



Zero-Day Attack Detection Using a Multi-Layer Perceptron with Explainable AI

Dr.Arul Selvan Gnanamonickam A ¹, Ravivishnu I ²

¹ **Assistant Professor**, Department of Software Systems, KG College of Arts and Science, Coimbatore, Tamil Nadu, India. aarulselvan@gmail.com

² Department of Software Systems, KG College of Arts and Science, Coimbatore, Tamil Nadu, India. iravivishnu@gmail.com

ABSTRACT

Zero-day network attacks exploit previously unseen behaviour patterns and routinely evade signature-based intrusion detection systems. This paper presents a prototype intrusion-detection system built on a Multi-Layer Perceptron (MLP) trained on the NSL-KDD benchmark dataset [1], supported by an Explainable AI (XAI) layer built with SHAP [3] and LIME [4], and delivered through a Flask web application. The system processes 41 standard network-traffic features, encodes categorical fields, applies standardized scaling, and classifies each record as normal traffic or one of several named attack categories. On the saved multiclass evaluation the MLP achieved 70.07% accuracy, 65.13% weighted precision, 70.07% weighted recall, and 65.97% weighted F1-score, with a binary ROC-AUC of 0.9251. On the saved binary normal-versus-attack evaluation the model achieved 86.75% accuracy, 90.51% precision, 85.72% recall, and 88.05% F1-score. A Random Forest baseline [6] was trained for comparison, reaching 72.18% accuracy but a lower weighted F1-score of 61.83%, illustrating that accuracy alone is an insufficient metric on imbalanced intrusion data. To support a genuine zero-day claim rather than an ordinary multiclass result, a held-out-label experiment was conducted: three attack types (satan, smurf, and guess_passwd) were individually removed from training and evaluated as unseen classes. Detection rates for these held-out attacks ranged from 2.27% to 58.91%, with combined binary accuracy between 86.67% and 95.78%, showing that generalisation to unseen attack families is partial and attack-type dependent rather than uniformly strong. SHAP and LIME visualisations are used to explain the global and local contribution of individual features to model predictions. The results indicate that MLP-based detectors can provide useful, interpretable intrusion-detection support, but that zero-day generalisation claims must be evaluated per-attack-family rather than assumed.

Keywords: Zero-day attack detection; Intrusion Detection System (IDS); Multi-Layer Perceptron; NSL-KDD; Explainable AI; SHAP; LIME; Random Forest.

1. INTRODUCTION

This paper documents the design, training, and evaluation of a Multi-Layer Perceptron (MLP) classifier trained on the NSL-KDD dataset [1], a cleaned and widely used benchmark derived from the original KDD Cup 1999 intrusion-detection dataset. Beyond raw classification accuracy, the project incorporates an Explainable AI (XAI) layer to make individual predictions interpretable, and a Flask-based web interface so that uploaded traffic samples can be analysed and reviewed by a human operator. A central contribution of this work is that it separates an ordinary multiclass evaluation from a properly controlled zero-day evaluation, in which specific attack labels are withheld entirely from training and tested only at evaluation time.

2. PROBLEM STATEMENT

Rule-based and signature-based intrusion detection is slow to update, dependent on manually maintained rule sets, and offers little interpretability to a human analyst reviewing an alert. The problem addressed by this work is twofold: first, to classify NSL-KDD-compatible network records into normal traffic or a specific attack category using a learned model rather than hand-written rules; and second, to determine, through a controlled held-out experiment, how well such a model generalises to attack types it has never seen during training — the defining characteristic of a zero-day attack.

3. OBJECTIVES

- Build and train an MLP classifier on NSL-KDD network-traffic features for multiclass attack classification.
- Establish a Random Forest baseline [6] for comparison against the MLP on identical data splits.
- Report both multiclass metrics and binary (normal-versus-attack) metrics, since these answer different operational questions.

- Design and run a held-out-label experiment that removes individual attack types from training to produce a defensible zero-day detection measurement.
- Integrate SHAP [3] and LIME [4] explainability so that predictions are not a black box to the reviewing analyst.
- Deliver the trained model through a Flask web application supporting CSV upload, prediction, visualisation, and result download.

