Journal of Information Systems Engineering and Management
2024, 9(3)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

WSE-XIDS: A Weighted Stacking Ensemble of Neural and Gradient-Boosted Learners with SHAP Explanations for Zero-Day-Aware Intrusion Detection

Namrata Nebhnani[1], Dr. Sudhir Agrawal[2]

[1]*Research Scholar,* [2]*Professor and Director General*

[1,2]*Department of Electronics & Communication Engineering, SAGE University, Indore, India*

ARTICLE INFO

Received: 01 Aug 2024

Revised: 20 Sep 2024

Accepted: 29 Sept 2024

ABSTRACT

Zero-day attacks exploit vulnerabilities for which no signature exists, so an intrusion detection system (IDS) must recognise them from behaviour alone. Many machine learning IDS studies report near-perfect accuracy, but they usually test on attacks of the same types seen in training, which says little about unseen attacks. This paper presents WSE-XIDS, a weighted stacking ensemble with eXplainable artificial intelligence (XAI) that combines a class-balanced multi-layer perceptron (MLP), an extreme gradient boosting (XGBoost) model and a light gradient boosting machine (LightGBM) through a class-balanced logistic regression meta-learner, and explains its decisions with SHapley Additive exPlanations (SHAP). The model is evaluated on the NSL-KDD benchmark under two protocols: a stratified 80/20 split of KDDTrain+ (known attacks) and the official KDDTest+ set, which contains 3,750 records of 17 attack types never seen in training (zero-day test). On known attacks WSE-XIDS reached 99.85% accuracy, 98.31% macro F1-score and the highest macro recall (99.42%) of eight models. On the zero-day test it reached 77.83% accuracy and the best macro F1-score (67.07%), detected 35.81% of the novel-attack records as attacks, and raised false alarms on 5.07% of normal traffic. SHAP analysis shows that source bytes, connection count and destination bytes drive most decisions. The large gap between the two protocols, and the low recall on unseen remote-access attacks, show that zero-day detection remains far from solved, and that known-attack accuracy alone overstates IDS readiness.

Keywords: Intrusion detection system, zero-day attack, stacking ensemble, XGBoost, LightGBM, explainable AI, SHAP, NSL-KDD.

1. INTRODUCTION

A zero-day attack exploits a vulnerability that is unknown to the vendor and to the defender, so no patch and no signature exist when it is first used. Measurements on real hosts show that such attacks can stay undetected for months [1], and recent surveys list them among the hardest problems for network defence, especially in industrial and Internet of Things (IoT) networks where patching is slow [2], [3]. Signature-based intrusion detection systems (IDSs) cannot recognise traffic that matches no stored pattern, which is why machine learning (ML) and deep learning (DL) detectors that learn the behaviour of normal and malicious traffic have become the main research direction [4], [5].

Learning-based IDSs have their own weakness. They are trained on a fixed set of attack types and are usually evaluated on a random split of the same data, so the test set contains the same attack types as the training set. Under that protocol many models report accuracies above 99%, yet such numbers do not show how a detector behaves on attacks it has never seen [6]. A second weakness is opacity: an analyst who receives an alert from a deep ensemble cannot easily tell why it was raised, which limits trust in operational settings.

This paper addresses both points with a deliberately simple, CPU-friendly design and an evaluation that separates known from unseen attacks. The contributions are as follows.

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Journal of Information Systems Engineering and Management
2024, 9(3)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

(i) WSE-XIDS, a weighted stacking ensemble of a class-balanced multi-layer perceptron (MLP), extreme gradient boosting (XGBoost) and a light gradient boosting machine (LightGBM), fused by a class-balanced logistic regression meta-learner trained on out-of-fold probabilities;

(ii) a two-protocol evaluation on NSL-KDD that reports known-attack performance on a hold-out split of KDDTrain+ and zero-day performance on KDDTest+, whose 17 novel attack types are absent from training, together with novel-attack detection and false-alarm rates;

(iii) global and class-wise explanations with SHapley Additive exPlanations (SHAP), and an honest comparison with seven single-model baselines and published NSL-KDD results.

