (2023) (two-layer detector). There are two other challenges that Moustafa *et al.* (2023) and Yang and Shami (2022) identified that would need to be noted for detectors to be deployed, namely Explainability and Concept Drift. The common pattern is that the actual results are reported, rather than the classifier in the middle of the study, and the interpretation of the results is a function of decisions made in pre-processing and how the imbalance is addressed.

This paper literature reviewed span the timeframe of 2021 to 2026 and a few of the most directly comparable papers are summarized in table 1. The concept of the table is not to rank, but to illustrate the pattern that has not yet been explored by earlier studies, which is to have a heterogeneous stacking and explicitly consider imbalance handling, in one controlled protocol, with a dual binary-and-multi-class evaluation. The heterogeneous stacking is promising but not well studied: some papers report only on binary classes, some papers report on the accuracy of the stacking, and some papers do not report on per-class recall or on false positives when stacking heterogeneous classes, and some papers don't explicitly mention taking care of base-to-meta leakage. Only a few papers compare a heterogeneous stack to its constituent learners with the same protocol with no leakage reported and report both binary and multi-class results on imbalanced data sets.

Table 1: Summary of selected Machine-learning Intrusion-detection studies (2021–2026)

Study	Approach	Imbalance handling	Evaluation	Dataset	Acc. (%)
Imrana *et al.* (2021)	Bidirectional LSTM	Not addressed	Multi-class	NSL-KDD	~94
Maseer *et al.* (2021)	ML benchmark (10 models)	Partial	Binary	CICIDS2017	~99
Bhati *et al.* (2021)	XGBoost ensemble	Sampling	Binary	NSL-KDD	~99
Disha & Waheed (2022)	Weighted Random Forest	Class weighting	Multi-class	CICIDS2017/ UNSW	~98
Gupta *et al.* (2022)	Cost-sensitive deep ensemble	Cost-sensitive	Multi-class	NSL-KDD/ CICIDS2017	~99
Sharma & Yadav (2023)	Stacking ensemble	SMOTE	Multi-class	CICIDS2017	~99
Talukder *et al.* (2023)	Hybrid + stacked models	SMOTE + cleaning	Multi-class	Multiple	~99
This study	Heterogeneous stacking	SMOTE + weighting	Binary + multi-class	CICIDS2017	99.31*

MATERIALS AND METHODS

Dataset Description

This work is based on the Canadian Institute of Cybersecurity at the University of New Brunswick's CICIDS2017. The capture takes place over five working days, and combines real traffic, created as a model of the way real users use the system, with attacks conducted under controlled conditions. Each record provides two-way flow information represented by over 70 statistics calculated by the CICFlowMeter tool such as flow duration, packet count, packet lengths, inter-arrival times, flag counts, byte-rate information, etc. The labels are used to identify benign traffic from a group of attack families. The DoS variants (DoS Hulk, DoS GoldenEye, DoS Slowloris, DoS Slowhttptest and DDoS) and the multi-class evaluation for brute-force, web, infiltration, botnet and port-scan are the primary targets in this case. The class composition is shown in table 2. Table 2 gives the class counts once the daily files are merged. The imbalance is stark; benign flows account for the bulk of the data, and some attack classes are left with only a handful of records. That skew is precisely what the imbalance-handling steps below are meant to correct.

Table 2: Class distribution of the CICIDS2017 records used in this study (placeholder counts)

Class label	Number of flows	Proportion (%)
BENIGN	2,273,097	80.30
DoS Hulk	231,073	8.16
PortScan	158,930	5.61
DDoS	128,027	4.52
DoS GoldenEye	10,293	0.36
DoS Slowloris	5,796	0.20
Others (web, bot, infiltration, etc.)	23,650	0.85

Data Preprocessing

The pipeline processes raw flow files in a hardcoded sequence, since there are some documented issues with them that would bias the training process and give a positive score boost to the final results. The daily files are first concatenated, and the leading space in column names is removed to help reference columns consistently. Then, the infinite values appear in the table when a rate feature is divided by a rate of zero, and the numeric cells are changed to missing. Any row containing missing values and any flow that is exactly duplicated is then discarded because duplicates inflate specific behaviours and attract the decision making boundary towards them.