4. RELATED WORK

Intrusion detection using the KDD Cup 1999 dataset and its refined successor, NSL-KDD [1], has been studied extensively with a range of classical and deep-learning models, including decision trees, support vector machines, random forests [6], and multi-layer neural networks [2]. Ensemble tree methods such as Random Forest are commonly used as strong baselines because they handle mixed categorical/numerical features well and are comparatively robust to class imbalance. Broader surveys of data-mining and machine-learning approaches to intrusion detection catalogue this landscape in detail [19, 20].

Deep-learning architectures have also been applied directly to NSL-KDD and related traffic data: recurrent architectures such as LSTM networks have been used for sequence-style intrusion classification [7, 9], convolutional networks have been applied to traffic feature maps [8], fully-connected deep networks have been benchmarked against classical baselines [10], and non-symmetric deep autoencoder architectures have been proposed specifically for network intrusion detection [11]. Related work has also extended neural intrusion detection to IoT network traffic [21]. Beyond NSL-KDD, alternative benchmark datasets such as UNSW-NB15 [12] and CICIDS2017 [13] have been introduced to address known limitations of the original KDD-derived data. Neural network approaches, including MLPs and deeper architectures, are studied for their ability to model nonlinear feature interactions, though they typically require careful regularisation (dropout, batch normalisation, class weighting) to avoid overfitting on imbalanced intrusion data. Separately, model explainability techniques such as SHAP [3] and LIME [4] have become standard tools for auditing security-relevant machine-learning models, since operators need to understand why a record was flagged, not just that it was flagged. This paper builds on these established techniques and combines them with an explicit held-out-class methodology for evaluating zero-day generalisation, which is often assumed rather than directly measured in comparable NSL-KDD studies [18].

5. DATASET AND PREPROCESSING

The project uses the NSL-KDD dataset (KDDTrain+ and KDDTest+) [1], which provides 125,973 labelled training records and 22,544 labelled test records. Each record has 41 network-traffic features spanning basic connection attributes (duration, protocol type, service, flag), content features, and traffic-based statistical features, plus an attack-name label and a difficulty-level field that is excluded from model input.

Three categorical features — protocol_type, service, and flag — are label-encoded into integer form. All 41 numerical input features are then standardised using scikit-learn's StandardScaler [5], which is fitted on the training split and reused, unmodified, on the test split and on any record submitted through the web application, to avoid data leakage and ensure consistent inference-time behaviour. The fitted scaler and label encoders are persisted (scaler.pkl, label_encoder.pkl) so that the same transformation is guaranteed across training, evaluation, and deployment.

For the primary multiclass model, the attack label warezclient is deliberately removed from the training rows before training. This removal on its own does not constitute a zero-day evaluation; a defensible zero-day conclusion requires the dedicated held-out experiment described in Section 7, run and reported separately from the ordinary multiclass results. Newer benchmark datasets such as UNSW-NB15 [12] and CICIDS2017 [13] were considered as alternatives to NSL-KDD but were not used for the primary experiments in this paper; they are identified in Section 10 as a direction for cross-dataset validation.