Section 2 reviews related work, Section 3 describes the method, Section 4 gives the experimental set-up, Section 5 presents and discusses the results, and Section 6 concludes.

2. LITERATURE REVIEW

2.1 Benchmarks and evaluation practice

The KDD Cup 1999 data, derived from the 1999 DARPA evaluation [7], remain the most cited IDS benchmark, but Tavallaee *et al.* [8] showed that about 78% of its training records and 75% of its test records are duplicates, which biases learners towards frequent records and inflates accuracy. Their refined NSL-KDD set removes duplicates and keeps a separate test set, KDDTest+, that includes attack types absent from training. Newer data sets such as CIC-IDS2017 and UNSW-NB15 capture more modern traffic [9]–[11], but NSL-KDD is still the most widely used reference for comparing detectors, particularly for the unseen-attack setting provided by KDDTest+.

2.2 Learning-based and zero-day detection

Deep models were applied to NSL-KDD early on: Javaid *et al.* [12] used self-taught learning with a sparse autoencoder, Yin *et al.* [13] used a recurrent neural network, and Vinayakumar *et al.* [14] compared deep networks across several benchmarks. Work aimed specifically at zero-day attacks has used autoencoders trained on benign traffic [15], unsupervised anomaly detectors chosen by attack class [16], zero-shot learning protocols that hold out whole attack types [17] and federated deep learning for zero-day botnet detection at the IoT edge [18]. Dahal *et al.* [19] studied truncated and class-weighted MLPs with SHAP on KDD99, and Mohamed *et al.* [20] reviewed ML for zero-day exploit detection, noting limited data and high false-positive rates as recurring problems.

2.3 Explainability in IDSs and research gap

Wang *et al.* [21] were among the first to use SHAP to explain IDS decisions on NSL-KDD, and Mahbooba *et al.* [22] used decision-tree rules to support trust management. Across this literature, three gaps recur: results are often reported only on known-attack splits, minority classes such as remote-to-local and user-to-root attacks receive little attention, and explanation is added to single models rather than to ensembles. WSE-XIDS targets all three at modest computational cost.

3. Proposed Methodology

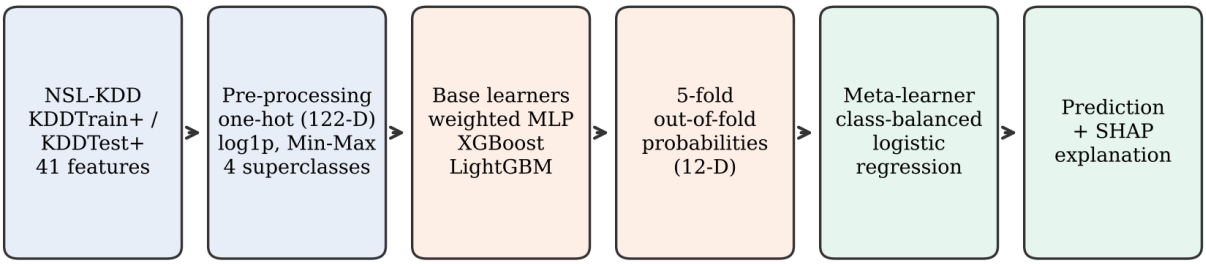


Figure 1. WSE-XIDS pipeline.

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Journal of Information Systems Engineering and Management
2024, 9(3)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Figure 1 summarises the pipeline. Raw NSL-KDD connection records are encoded and scaled, mapped to four traffic classes, and passed to three base learners with different inductive biases. Their class probabilities, computed out-of-fold on the training data, form a 12-dimensional input for a class-balanced logistic regression meta-learner that outputs the final class. SHAP values computed on the tree learner explain which features drive each decision.