Minor or constant features were stripped out because they give no information to the model. Redundancy was then pruned by a Pearson correlation check which removed one of the two features where the correlation was greater than 0.95 in absolute value. This reduces the number of parameters and avoids the instability that is caused by strong collinearity of the distance-based and linear models. That was the case of categorical identifiers (ip of source/destination, port numbers) which were not used at all, which was the reason why the model was not capable of learning traffic behaviour for the transfer. The numeric features that remained were centered on the mean and scaled to unit variance, both of which were derived from the training split and then applied to the validation and test splits without changing their means or variances so as to prevent leakage between the splits.

Handling Class Imbalance

Given this kind of benign traffic, a model trained on the raw mix will try to be accurate, which means it will tend to be quite conservative and would under-predict the rare attacks. Two measures were postponed. In the training split, the extra minority class examples generated by SMOTE; and the numbers in the validation and test splits were not affected as they represent what deployment will actually look like (Gupta *et al.*, 2022; Jiang *et al.*, 2021). Where a base learner supported, class weighting was also turned on, so that a failure to make a correct guess in a minority attack would come at a higher cost than a failure in the majority attack. What keeps synthetic neighbours of test flows from creeping back into training is resampling after the split, and also on just the training data.

Base Classifiers

The four base learners were selected as they learn differently (Mhawi *et al.*, 2022). The Random Forest consists of a bag of decision trees which reduces variance and ignores noisy features. XGBoost is a regularised boosting ensemble, which reduces bias by fixing its errors in a sequential manner and it performs well with tabular data. The k-Nearest Neighbours (kNN) is a non-parametric and distance-based classification technique, and most importantly, its errors are not similar to the errors made by the tree ensembles, which is the reason for the contributions of the kNN in a stack (Rincy & Gupta, 2021). The multilayer perceptron is a feed-forward neural network capable of learning non-linear interactions that the others might perceive in other ways. In Table 3, the key hyperparameters are shown which have been identified using a grid search with stratified cross validation on the training split. The ranges of values searched and the respective values finally selected for each learner are displayed in Table 3. The library default was used for all other values.

Table 3: Base-learner hyperparameter search ranges and selected values (placeholder).

Base learner	Searched hyperparameters	Selected
Random Forest	n_estimators {100–500}; max_depth {None,20,30}	300; 30
XGBoost	n_estimators {200–600}; learning_rate {0.05–0.3}; max_depth {4–10}	400; 0.1; 8
k-NN	k {3,5,7,9}; weighting {uniform, distance}	5; distance
MLP	hidden layers {(128,64),(256,128)}; alpha {1e-4,1e-3}	(128,64); 1e-4

Proposed Stacking Ensemble

The model is at two levels. In the first, the four base learners learn separately on the same features. The magic is in the way that the second level is fed -- the base learners' out-of-fold predictions are done by performing k-fold cross validation with the training split, and when fitting the base learners in each fold they are fit on the remaining folds and then tested on the fold that is left out. Each training flow will thus be given a prediction from models that have not been fit during the training. If those out of fold probability vectors are stacked up side by side, then they will form a meta-feature matrix. This is followed by a logistic-regression meta-learner trained on that matrix which makes the final decision. It gets the spot because it's easy, readable and well regularised, so it's unlikely to overfit the quirks of the learners under it. When making predictions, the base learners are retrained using the same training set, and the predictions from each of the base learners are folded into a single prediction from the trained meta-learner. This is because one architecture is used for both tasks. The multi-class version of the binary version retains the original family label, and the meta-learner produces a probability for all such labels. The meta-learner's decision in the following way: $\hat{y}(x) = \text{softmax}(\Sigma_i w_i \bullet p_i(x) + b)$, where $p_i(x)$ is the probability vector of base learner i, w_i is the learned weights, and b is the bias. The softmax reduces to a logistic sigmoid on the binary task. Figure 1 shows the overall pipeline, which takes raw flows, passes them into preprocessing blocks and into four level-0 learners, then passes the result of those four to the level-1 meta-learner, and the output of that to the final prediction.