System Methodology / Architecture Pipeline

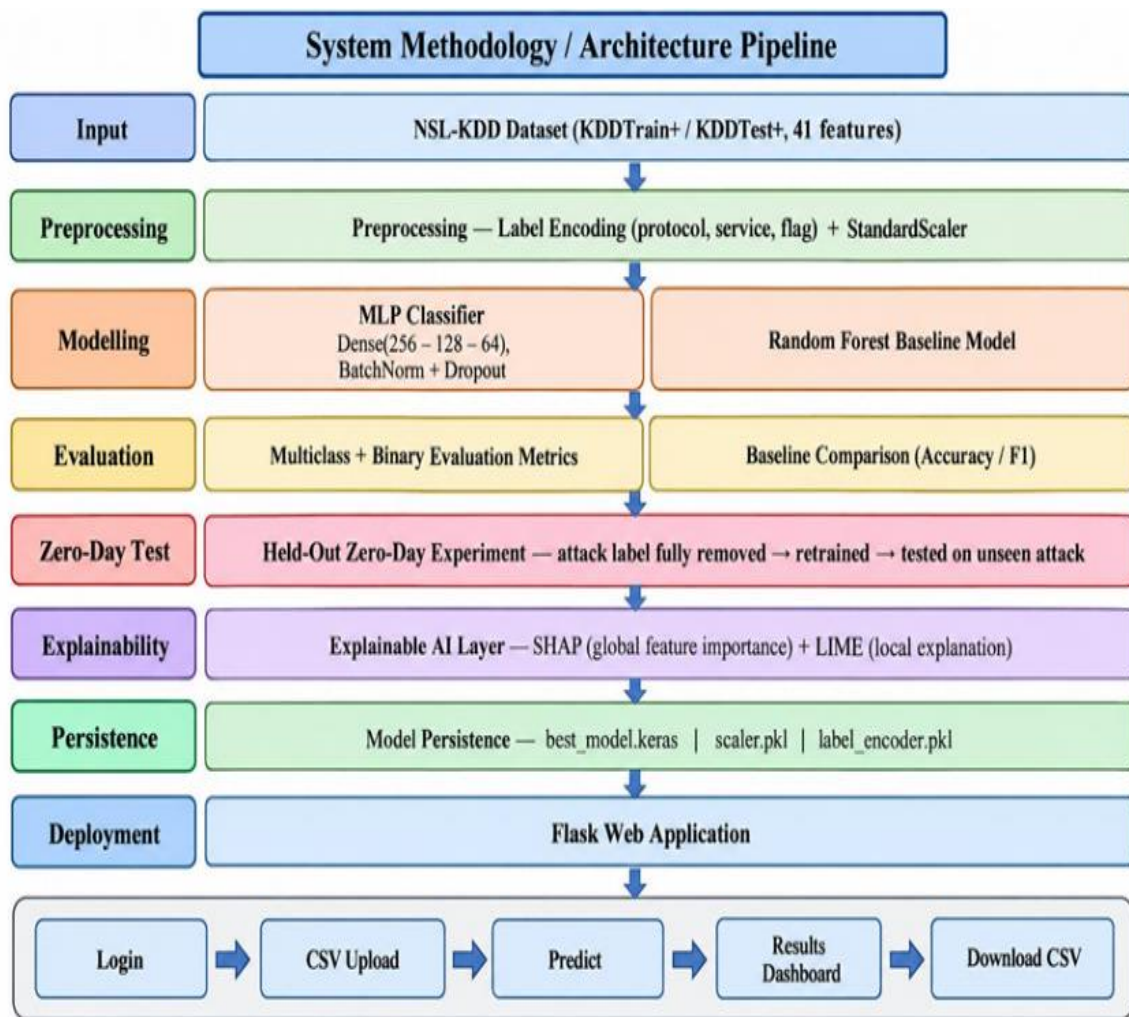


Figure 1. End-to-end system methodology and architecture pipeline, from raw NSL-KDD records through preprocessing, MLP/Random Forest modelling, evaluation, the held-out zero-day experiment, SHAP/LIME explainability, model persistence, and Flask deployment.

6. Methodology

6.1 MLP Architecture

The classifier is a feed-forward Multi-Layer Perceptron, implemented with TensorFlow/Keras [2], with a 41-feature input layer, followed by three hidden blocks of Dense(256), Dense(128), and Dense(64) units, each using ReLU activation and followed by Batch Normalization and Dropout (rates 0.30, 0.30, and 0.20 respectively). The output layer is a softmax layer sized to the number of attack classes in the training data. The network has 56,231 total parameters, of which 55,335 are trainable.

The model is compiled with the Adam optimizer [16] and sparse categorical cross-entropy loss. Batch Normalization [14] is applied after each dense layer to stabilise and accelerate training, and Dropout [15] is used for regularisation against overfitting. To address class imbalance, per-class training weights are computed and applied during fitting rather than resampling the dataset. Training uses EarlyStopping, ReduceLROnPlateau, and ModelCheckpoint callbacks, is configured for up to 30 epochs with a batch size of 64, and the best-performing checkpoint (by validation loss) is persisted as best_model.keras.

6.2 Baseline Model

A Random Forest classifier [6], implemented via scikit-learn [5], is trained on the same preprocessed features as a non-neural baseline, allowing the MLP's behaviour to be interpreted relative to a well-understood ensemble method rather than in isolation.

6.3 Web Application

The trained MLP, scaler, and label encoder are served through a Flask web application [22]. The interface supports user login, CSV upload with validation against the expected 41-feature schema, a choice between quick analysis (first 100 records, for fast demonstrations) and full analysis (all uploaded records), a results dashboard showing per-class counts, percentages and prediction confidence, an interactive chart of the attack-family distribution, a downloadable CSV of predictions, and links to the saved SHAP and LIME explanation artifacts.