3.1 Pre-processing and class mapping

Each record has 41 features. The three categorical features (protocol type, service and flag) are one-hot encoded, giving 122 numeric inputs. Heavy-tailed counters such as duration, source bytes and destination bytes are compressed with $\log(1 + x)$, and every feature is rescaled to [0, 1] using statistics from the training data only:

$$x_j' = \frac{x_j - min_j}{max_j - min_j} \tag{1}$$

Test values outside the training range are clipped to [0, 1]. The attack labels are mapped to four classes: Normal, denial of service (DoS), Probe, and unauthorised access (UA), which merges the rare remote-to-local and user-to-root categories. Novel attack types in KDDTest+ are mapped to the same four classes according to their documented category, so the zero-day test asks whether a model can place an unseen attack in the right family.

3.2 Base learners

Three learners with complementary biases are used. (a) A weighted MLP with two hidden layers of 128 and 64 ReLU units, trained with the Adam optimiser [26] and early stopping; minority classes are randomly oversampled to 15% of the majority class size, a simple alternative to synthetic oversampling [27]. (b) XGBoost [24] with 300 trees of maximum depth 8 and learning rate 0.1, trained with softened class weights

$$w_i = \left(\frac{N}{K \cdot N_{y_i}}\right)^{1/2} \tag{2}$$

where N is the number of training records, $K = 4$ and N_y is the size of class y; the square root keeps rare classes visible without letting a few UA records dominate. (c) LightGBM [25] with 300 trees, 63 leaves, learning rate 0.05 and fully balanced class weights. The MLP captures smooth non-linear interactions, while the two boosted ensembles capture sharp thresholds on counters and flags.

3.3 Stacking meta-learner

Stacked generalisation [23] trains a second-level model on the predictions of the first-level models. To avoid leakage, the training set is split into five stratified folds; for each fold, the three base learners are trained on the other four folds and predict the held-out fold, giving an out-of-fold vector for every record

$$z_i = \left(p_{MLP}(x_i), p_{XGB}(x_i), p_{LGBM}(x_i)\right) \in \mathbb{R}^{12} \tag{3}$$

A multinomial logistic regression with balanced class weights is fitted on these vectors and predicts

$$P(y = c \mid z_i) = \frac{exp(\beta_c{}^{\top} z_i + b_c)}{\sum_{k=1}^{4} exp(\beta_k{}^{\top} z_i + b_k)} \tag{4}$$

After the meta-learner is fitted, the three base learners are retrained on the full training set for inference. The absolute meta-learner coefficients indicate how much each base learner contributes.

3.4 SHAP explanations

SHAP [28] attributes a prediction $f(x)$ to the features through Shapley values, so that the explanation is additive:

$$f(x) = \phi_0 + \sum_{j=1}^{M} \phi_j(x) \tag{5}$$

where ϕ_0 is the expected model output and ϕ_j is the contribution of feature j. TreeSHAP [29] computes exact values for tree ensembles in polynomial time, so it is applied to the XGBoost learner, which receives a large share of the meta-learner weight. Global importance is the mean absolute SHAP value per feature and class over 3,000 test records. Local explainers such as LIME [30] could be added for single alerts but are not used here.

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Journal of Information Systems Engineering and Management
2024, 9(3)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Algorithm 1 lists the training and inference steps. Scaling statistics, fold assignments and class weights are all computed from training data only, so no information from either test set enters training.

Algorithm 1. WSE-XIDS training and inference

Input: training records $\{(\mathbf{x}_i, y_i)\}$, folds $F = 5$; **Output:** class $\hat{y}$ and SHAP explanation

1: one-hot encode protocol, service and flag; apply $\log(1 + x)$ to counters; fit Min-Max scaler on training data (Eq. 1)

2: for each fold $f = 1 \ldots F$: train weighted MLP, XGBoost (weights from Eq. 2) and LightGBM on the other folds; store their probabilities for fold f (Eq. 3)

3: fit class-balanced logistic regression on the out-of-fold vectors (Eq. 4); retrain the three base learners on all training data

4: **Inference:** scale $\mathbf{x}$, compute $\mathbf{z}$ from the three learners, return $\hat{y} = \arg\max_c P(y = c \mid \mathbf{z})$

5: **Explanation:** compute TreeSHAP values of the XGBoost learner for $\mathbf{x}$ (Eq. 5) and report the top features

4. EXPERIMENTAL SET-UP

4.1 Dataset and protocols

Table 1. NSL-KDD class distribution.

