



Generalization-Aware Evaluation of a Regularized Multi-Layer Perceptron for Binary Network Intrusion Detection on NSL-KDD

Mazin M. Ismael 

College of Computing and Informatics, University of Al-Hamdaniya, Nineveh 41006, Iraq

Corresponding Author Email: mazinmohammed@uohamdaniya.edu.iq

Copyright: ©2026 The author. This article is published by IIETA and is licensed under the CC BY 4.0 license (<http://creativecommons.org/licenses/by/4.0/>).

<https://doi.org/10.18280/isi.310926>

ABSTRACT

Received: 26 May 2026

Revised: 13 August 2026

Accepted: 31 August 2026

Available online: 30 September 2026

Keywords:

network intrusion detection, multi-layer perceptron, NSL-KDD, distribution shift, generalization analysis, leakage-controlled evaluation

Reliable evaluation of network intrusion detection models requires more than high accuracy on internal data splits, particularly when benchmark test distributions differ from the training data. This study presents a leakage-controlled and generalization-aware evaluation of a regularized multi-layer perceptron (MLP) for binary intrusion detection using the NSL-KDD benchmark. The model incorporates batch normalization (BN), LeakyReLU activation, L2 regularization, dropout, and class-weighted learning, while preprocessing parameters are fitted exclusively on the training data. Performance is assessed on both a KDDTrain+-derived internal hold-out set and the untouched official KDDTest+ set, with the latter treated as the primary measure of generalization under distribution shift. Across five independent random seeds, the model achieved $78.14\% \pm 0.46\%$ accuracy, $92.37\% \pm 0.32\%$ precision, $67.15\% \pm 0.81\%$ recall, $77.76\% \pm 0.57\%$ F1-score, and an area under the receiver operating characteristic curve (AUC-ROC) of 0.8767 ± 0.0162 on KDDTest+. In contrast, the internal hold-out accuracy reached 99.43%, revealing a substantial generalization gap. Threshold analysis showed that lowering the decision threshold improved recall but did not eliminate the missed-attack problem under the shifted test distribution. These findings show that strong within-distribution performance can substantially overestimate intrusion-detection capability. The study therefore provides a reproducible MLP baseline and highlights the importance of leakage control, distribution-aware testing, and cautious interpretation of benchmark results.

1. INTRODUCTION

The proliferation of interconnected information systems, cloud services, edge platforms, and internet-connected devices has led to substantial growth in network traffic and a wider variety of cyber threats. Network intrusion detection systems (NIDSs) are therefore important analytical tools for identifying traffic patterns associated with malicious activity. Signature-based systems compare observed traffic with known attack patterns and generate alerts when those patterns are detected, whereas anomaly-based systems learn or estimate normal behavior and report deviations from that baseline. The latter can assist in detecting previously unseen behavior, but their effectiveness depends on the representativeness of the training data, the false-alarm rate, and performance stability under changes in traffic distribution [1, 2].

Machine learning and deep learning have broadened the design space for anomaly-based intrusion detection by enabling nonlinear feature interactions to be learned from high-dimensional network data. Recent studies have investigated autoencoder-assisted deep networks, recurrent and hybrid architectures, cost-sensitive learning, self-supervised graph representations, and sequential deep models [3-9]. These advances demonstrate the flexibility of data-driven NIDSs, but they also show that reported accuracy

values are highly sensitive to dataset age, class balance, preprocessing choices, and evaluation protocol.

A connection-level classifier should be considered one component of a layered control architecture rather than a replacement for signature-based, session-level, or cryptographic controls. Its usefulness depends not only on classification accuracy, but also on attack recall, false-positive behavior, computational cost, calibration, and robustness to out-of-distribution traffic. Accordingly, this study emphasizes evaluation discipline and generalization analysis rather than claims of production-deployment readiness.

NSL-KDD continues to be used as a historical benchmark because it reduces many of the duplicate-record problems present in KDD Cup 1999 and facilitates reproducible comparisons [10]. However, its traffic behavior and attack patterns reflect an earlier generation of network environments. More recent traffic characteristics and attack scenarios are represented in contemporary benchmarks such as CICIDS-2017 [11] and UNSW-NB15 [12], while recent research has increasingly emphasized cross-environment generalizability and robustness in addition to within-dataset accuracy [9, 13-15]. This distinction is central to the present evaluation.