7. Zero-Day Evaluation Methodology

A record being misclassified in an ordinary multiclass test split does not demonstrate zero-day generalisation, because most attack types in that split are still represented in the training data. To produce a defensible zero-day measurement, this project uses a strict held-out-label protocol:

- Select a single attack label (for example, `satan`) and remove every training record carrying that label.
- Train a separate binary experiment model on normal traffic versus the remaining known attacks.
- Evaluate the model only on normal traffic and on the held-out attack's test records, which the model has never encountered in any form during training.
- Report the zero-day detection rate (recall on the unseen attack), the normal true-negative rate, and the combined binary accuracy across both classes.

This procedure was repeated independently for three attack types of differing character: `guess_passwd` (an R2L-style credential-guessing attack), `satan` (a probe/scanning attack), and `smurf` (a high-volume DoS attack), so that zero-day performance could be compared across attack families rather than reported for a single case.

8. Results

8.1 Multiclass MLP Evaluation

Metric	Result
Accuracy	70.07%
Weighted Precision	65.13%
Weighted Recall	70.07%
Weighted F1-score	65.97%
Binary ROC-AUC	0.9251

Table 1. Saved multiclass evaluation of the MLP on the NSL-KDD test split.

8.2 Binary (Normal vs. Attack) Evaluation

Metric	Result
Accuracy	86.75%
Precision	90.51%
Recall	85.72%
F1-score	88.05%

Table 2. Saved binary normal-versus-attack evaluation of the MLP.

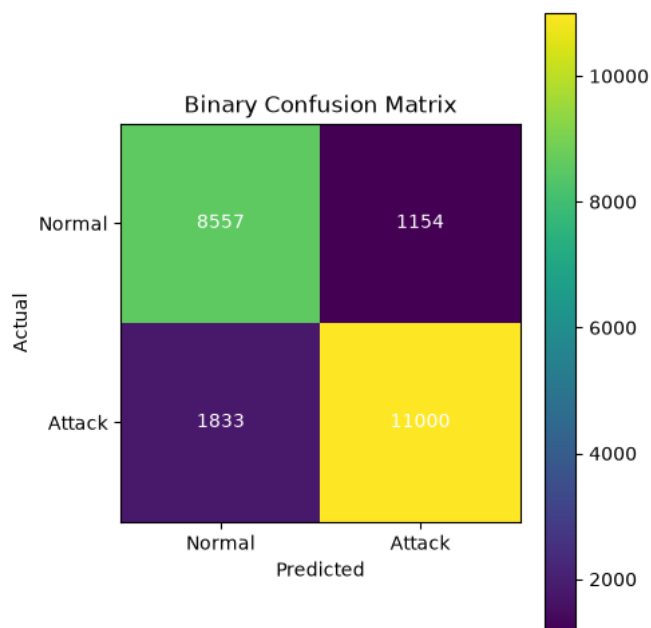


Figure 2. Binary confusion matrix (normal vs. attack).

8.3 Baseline Comparison

Model	Acc.	Prec.	Rec.	F1
MLP	70.07%	65.13%	70.07%	65.97%
Random Forest	72.18%	56.96%	72.18%	61.83%

Table 3. MLP vs. Random Forest on identical multiclass data.

Figure 3. Visual comparison of MLP and Random Forest metrics.

Random Forest attains higher raw accuracy, while the MLP attains a higher weighted F1-score. This divergence is characteristic of imbalanced multiclass intrusion data, where a model can score well on accuracy by favouring majority classes while performing worse on minority attack categories that weighted F1 penalises more visibly. This is the reason both metrics, not accuracy alone, are reported throughout this paper.