Class	KDDTrain+	KDDTest+	Novel types in KDDTest+
Normal	67,343	9,711	—
DoS	45,927	7,458	apache2, mailbomb, processtable, udpstorm
Probe	11,656	2,421	mscan, saint
UA (R2L + U2R)	1,047	2,954	httptunnel, named, ps, sendmail, snmpgetattack, snmpguess, sqlattack, worm, xlock, xsnoop, xterm
Total	**125,973**	**22,544**	**17 types, 3,750 records**

Table 1 gives the class distribution. Two protocols are used. Protocol A (known attacks) splits KDDTrain+ into 80% training and 20% test records with stratification and a fixed seed, so training and test share the same attack types. Protocol B (zero-day test) trains on all of KDDTrain+ and tests on KDDTest+, where 3,750 records belong to 17 attack types that never appear in training and the UA class is almost three times larger than in training. Protocol B therefore measures both generalisation to unseen attack types and robustness to a shift in class proportions.

4.2 Baselines, metrics and environment

Seven baselines use the same pre-processing: logistic regression (LR), a decision tree (DT), a random forest (RF, 200 trees), an unweighted MLP, the weighted MLP (W-MLP), XGBoost and LightGBM. We report accuracy, macro-averaged precision, recall and F1-score, and weighted F1-score; macro averaging gives the small UA and Probe classes equal weight, so macro F1 is the main criterion. For each class c with precision P_c and recall R_c,

$$Macro - F1 = \frac{1}{K}\sum_{c=1}^{K}\frac{2P_cR_c}{P_c+R_c} \quad (6)$$

For Protocol B we also report the detection rate on known attack types, the detection rate on novel attack types (fraction flagged as any attack class) and the false-alarm rate on normal traffic. All experiments ran in Python with scikit-learn, XGBoost, LightGBM and SHAP on a two-core CPU without a GPU.

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Journal of Information Systems Engineering and Management
2024, 9(3)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

5. RESULTS AND DISCUSSION

5.1 Known-attack performance (Protocol A)

Table 2. Known-attack results (Protocol A, 25,195 test records).

Model	ACC (%)	PRE (%)	REC (%)	F1 (%)	W-F1 (%)	Train (s)
LR	99.28	95.30	93.76	94.51	99.27	5.9
DT	99.80	98.37	98.39	98.38	99.80	1.3
RF	99.87	99.78	98.31	99.02	99.87	7.5
MLP	99.78	99.50	96.95	98.15	99.77	18.3
W-MLP	99.71	95.58	98.70	97.03	99.72	23.5
XGBoost	99.90	99.42	99.16	99.29	99.90	17.0
LightGBM	99.91	99.42	99.09	99.25	99.91	9.3
WSE-XIDS (proposed)	**99.85**	**97.28**	**99.42**	**98.31**	**99.86**	**311.4**

Table 2 shows that every model except LR exceeds 99.7% accuracy when the test attacks are of known types. WSE-XIDS reaches 99.85% accuracy and 98.31% macro F1-score, and it has the highest macro recall (99.42%) of all models, because the balanced meta-learner recovers 205 of the 209 UA test records. It is not the most accurate model on this protocol: LightGBM (99.91%) and XGBoost (99.90%) are slightly more accurate and have higher macro F1-scores (99.25% and 99.29%), because the balanced weighting of WSE-XIDS trades 21 false UA alarms on normal traffic for its higher minority recall. With differences in the second decimal place, Protocol A cannot separate the strong models; it mainly confirms that NSL-KDD known attacks are easy for any non-linear learner.