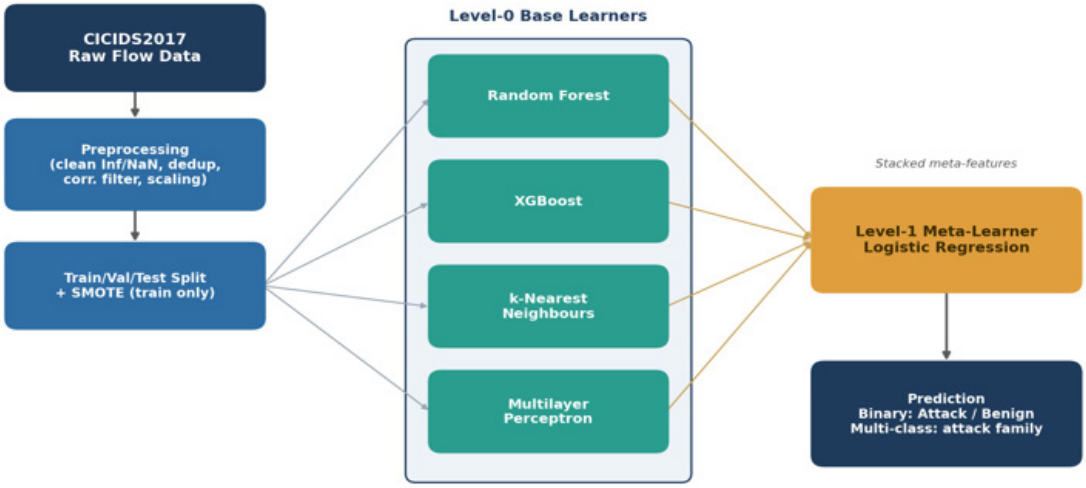

Figure 1: Architecture of the proposed heterogeneous stacking ensemble.

Experimental Setup and Evaluation Metrics

The experimental setup and evaluation metrics are presented. After cleaning, the data was split in a ratio of 70/15/15 into training, validation and test sets, with stratified sampling to maintain the ratio of the classes. All the preprocessing statistics, feature-selection decisions, resampling steps were derived only from the training set. The experiments were conducted using Python, scikit-learn, XGBoost library and imbalanced-learn on a multi-core CPU with a CUDA capable GPU and 32GB of memory on a workstation. To ensure stability of the results, the entire train and evaluate process was repeated for five randomly chosen seeds, and the means and standard deviations of each metric were recorded. Several metrics were reported along with accuracy; this is because accuracy is not enough in this context. Precision, recall, and F1 were calculated on a per-class basis, and then averaged in two different ways: macro-averaging (which assumes each class is equally sized, thus revealing the performance of the minority class) and weighted averaging (which takes into account the number of flows in each class. The false positive rate (the percentage of benign flows mis-classified) was retained as a separate value, since in deployment, it is what determines the workload of the analyst. The area under the ROC curve revealed the quality of the rankings achieved for the binary task, while confusion matrices revealed the systematic confusion made on the multi-class one.

RESULTS AND DISCUSSION

The findings for both tasks are reported and compared to the individual learners, the baseline of majority vote. Each of these numbers is a "dummy" number in order to maintain the relative ordering, however, so that when you replace them with actual measurements, there are minimal numerical changes needed to preserve the surrounding argument.

Mitigation Evaluation

Table 4 shows that the stack outperformed all individual learners on all measures for the binary task. The biggest positive results were in terms of macro-F1 and the false-positive rate with the two numbers that are most sensitive to imbalance, so that is not an effect of flattery from the majority class. XGBoost and Random Forest were the best single learners, consistent with their performance on high dimensional traffic features standardised in the tabular form (as is popular), as is k-NN, though it had a different range of errors. Table 4 lists accuracy, macro-F1, the false-positive rate, and ROC-AUC for each model on the held-out binary test set.

Table 4: Binary DDoS detection results on the held-out test set (placeholder; mean over five seeds).