8.4 Training Behaviour

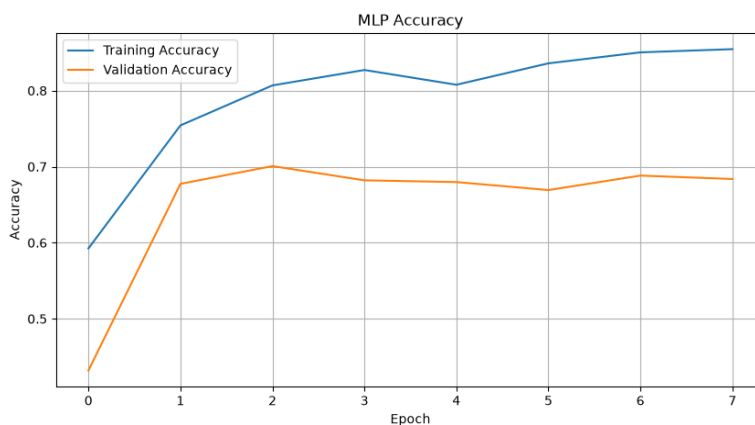


Figure 4. Training and validation accuracy across epochs.

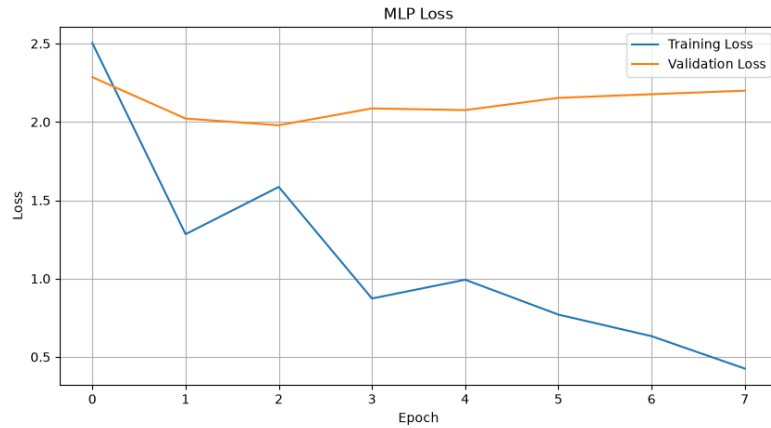


Figure 5. Training and validation loss across epochs.

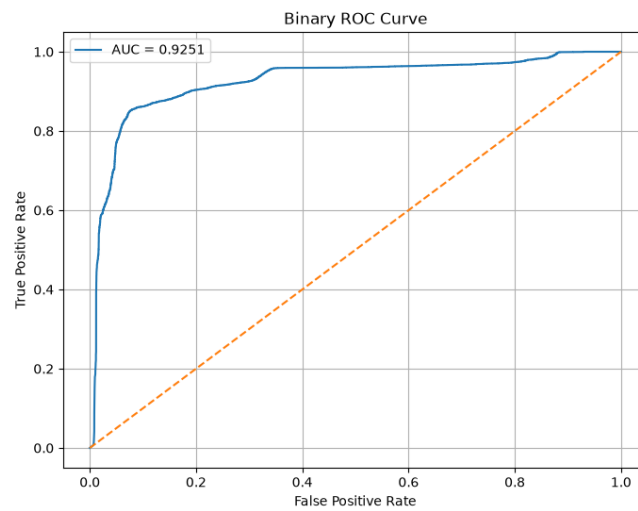


Figure 6. Binary ROC curve for the MLP (AUC = 0.9251).

8.5 Held-Out Zero-Day Experiment Results

The three independent held-out-label experiments produced the results shown in Table 4 below. Each row corresponds to a separate model trained with the named attack fully excluded, then tested only on normal traffic and that excluded attack's test records.

Held-out attack	Removed from training	Held-out test records	Normal test records	Zero-day detection rate	Normal true-negative rate	Combined binary accuracy
guess_passwd	53	1,231	9,711	2.27%	97.36%	86.67%
satan	3,633	735	9,711	58.91%	98.57%	95.78%
smurf	2,646	665	9,711	3.76%	97.57%	91.56%

Table 4. Held-out zero-day detection results for three withheld attack types.

The results show that zero-day generalisation is highly attack-dependent rather than uniform. The probe-type attack *satan*, which is structurally closer in feature space to other probe and scan patterns retained in training, achieved a 58.91% detection rate as a completely unseen class. In contrast, the DoS-type attack *smurf* (3.76% detection) and the R2L-type attack *guess_passwd* (2.27% detection) were detected only rarely when withheld entirely, despite each producing a high normal true-negative rate (97.36%–98.57%) and a respectable combined binary accuracy (86.67%–95.78%) driven mostly by correct classification of normal traffic. This distinction matters operationally: a high combined accuracy figure alone would overstate zero-day capability

for guess_passwd and smurf, because it is inflated by the large number of correctly classified normal records rather than by genuine detection of the unseen attack.