5.2 Zero-day performance (Protocol B)

Table 3. Zero-day results on KDDTest+ (Protocol B, 22,544 records).

Model	ACC (%)	PRE (%)	REC (%)	F1 (%)	Binary ACC (%)	Novel DR (%)	FAR (%)
LR	74.27	68.85	61.14	59.93	75.56	31.04	7.42
DT	75.98	84.69	61.54	61.89	77.40	31.47	3.62
RF	75.47	86.44	60.26	60.43	76.46	23.60	2.62
MLP	75.97	79.04	63.93	64.64	77.49	33.33	7.33
W-MLP	76.43	82.44	64.46	65.49	77.71	32.91	7.35
XGBoost	78.03	85.58	64.32	63.97	79.80	41.65	2.87
LightGBM	76.36	85.57	62.62	63.91	77.79	26.48	2.92
WSE-XIDS (proposed)	**77.83**	**84.41**	**65.59**	**67.07**	**79.26**	**35.81**	**5.07**

DR: detection rate (novel-attack records flagged as any attack class); FAR: false-alarm rate on normal traffic; binary accuracy treats all attack classes as one.

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Journal of Information Systems Engineering and Management
2024, 9(3)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Table 3 shows how much harder the unseen-attack setting is. Accuracy drops by more than 20 points for every model. WSE-XIDS gives the best macro F1-score (67.07%), the best macro recall (65.59%) and the best weighted F1-score (75.22%), ahead of the weighted MLP (65.49%) and XGBoost (63.97%). XGBoost, however, is slightly more accurate (78.03% versus 77.83%), detects more novel-attack records (41.65% versus 35.81%) and raises fewer false alarms (2.87% versus 5.07%). The stacking ensemble therefore improves the balance across classes rather than raw attack coverage, and no model detects even half of the novel-attack records.

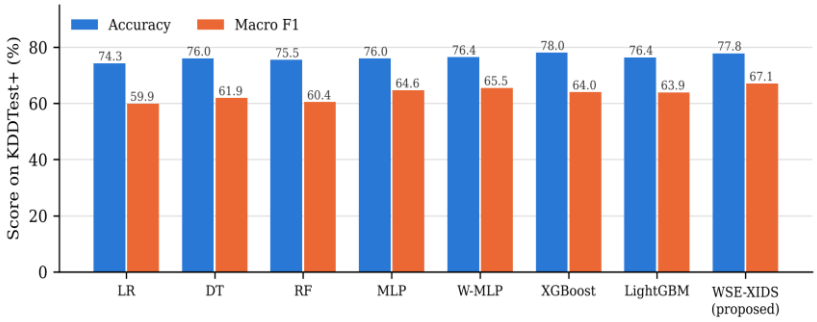


Figure 2. Accuracy and macro F1-score on the zero-day test.

Figure 2 plots accuracy and macro F1-score for all models on KDDTest+. The gap between the two bars, about 11 to 15 points for every model, comes from the UA class: it is rare in training but large in the test set, so models that miss it lose much more macro F1 than accuracy. The ensemble narrows this gap most, which is the main benefit of combining a smooth neural learner with two boosted tree learners under a balanced meta-learner.

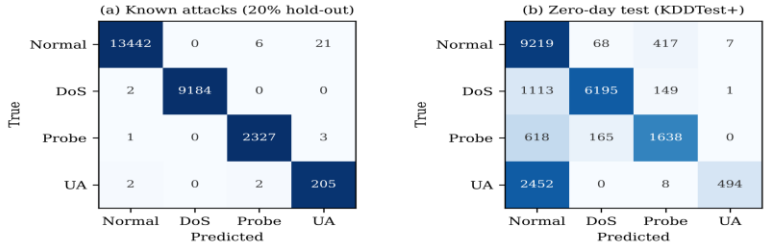


Figure 3. Confusion matrices of WSE-XIDS.