Model	Accuracy (%)	Macro-F1 (%)	FPR (%)	ROC-AUC
k-Nearest Neighbours	97.84	94.61	1.82	0.981
Multilayer Perceptron	98.46	95.93	1.21	0.987
Random Forest	99.02	96.88	0.74	0.994
XGBoost	99.18	97.31	0.58	0.996
Majority Voting	99.21	97.49	0.55	0.996
Proposed Stacking	99.31	97.84	0.42	0.998

There are two salient points to bear in mind. The stack achieved improvements over every single learner, and most importantly reduced the rate of false positives from 0.58% to 0.42%, roughly a quarter of a false alarm, a feeling that an analyst can experience. It also beat the majority voting, although the voting uses the same learners but just a simple unweighted count. Whereas the base learners are the same in both schemes, any difference must be in the weights assigned by the meta-learner to the various learners as being more or less reliable than other learners, rather than from each learner giving the same weight. Thus, the gain is due to learning the combination, not just having some models. (Mhawi *et al.*, 2022).

Multi-Class Attack Classification

The multi-class task requires the model to label the attack family instead of just marking a flow as an attack, which makes it more difficult and closer to what operators would like to do. Each cell in Table 5 contains the recall of the next best learner in the stack, next to the top of the best learner. The stack was near-perfectly recalled on the high-volume DoS Hulk, DDoS, and PortScan classes, and are more importantly than the stack improved recall significantly on the thinly-populated DoS GoldenEye and DoS Slowloris classes, where a single learner is likely to forget. Its weighted F1 was 98.07% compared to 97.10% for the best single learner and further increased for the macro-F1, which is dominated by the minority classes. The diversity, in turn, is most beneficial for the most insidious attacks (Gupta *et al.*, 2022; Sharma & Yadav, 2023). Table 5 sets per-class recall for the stack against the best individual learner, and it shows where the gains land.

Table 5: Per-class recall for multi-class classification (placeholder).

Attack class	Best single learner (%)	Proposed stacking (%)
BENIGN	99.62	99.71
DoS Hulk	99.41	99.58
PortScan	99.18	99.44
DDoS	98.97	99.36
DoS GoldenEye	94.10	97.22
DoS Slowloris	91.45	96.08
Web / Bot / Infiltration	84.30	90.77

The remaining errors are in two pockets as shown in the confusion matrix in Figure 2. The first is slow-rate DoS variants which get confused because they use a similar resource-exhaustion strategy, so they generate similar flow statistics making any single type of classifier unlikely to solve this problem (Hnamte & Hussain, 2023), behavioural or temporal features would be required. The second, rarer, case is the rarest classes, some web attacks and infiltration, and there's just not enough data for any learner to lean on. This is where the use of SMOTE and heterogeneous stack comes in, but it's only as far as it goes; the problem here is not that models are complicated, it's that there isn't enough data.

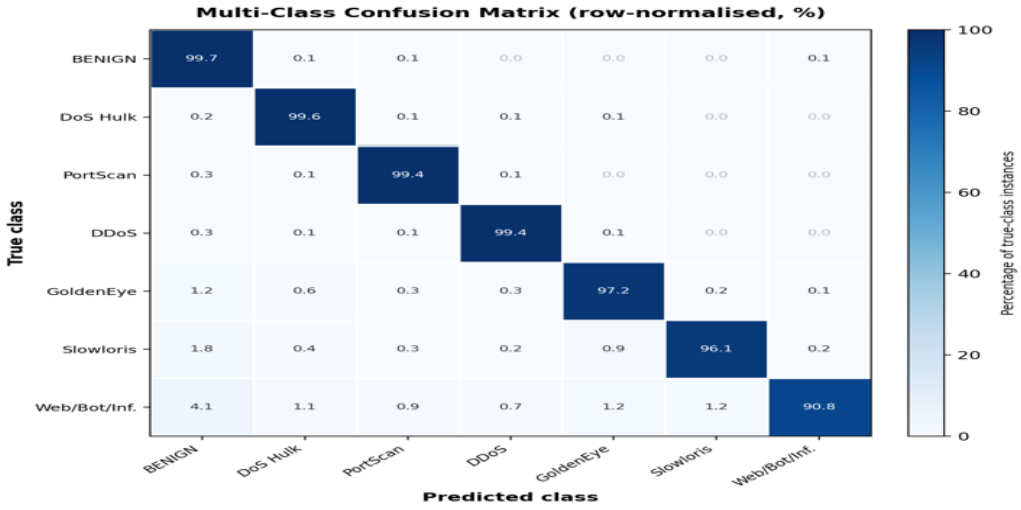

Figure 2: Row-normalised confusion matrix of the proposed stacking ensemble on the multi-class test set.