The study does not claim a novel neural-network architecture. Rather, it reports four evaluation-oriented contributions: (1) a reproducible regularized multi-layer

perceptron (MLP) baseline that combines established batch normalization (BN), LeakyReLU, L2, and dropout components; (2) a leakage-controlled training and evaluation protocol in which learned transformations are fitted to training data only; (3) threshold-sensitivity and computational-cost analyses; and (4) a direct contrast between confirmatory multi-seed KDDTest+ results and exploratory single-seed ablation and imbalance-handling comparisons.

NIDSs are commonly described along two dimensions: signature-based versus anomaly-based detection, and host-based versus network-based monitoring. Signature-based NIDSs perform well when reliable signatures are available for known attacks, but their effectiveness is limited when attack signatures change or when a new attack has no established signature. In contrast, anomaly-based methods model typical or previously observed behavior and identify deviations from that baseline, which can improve sensitivity to novel behavior but may also increase false-positive alerts. This trade-off supports the use of precision, recall, F1-score, receiver operating characteristic (ROC) measures, and threshold-dependent error analysis rather than accuracy alone [1].

Deep models are helpful for learning nonlinear representations from mixed numerical and categorical network features, though there is no guarantee that more complex models will generalize better. NSL-KDD is a tabular dataset at the connection level, instead of a raw sequence of packets, which means that it has a relatively straightforward baseline with a limited number of architectural assumptions, provided by a feed-forward MLP. Since regularization is applied to control the fitting behavior in KDDTrain+ in this study, it cannot be assumed that these choices lead to better performance on a shifted test distribution.

2. RELATED WORK

2.1 Machine learning and tabular intrusion detection

Recent intrusion-detection studies continue to show that classical or shallow learning approaches can remain competitive when applied to well-structured network-flow features. Zhou et al. [16] combined feature selection and ensemble learning techniques, and Ali et al. [2] directly compared machine-learning and deep learning methods on contemporary intrusion-detection data. These studies caution against assuming that detection quality is determined primarily by model depth: preprocessing, class imbalance, feature representation, and evaluation design can be as influential as architectural choice.

Accordingly, performance comparisons should be protocol-specific. Results obtained under different class formulations, feature sets, resampling strategies, internal splits, or official benchmark partitions are not directly comparable. A stricter evaluation protocol may yield lower accuracy while providing a more informative estimate of generalization, particularly when a model is assessed under distribution shift rather than only on a random within-dataset split.

2.2 Deep learning for intrusion detection

Research on deep learning NIDSs has explored a wide range of representation and optimization strategies. Kunang et al. [3] applied deep autoencoder pretraining to a Deep Neural

Network (DNN), while Duan et al. [4] combined an improved artificial bee colony algorithm with a backpropagation neural network. Wang et al. [5] evaluated integrated deep models across multiple network and host datasets, and Staudemeyer [6] applied long short-term memory (LSTM) recurrent neural networks to intrusion detection. To address minority attacks in NSL-KDD, CIDDS-001, and CICIDS-2017, Gupta et al. [7] proposed cost-sensitive deep learning with ensemble stages.

More recent research has shifted toward generalizability and robustness. Caville et al. [8] applied self-supervised graph representations to network anomaly detection, Sarhan et al. [13] explored feature transfer across different network environments, Yan and Han [17] studied a stacked sparse autoencoder (SSAE) for automatic feature extraction in intrusion detection, and Hore et al. [9] explicitly addressed zero-day, out-of-distribution, and adversarial conditions. Similarly, Farhan et al. [15] evaluated a deep learning model on UNSW-NB15. Overall, this literature supports a more conservative interpretation of results obtained from a single legacy benchmark and motivates the generalization-focused language adopted in the present study.

2.3 Benchmark dataset evolution and evaluation validity

For many years, the KDD Cup 1999 family of datasets has played a major role in benchmarking intrusion-detection systems. Tavallaee et al. [10] documented extensive duplication in KDD Cup 1999 and proposed the analysis that motivated the cleaner NSL-KDD benchmark. NSL-KDD reduces duplicate-record effects and provides KDDTrain+ and official KDDTest+ partitions, allowing researchers to distinguish performance on data drawn from the training distribution from performance on an official test set containing differently represented attack patterns.