Figure 3 compares the confusion matrices of WSE-XIDS under the two protocols. On known attacks (a) almost all records lie on the diagonal. On the zero-day test (b), recall is 94.93% for Normal, 83.07% for DoS and 67.66% for Probe, but only 16.72% for UA: 2,452 of the 2,954 UA records are classified as normal. The UA test class is dominated by guess_passwd (1,231 records) and warezmaster (944), whose test-set versions differ from the few training examples, and it adds 718 records of novel types such as snmpguess and snmpgetattack; these attacks consist of short, low-volume sessions that look much like legitimate logins. Precision stays high for UA (98.41%), so the alerts that are raised are reliable, but the coverage of stealthy unseen attacks is clearly insufficient.

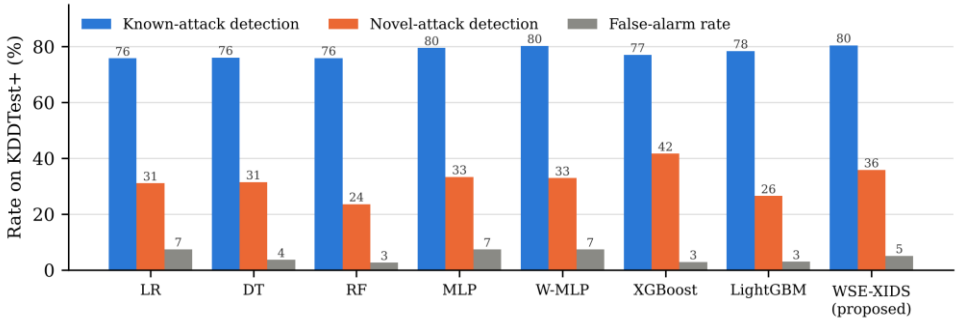


Figure 4. Known- and novel-attack detection and false alarms.

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Journal of Information Systems Engineering and Management
2024, 9(3)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Figure 4 separates detection on attack types seen during training from detection on the 17 novel types. All models detect about 76% to 80% of known-type attack records in KDDTest+, but only 24% to 42% of novel-type records. WSE-XIDS has the highest known-type detection (80.45%) and the second-highest novel-type detection (35.81%), at a false-alarm rate of 5.07%, between the tree ensembles (about 3%) and the MLPs (about 7%).

5.3 Explainability

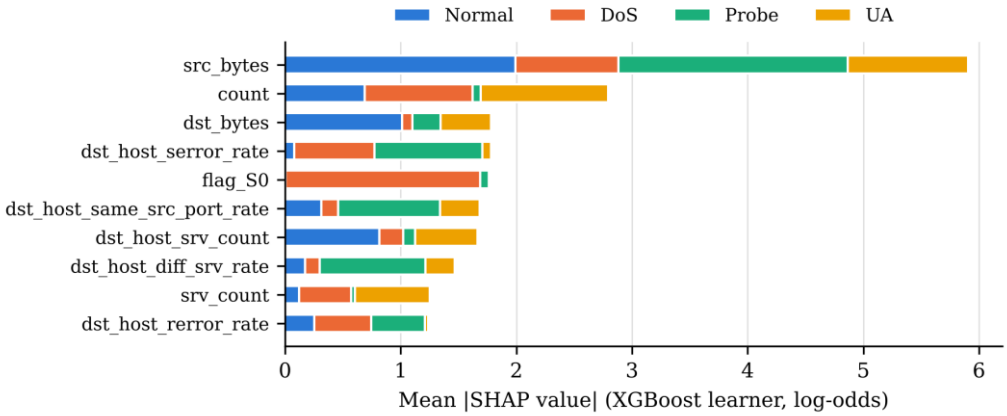


Figure 5. Top-10 SHAP features of the XGBoost learner.