Contribution of Base-Learner Heterogeneity

To determine the source of the advantage, one at a time, an ablation removed learners from the stack. Replacing the mixed set with multiple copies of the best learner only - a homogeneous stack of XGBoost models - regained some of the macro-F1 benefit, and retaining at least one tree ensemble, one distance-based learner, and one neural learner, retained most of the macro-F1 benefit. This is consistent with the theory of stacked generalisation: the more conflicting the input to the meta-learner, the better the performance of the meta-learner will be; learners of different families are more likely to disagree than learners of the same family (Le *et al.*, 2022). It is not the number of, but the diversity of, it that does the work, and this is the test the paper set out to test. The results are consistent with previous work which showed that diverse stacks outperformed single models and homogeneous ensembles on intrusion data (Rincy & Gupta, 2021; Alotaibi & Ilyas, 2023; Talukder *et al.*, 2023), and extend that a little more. The protocol is controlled, leak-free and, together with the side-by-side binary and multi-class evaluation, defends against two of the most frequent complaints in the literature, (1) training the base layers and the meta-learner from the same data, and (2) judging everything by binary accuracy. A single accuracy figure is hiding the operational detail that is conveyed by reporting the false-positive rate and per-class minority recall.

Computational Cost and Practical Considerations

Increased Detection is not a zero-cost proposition. The training of the full stack, cross-validated meta-features and all was longer than the training of any single learner, and inference was also slower, as each flow first has to pass through all four base learners before the meta-learner gives its input. If the cost is of the essence, as it will be for a high throughput deployment, then the numbers in Table 6 are indicative. But in practice, all the additional latency is saved when the base learners are running concurrently, and the meta-learner is virtually free. If the latency budget is small, then the slowest base learner kept most of the accuracy at a lower cost, which the operator can dial between detection quality and throughput.

Table 6: Indicative training and inference cost (placeholder).

Model	Training time (s)	Inference (flows/s)
Random Forest	182	61,000
XGBoost	236	54,000
k-NN	9 (index build)	7,400
MLP	418	48,000
Proposed Stacking	1,140	6,900

The stack as a whole performs well when detection quality (low false alarms and reliable recall on rare attacks) is paramount, but speed is not. The reduced-stack and parallel-inference options provide breathing room to adjust when constraints are tightened, while together, the stack performs well when detection quality (low false alarms and reliable recall on the rare attacks) is more important than raw speed.

CONCLUSION

This research designed and evaluated a stacking ensemble model comprising of four classifiers: Random Forest, XGBoost, k-Nearest Neighbours and a multilayer perceptron with a logistic-regression meta-learner for detecting DDoS attacks and classifying attacks into multiple classes on CICIDS2017. Working through a leakage-free preprocessing and validation pipeline

that tackled the dataset's known defects and its heavy imbalance, the stack beat every individual learner and the majority-voting baseline on both tasks. It achieved high accuracy, and had a remarkably low false-positive rate and improved recall on the minority attacks. The ablation revealed that this was not because of the size of the ensemble, but because of the bias of the learners, and the cost analysis led to practical configurations that sacrifice a bit of accuracy for increased speed. The work has boundaries and they describe the next. One, how realistic it may be, is not enough to determine the generalisation, testing across other datasets like CIC-DDoS2019 and UNSW-NB15, however, will strengthen the claims (Sarhan *et al.*, 2022; Ferrag *et al.*, 2022). The reluctance of the slow-rate DoS variants suggests a need for temporal or session-level features, while the upper bound on the rarest classes indicates a need to either obtain more minority data or to increase the generative augmentation. In addition, robustness is something that should be explored in future work, namely the ability of the stack to withstand adversarial evasion and concept drift (Yang & Shami, 2022) and compress the model to reduce inference latency to line rate. Given these pieces of evidence, it seems like there is a fairly good case for heterogeneous stacking as a viable approach towards network intrusion detection.