These findings support a specific, evidence-based claim: the MLP has a demonstrated but partial ability to generalise to previously unseen attack types, with clearly stronger generalisation for attacks that resemble already-known attack families in feature space, and weak generalisation for attacks whose traffic signatures are more distinct from anything seen in training.

9. Explainable AI

To avoid presenting the MLP as an un-auditable black box, SHAP [3] is used to compute global feature-contribution values across the evaluation set, and LIME [4] is used to explain a single representative prediction at the local level. Together these techniques let a reviewing analyst see which of the 41 input features drove a given classification, which supports trust and manual verification of automated alerts, though neither technique establishes a causal relationship between a feature and the underlying network event — both describe correlational contribution within the trained model.

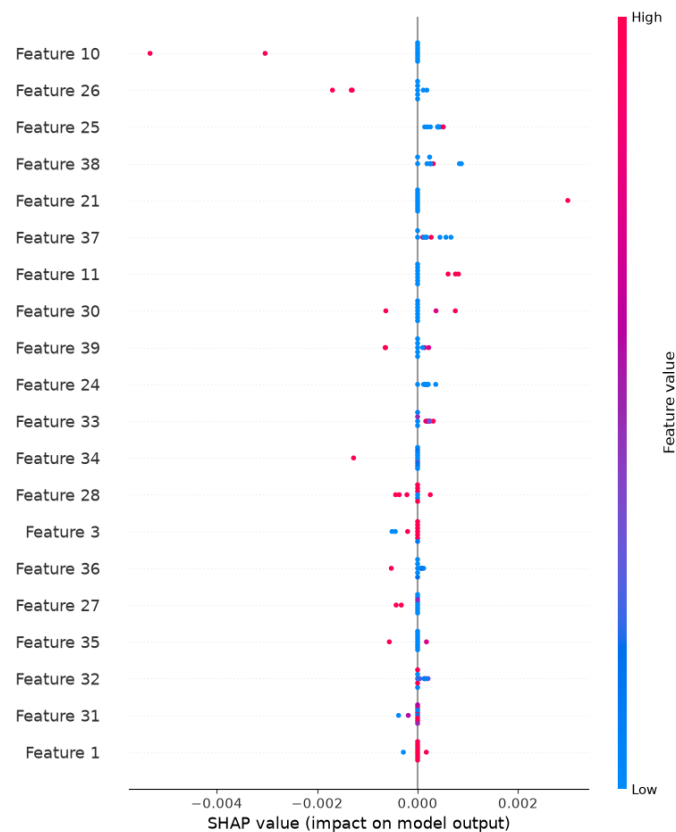


Figure 7. SHAP summary plot showing global feature contribution and impact direction.

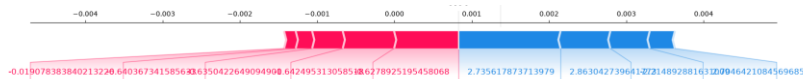


Figure 8. SHAP force plot for a single representative prediction.

10. Limitations and Future Work

As Sommer and Paxson observe, machine-learning intrusion detectors that perform well on closed benchmark splits often generalise poorly once deployed outside the closed world of their training distribution [18], which is precisely the gap the held-out zero-day experiment in this paper attempts to measure directly rather than assume. Broader surveys of the field report a similar pattern across many ML/DL-based network intrusion detectors [19, 20].

- The dataset is historical and offline; this prototype is not validated against live, real-time network traffic.
- The zero-day experiment was run for three attack types; a fuller study would repeat this protocol across every available attack label to build a complete per-attack generalisation profile, and would benefit from cross-dataset validation on UNSW-NB15 [12] or CICIDS2017 [13].

- The saved LIME artifact explains a single selected sample rather than every uploaded record; per-upload LIME generation was not implemented in this prototype.
- Authentication in the web application is not backed by a production-grade user database.
- Class-imbalance handling currently relies on per-class training weights; resampling approaches such as SMOTE [17] were not evaluated but are a natural extension.
- Future work should add live packet-capture integration, continuous model monitoring for concept drift, and periodic retraining as new attack families are documented.