Figure 5 shows the ten features with the largest mean absolute SHAP values, split by class. Source bytes is the most influential feature for every class and dominates Probe, connection count and flag S0 (half-open connections) drive DoS decisions, and destination bytes and destination-host service count separate normal from UA traffic. These patterns agree with the nature of the attacks: floods create many half-open connections, scans touch many services with little payload, and remote-access attacks move small amounts of data within otherwise normal sessions, which also explains why unseen UA attacks are hard to separate from normal traffic. The meta-learner assigns 25.5% of its absolute weight to the MLP, 36.7% to XGBoost and 37.7% to LightGBM, so all three learners contribute.

5.4 Comparison with published NSL-KDD results

Table 4. Comparison with published results on KDDTest+.

Study	Model	Classes	Accuracy on KDDTest+ (%)
Javaid et al. [12]	Self-taught learning (sparse autoencoder)	2 / 5	88.39 / 79.10
Yin et al. [13]	Recurrent neural network	2 / 5	83.28 / 81.29
This work	XGBoost baseline	2 / 4	79.80 / 78.03
This work	WSE-XIDS (proposed)	2 / 4	79.26 / 77.83

Published values are as reported by the original authors; their pre-processing and class definitions (five classes versus our four) differ, so the comparison is indicative only.

Table 4 places the results beside published deep models evaluated on KDDTest+. WSE-XIDS is in the same range but below the recurrent network of Yin *et al.* [13] and the self-taught learner of Javaid *et al.* [12] in both binary and multi-class accuracy. Its advantages are lower training cost (about 5 minutes on a two-core CPU, including five-fold stacking), balanced class performance and built-in explanations, not higher headline accuracy.

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

5.5 Limitations

First, NSL-KDD is old and does not represent modern encrypted or IoT traffic; results should be confirmed on newer data such as CIC-IDS2017 [10] or UNSW-NB15 [11]. Second, the zero-day test maps novel attacks to known families, which is easier than open-set detection of completely new behaviour. Third, recall on unseen UA attacks is low, so WSE-XIDS should be paired with anomaly-based detectors [15], [16]. Fourth, all results come from single runs with a fixed seed, and SHAP explains only the XGBoost learner rather than the full ensemble.

6. CONCLUSION

This paper presented WSE-XIDS, a weighted stacking ensemble of an MLP, XGBoost and LightGBM with a class-balanced logistic regression meta-learner and SHAP explanations, and evaluated it on NSL-KDD under separate known-attack and zero-day protocols. On known attacks it reached 99.85% accuracy with the highest macro recall (99.42%); on the zero-day test it reached 77.83% accuracy and the best macro F1-score (67.07%) of eight models, detected 35.81% of novel-attack records and kept false alarms at 5.07%. The 22-point drop between the protocols shows that known-attack accuracy alone greatly overstates readiness for zero-day attacks. Future work will add anomaly-based components for unseen remote-access attacks, explain the full ensemble, and evaluate on modern traffic data sets.

REFERENCES

[1] L. Bilge and T. Dumitraş, "Before we knew it: An empirical study of zero-day attacks in the real world," in *Proc. ACM Conference on Computer and Communications Security (CCS)*, 2012, pp. 833–844.

[2] R. Ahmad, I. Alsmadi, W. Alhamdani, and L. Tawalbeh, "Zero-day attack detection: A systematic literature review," *Artificial Intelligence Review*, vol. 56, no. 10, pp. 10733–10811, 2023.

[3] I. Stellios, P. Kotzanikolaou, and M. Psarakis, "Advanced persistent threats and zero-day exploits in industrial Internet of Things," in *Security and Privacy Trends in the Industrial Internet of Things*. Cham, Switzerland: Springer, 2019, pp. 47–68.

[4] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1153–1176, 2016.

[5] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: Techniques, datasets and challenges," *Cybersecurity*, vol. 2, Art. no. 20, 2019.

[6] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in *Proc. IEEE Symposium on Security and Privacy*, 2010, pp. 305–316.

[7] R. Lippmann, J. W. Haines, D. J. Fried, J. Korba, and K. Das, "The 1999 DARPA off-line intrusion detection evaluation," *Computer Networks*, vol. 34, no. 4, pp. 579–595, 2000.