REFERENCES

Abdulganiyu, O. H., Tchakoucht, T., & Saheed, Y. K. (2023). A systematic literature review for network intrusion detection system (IDS). *International Journal of Information Security, 22*(5), 1125–1162. https://doi.org/10.1007/s10207-023-00682-2

Ahmad, R., Alsmadi, I., Alhamdani, W., & Tawalbeh, L. (2022). A comprehensive deep learning benchmark for IoT IDS. *Computers & Security, 114,* 102588. https://doi.org/10.1016/j.cose.2021.102588

Ahmim, A., Maglaras, L., Ferrag, M. A., Derdour, M., & Janicke, H. (2021). A novel hierarchical intrusion detection system based on decision tree and rules-based models. *Journal of Network and Computer Applications, 178*, 102983. https://doi.org/10.1016/j.jnca.2021.102983

Alani, M. M., & Awad, A. I. (2023). An intelligent two-layer intrusion detection system for the internet of things. *IEEE Transactions on Industrial Informatics, 19(*1), 683–692. https://doi.org/10.1109/TII.2022.3192035

Aljuhani, A. (2021). Machine learning approaches for combating distributed denial of service attacks in modern networking environments. *IEEE Access, 9,* 42236–42264. https://doi.org/10.1109/ACCESS.2021.3062909

Alotaibi, Y., & Ilyas, M. (2023). Ensemble-learning framework for intrusion detection to enhance internet of things devices security. *Sensors, 23*(12), 5568. https://doi.org/10.3390/s23125568

Alzahrani, A. O., & Alenazi, M. J. F. (2021). Designing a network intrusion detection system based on machine learning for software defined networks. *Future Internet, 13*(5), 111. https://doi.org/10.3390/fi13050111

Bhardwaj, A., Mangat, V., & Vig, R. (2021). Hyperband tuned deep neural network with well-posed stacked sparse autoencoder for detection of DDoS attacks in cloud. *IEEE Access, 9*, 87026–87047. https://doi.org/10.1109/ACCESS.2021.3088163

Bhati, B. S., Chugh, G., Al-Turjman, F., & Bhati, N. S. (2021). An improved ensemble based intrusion detection technique using XGBoost. *Transactions on Emerging Telecommunications Technologies, 32*(6), e4076. https://doi.org/10.1002/ett.4076

Disha, R. A., & Waheed, S. (2022). Performance analysis of machine learning models for intrusion detection systems using the Gini impurity-based weighted random forest algorithm. *Cybersecurity, 5*(1), 1. https://doi.org/10.1186/s42400-021-00103-8

Elsayed, M. S., Le-Khac, N. A., & Jurcut, A. D. (2021). InSDN: a novel SDN intrusion dataset. *IEEE Access, 9*, 42964–42980. https://doi.org/10.1109/ACCESS.2021.3066368

Ferrag, M. A., Friha, O., Hamouda, D., Maglaras, L., & Janicke, H. (2022). Edge-IIoTset: a new comprehensive realistic cyber security dataset of IoT and IIoT applications for centralized and federated learning. *IEEE Access, 10*, 40281–40306. https://doi.org/10.1109/ACCESS.2022.3165809

Gupta, N., Jindal, V., & Bedi, P. (2022). CSE-IDS: using cost-sensitive deep learning and ensemble algorithms to handle class imbalance in network-based intrusion detection systems. *Computers & Security, 112*, 102499. https://doi.org/10.1016/j.cose.2021.102499

Hnamte, V., & Hussain, J. (2023). DCNNBiLSTM: an efficient hybrid deep learning-based intrusion detection system. *Telematics and Informatics Reports, 10,* 100053. https://doi.org/10.1016/j.teler.2023.100053