11. Conclusion

This paper presented an end-to-end network intrusion-detection prototype combining MLP-based multiclass classification, a Random Forest baseline for comparison, binary normal-versus-attack evaluation, a controlled held-out-label protocol for measuring genuine zero-day generalisation, SHAP/LIME explainability, and a Flask web interface for practical use. The MLP achieved competitive multiclass and binary performance relative to the Random Forest baseline, and the held-out experiments showed that zero-day detection capability is real but uneven across attack families — strong for attacks resembling known patterns (58.91% for *satan*) and weak for more distinct attack signatures (2.27%–3.76% for *guess_passwd* and *smurf*). Reporting these results transparently, rather than inferring a single blanket zero-day accuracy figure, is presented as the more defensible way to evaluate and communicate this class of system.

References

- [1] Tavallae, M., Bagheri, E., Lu, W., and Ghorbani, A. A. "A Detailed Analysis of the KDD CUP 99 Data Set." *Proceedings of the IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 2009.
- [2] Abadi, M. et al. "TensorFlow: A System for Large-Scale Machine Learning." *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2016.
- [3] Lundberg, S. M., and Lee, S.-I. "A Unified Approach to Interpreting Model Predictions." *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.
- [4] Ribeiro, M. T., Singh, S., and Guestrin, C. "“Why Should I Trust You?” Explaining the Predictions of Any Classifier." *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.
- [5] Pedregosa, F. et al. "Scikit-learn: Machine Learning in Python." *Journal of Machine Learning Research*, 12, 2011.
- [6] Breiman, L. "Random Forests." *Machine Learning*, 45(1), 2001.
- [7] Kim, J., Kim, J., Thu, H. L. T., and Kim, H. "Long Short Term Memory Recurrent Neural Network Classifier for Intrusion Detection." *International Conference on Platform Technology and Service (PlatCon)*, 2016.
- [8] Vinayakumar, R., Soman, K. P., and Poornachandran, P. "Applying Convolutional Neural Network for Network Intrusion Detection." *International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 2017.
- [9] Yin, C., Zhu, Y., Fei, J., and He, X. "A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks." *IEEE Access*, vol. 5, 2017.
- [10] Javaid, A., Niyaz, Q., Sun, W., and Alam, M. "A Deep Learning Approach for Network Intrusion Detection System." *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (BICT)*, 2016.
- [11] Shone, N., Ngoc, T. N., Phai, V. D., and Shi, Q. "A Deep Learning Approach to Network Intrusion Detection." *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 1, 2018.
- [12] Moustafa, N., and Slay, J. "UNSW-NB15: A Comprehensive Data Set for Network Intrusion Detection Systems." *Military Communications and Information Systems Conference (MilCIS)*, 2015.
- [13] Sharafaldin, I., Lashkari, A. H., and Ghorbani, A. A. "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization." *International Conference on Information Systems Security and Privacy (ICISSP)*, 2018.
- [14] Ioffe, S., and Szegedy, C. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift." *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, 2015.
- [15] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting." *Journal of Machine Learning Research*, vol. 15, 2014.
- [16] Kingma, D. P., and Ba, J. "Adam: A Method for Stochastic Optimization." *International Conference on Learning Representations (ICLR)*, 2015.

-
- [17] Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. "SMOTE: Synthetic Minority Over-sampling Technique." *Journal of Artificial Intelligence Research*, vol. 16, 2002.
- [18] Sommer, R., and Paxson, V. "Outside the Closed World: On Using Machine Learning for Network Intrusion Detection." *IEEE Symposium on Security and Privacy*, 2010.
- [19] Buczak, A. L., and Guven, E. "A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection." *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, 2016.
- [20] Ahmad, Z., Shahid Khan, A., Wai Shiang, C., Abdullah, J., and Ahmad, F. "Network Intrusion Detection System: A Systematic Study of Machine Learning and Deep Learning Approaches." *Transactions on Emerging Telecommunications Technologies*, vol. 32, no. 1, 2021.
- [21] Hodo, E., Bellekens, X., Hamilton, A., Dubouilh, P.-L., Iorkyase, E., Tachtatzis, C., and Atkinson, R. "Threat Analysis of IoT Networks Using Artificial Neural Network Intrusion Detection System." *International Symposium on Networks, Computers and Communications (ISNCC)*, 2016.
- [22] Pallets Projects. Flask Documentation. <https://flask.palletsprojects.com/>