[8] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proc. IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA)*, 2009, pp. 1–6.

[9] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, "A survey of network-based intrusion detection data sets," *Computers & Security*, vol. 86, pp. 147–167, 2019.

[10] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. 4th International Conference on Information Systems Security and Privacy (ICISSP)*, 2018, pp. 108–116.

[11] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems," in *Proc. Military Communications and Information Systems Conference (MilCIS)*, 2015, pp. 1–6.

[12] A. Javaid, Q. Niyaz, W. Sun, and M. Alam, "A deep learning approach for network intrusion detection system," in *Proc. 9th EAI International Conference on Bio-inspired Information and Communications Technologies (BICT)*, 2016, pp. 21–26.

[13] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21954–21961, 2017.

[14] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep learning approach for intelligent intrusion detection system," *IEEE Access*, vol. 7, pp. 41525–41550, 2019.

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

[15] H. Hindy, R. Atkinson, C. Tachtatzis, J.-N. Colin, E. Bayne, and X. Bellekens, "Utilising deep learning techniques for effective zero-day attack detection," *Electronics*, vol. 9, no. 10, Art. no. 1684, 2020.
[16] T. Zoppi, A. Ceccarelli, and A. Bondavalli, "Unsupervised algorithms to detect zero-day attacks: Strategy and application," *IEEE Access*, vol. 9, pp. 90603–90615, 2021.
[17] M. Sarhan, S. Layeghy, M. Gallagher, and M. Portmann, "From zero-shot machine learning to zero-day attack detection," *International Journal of Information Security*, vol. 22, pp. 947–959, 2023.
[18] S. I. Popoola, R. Ande, B. Adebisi, G. Gui, M. Hammoudeh, and O. Jogunola, "Federated deep learning for zero-day botnet attack detection in IoT-edge devices," *IEEE Internet of Things Journal*, vol. 9, no. 5, pp. 3930–3944, 2022.
[19] A. Dahal, P. Bajgai, and N. Rahimi, "Analysis of zero day attack detection using MLP and XAI," in *Proc. World Congress in Computer Science, Computer Engineering & Applied Computing*. Cham, Switzerland: Springer, 2024, pp. 57–67.
[20] N. Mohamed, H. Taherdoost, and M. Madanchian, "Review on machine learning for zero-day exploit detection and response," in *Proc. International Conference on Smart Technology*. Cham, Switzerland: Springer, 2024, pp. 163–176.
[21] M. Wang, K. Zheng, Y. Yang, and X. Wang, "An explainable machine learning framework for intrusion detection systems," *IEEE Access*, vol. 8, pp. 73127–73141, 2020.
[22] B. Mahbooba, M. Timilsina, R. Sahal, and M. Serrano, "Explainable artificial intelligence (XAI) to enhance trust management in intrusion detection systems using decision tree model," *Complexity*, vol. 2021, Art. no. 6634811, 2021.
[23] D. H. Wolpert, "Stacked generalization," *Neural Networks*, vol. 5, no. 2, pp. 241–259, 1992.
[24] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 785–794.
[25] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "LightGBM: A highly efficient gradient boosting decision tree," in *Advances in Neural Information Processing Systems 30*, 2017, pp. 3146–3154.
[26] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. International Conference on Learning Representations (ICLR)*, 2015.
[27] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.
[28] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems 30*, 2017, pp. 4765–4774.
[29] S. M. Lundberg, G. Erion, H. Chen, A. DeGrave, J. M. Prutkin, B. Nair, R. Katz, J. Himmelfarb, N. Bansal, and S.-I. Lee, "From local explanations to global understanding with explainable AI for trees," *Nature Machine Intelligence*, vol. 2, no. 1, pp. 56–67, 2020.
[30] M. T. Ribeiro, S. Singh, and C. Guestrin, "'Why should I trust you?': Explaining the predictions of any classifier," in *Proc. 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 1135–1144.

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.