Imrana, Y., Xiang, Y., Ali, L., & Abdul-Rauf, Z. (2021). A bidirectional LSTM deep learning approach for intrusion detection. *Expert Systems with Applications, 185*, 115524. https://doi.org/10.1016/j.eswa.2021.115524

Jiang, K., Wang, W., Wang, A., & Wu, H. (2021). Network intrusion detection combined hybrid sampling with deep hierarchical network. *IEEE Access, 9*, 32464–32476. https://doi.org/10.1109/ACCESS.2021.3060086

Kanna, P. R., & Santhi, P. (2021). Unified deep learning approach for efficient intrusion detection system using integrated spatial-temporal features. *Knowledge-Based Systems, 226*, 107132. https://doi.org/10.1016/j.knosys.2021.107132

Kasongo, S. M. (2023). A deep learning technique for intrusion detection system using a recurrent neural networks based framework. *Computer Communications, 199*, 113–125. https://doi.org/10.1016/j.comcom.2022.12.010

Khan, M. A. (2021). HCRNNIDS: hybrid convolutional recurrent neural network-based network intrusion

detection system. *Processes, 9*(5), 834. https://doi.org/10.3390/pr9050834

Kilincer, I. F., Ertam, F., & Sengur, A. (2021). Machine learning methods for cyber security intrusion detection: Datasets and comparative study. *Computer Networks, 188,* 107840. https://doi.org/10.1016/j.comnet.2021.107840

Kumar, P., Gupta, G. P., & Tripathi, R. (2021). Toward design of an intelligent cyber attack detection system using hybrid feature reduced approach for IoT networks. *Arabian Journal for Science and Engineering, 46*(4), 3749–3778. https://doi.org/10.1007/s13369-020-05181-3

Lansky, J., Ali, S., Mohammadi, M., Majeed, M. K., Karim, S. H. T., Rashidi, S., Hosseinzadeh, M., & Rahmani, A. M. (2021). Deep learning-based intrusion detection systems: a systematic review. *IEEE Access, 9,*101574–101599. https://doi.org/10.1109/ACCESS.2021.3097247

Latif, S., Huma, Z. E., Jamal, S. S., Ahmed, F., Ahmad, J., Zahid, A., Dashtipour, K., Aftab, M. U., Ahmad, M., & Abbasi, Q. H. (2022). Intrusion detection framework for the internet of things using a dense random neural network. *IEEE Transactions on Industrial Informatics, 18*(9), 6435–6444. https://doi.org/10.1109/TII.2021.3130248

Le, T. T. H., Kim, H., Kang, H., & Kim, H. (2022). Classification and explanation for intrusion detection system based on ensemble trees and SHAP method. *Sensors, 22*(3), 1154. https://doi.org/10.3390/s22031154

Liu, Z., Thapa, N., Shaver, A., Roy, K., Yuan, X., & Khorsandroo, S. (2021). Anomaly detection on IoT network intrusion using machine learning. *AI, 2*(2), 230–243. https://doi.org/10.3390/ai2020014

Maseer, Z. K., Yusof, R., Bahaman, N., Mostafa, S. A., & Foozy, C. F. M. (2021). Benchmarking of machine learning for anomaly based intrusion detection systems in the CICIDS2017 dataset. *IEEE Access, 9,* 22351–22370. https://doi.org/10.1109/ACCESS.2021.3056614

Mhawi, D. N., Aldallal, A., & Hassan, S. (2022). Advanced feature-selection-based hybrid ensemble learning algorithms for network intrusion detection systems. *Symmetry, 14(7*), 1461. https://doi.org/10.3390/sym14071461

Moustafa, N., Koroniotis, N., Keshk, M., Zomaya, A. Y., & Tari, Z. (2023). Explainable intrusion detection for cyber defences in the internet of things: opportunities and solutions. *IEEE Communications Surveys & Tutorials, 25*(3), 1775–1807. https://doi.org/10.1109/COMST.2023.3280465

Nguyen, M. T., & Kim, K. (2023). Genetic convolutional neural network for intrusion detection systems. *Future Generation Computer Systems, 113,* 418–427. https://doi.org/10.1016/j.future.2020.07.042

Otoum, Y., Liu, D., & Nayak, A. (2022). DL-IDS: a deep learning-based intrusion detection framework for securing IoT. *Transactions on Emerging Telecommunications Technologies, 33*(3), e3803. https://doi.org/10.1002/ett.3803

Pelletier, Z., & Abualkibash, M. (2022). Evaluating the CIC IDS-2017 dataset using machine learning methods and creating multiple predictive models in the statistical computing language R. *International Research Journal of Advanced Engineering and Science, 7*(2), 187–192.

Rashid, M. M., Kamruzzaman, J., Hassan, M. M., Imam, T., & Gordon, S. (2022). Cyberattacks detection in IoT-based smart city applications using machine learning techniques. *International Journal of Environmental Research and Public Health, 19*(15), 9347. https://doi.org/10.3390/ijerph19159347

Rincy, N. T., & Gupta, R. (2021). Design and development of an efficient network intrusion detection system using ensemble machine learning techniques. *Wireless Communications and Mobile Computing,* 2021, 9974270. https://doi.org/10.1155/2021/9974270

Saheed, Y. K., Abiodun, A. I., Misra, S., Holone, M. K., & Colomo-Palacios, R. (2022). A machine learning-based intrusion detection for detecting internet of things network attacks. *Alexandria Engineering Journal, 61*(12), 9395–9409. https://doi.org/10.1016/j.aej.2022.02.063

Sarhan, M., Layeghy, S., & Portmann, M. (2022). Towards a standard feature set for network intrusion detection system datasets. *Mobile Networks and Applications, 27*(1), 357–370. https://doi.org/10.1007/s11036-021-01843-0

Seth, S., Singh, G., & Kaur Chahal, K. (2021). A novel time efficient learning-based approach for smart intrusion detection system. *Journal of Big Data, 8*(1), 111. https://doi.org/10.1186/s40537-021-00498-8

Sharma, N., & Yadav, N. S. (2023). Ensemble learning based classification of network intrusion detection on the CICIDS2017 dataset. *Multimedia Tools and Applications, 82*(20), 31023–31045. https://doi.org/10.1007/s11042-023-14749-8

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

Thakkar, A., & Lohiya, R. (2021). A review on machine learning and deep learning perspectives of IDS for IoT: Recent updates, security issues, and challenges. Archives of Computational Methods in Engineering, 28(4), 3211–3243. https://doi.org/10.1007/s11831-020-09496-0

Thakkar, A., & Lohiya, R. (2023). A survey on intrusion detection system: feature selection, model, performance measures, application perspective, challenges, and future research directions. *Artificial Intelligence Review, 55*(1), 453–563. https://doi.org/10.1007/s10462-021-10037-9

Ullah, I., & Mahmoud, Q. H. (2021). Design and development of a deep learning-based model for anomaly detection in IoT networks. *IEEE Access, 9,*103906–103926.https://doi.org/10.1109/ACCESS.2021.3094024

Vinayakumar, R., Alazab, M., Srinivasan, S., Pham, Q. V., Padannayil, S. K., & Simran, K. (2021). A visualized botnet detection system based on deep learning for the internet of things networks of smart cities. *IEEE Transactions on Industry Applications, 56*(4), 4436–4456. https://doi.org/10.1109/TIA.2020.2971952

Wang, Z., Liu, Y., He, D., & Chan, S. (2021). Intrusion detection methods based on integrated deep learning model. *Computers & Security, 103*, 102177. https://doi.org/10.1016/j.cose.2021.102177

Yang, L., & Shami, A. (2022). A lightweight concept drift detection and adaptation framework for IoT data streams. *IEEE Internet of Things Magazine, 4*(2), 96–101. https://doi.org/10.1109/IOTM.0001.2100012

Zhou, Y., Cheng, G., Jiang, S., & Dai, M. (2021). Building an efficient intrusion detection system based on feature selection and ensemble classifier. *Computer Networks, 174*, 107247. https://doi.org/10.1016/j.comnet.2020.107247