Later benchmarks such as CICIDS-2017 [11] and UNSW-NB15 [12] were developed to represent more modern attack activities and network topologies. The present study does not treat NSL-KDD as sufficient evidence for current production deployment; instead, it uses NSL-KDD to support reproducibility and to investigate the generalization gap between an internal split and the official test set. Validation on contemporary datasets is therefore a necessary next step rather than an optional extension.

2.4 Research gap and study positioning

The gap that is relevant here is not architectural, but methodological. Many of these complex IDS architectures are described in the literature, but it is not always easy to compare reported performance with other studies because there are many differences in data partitions, leakage controls, class definitions, how to deal with imbalance, and whether to perform the evaluation using an internal split or an official distribution-shifted test set [1, 9, 13]. This study thus serves as a controlled baseline with which to compare the results of the apparently high performance on the KDDTrain+-derived hold-out set within the context of the official KDDTest+ distribution to ask the more narrowly defined question: how much of the apparently high performance remains when the evaluation shifts from the KDDTrain+-derived hold-out to the official KDDTest+ distribution? The model's significance is that it allows that contrast to be replicated and made clear, rather than simply being defined as architectural novelty.

2.5 Methodological limitations in existing work

Several common evaluation practices can overestimate or obscure IDS performance. Preprocessing before data partitioning can leak information from test data into training; random within-dataset splits may contain examples that closely resemble training samples; class imbalance can obscure poor minority-attack recall; and aggregate accuracy can mask false-negative behavior across decision thresholds. Recent studies on generalization, transfer learning, and robustness further emphasize the importance of reporting the evaluation context and avoiding confusion between within-distribution fit and real-world effectiveness [1, 8, 9, 13, 14, 17–19]. In the present protocol, these concerns are addressed through training-only scaling, explicit separation of internal and official testing, five-seed KDDTest+ evaluation, threshold analysis, and cautious interpretation of exploratory single-seed comparisons.

3. METHODOLOGY

3.1 System overview and design rationale

The system consists of four steps: (i) data preparation and binary labeling, (ii) categorical encoding and numerical standardization, (iii) training of the regularized MLP, and (iv) testing on both KDDTrain+-derived partitions and the official KDDTest+ set. All preprocessing parameters are learned from the training data and then applied unchanged to the validation and test data. The implementation uses Python 3.9, TensorFlow 2.10, scikit-learn 1.0.2, and pandas 1.3.5. The reference run uses seed 42, while five independent random-seed runs are used to obtain the main KDDTest+ generalization statistics.

A feed-forward MLP was chosen because each NSL-KDD connection is represented as a fixed tabular feature vector rather than as an ordered sequence of raw packets. Recurrent architectures introduce a temporal inductive bias that is not directly aligned with this representation, while convolutional mappings impose spatial structure on heterogeneous features. The MLP is therefore used as a deliberately conventional baseline against which the effects of regularization and the evaluation protocol can be examined without introducing claims of a new architectural mechanism.

3.2 Dataset description

NSL-KDD is an improved benchmark based on the KDD Cup 1999/DARPA traffic lineage [10]. There are 41 input attributes for each record, an attack label, and a difficulty indicator. KDDTrain+ includes 125,973 records from the following attack families: denial-of-service (DoS), Probe, Remote-to-Local (R2L), and User-to-Root (U2R). The official KDDTest+ set is kept unchanged and contains 22,544 records for generalization analysis. The dataset statistics and partitioning used in this study are summarized in Table 1.

The experiments use a binary target in which all attack subcategories are grouped into a single attack class and normal traffic forms the second class. This formulation focuses the analysis on attack-versus-normal discrimination and allows the effect of distribution shift to be interpreted directly through attack precision and recall. It does not support conclusions about the model's ability to identify individual attack families.

Table 1. NSL-KDD dataset statistics and partitioning

Partition	Samples	Normal	Attack
KDDTrain+ (Full)	125,973	67,343 (53.5%)	58,630 (46.5%)
Train split (70%)	88,181	47,177 (53.5%)	41,004 (46.5%)
Validation (15%)	18,896	10,104 (53.5%)	8,792 (46.5%)
Test (15%)	18,896	8,795 (46.5%)	10,101 (53.5%)
KDDTest+ (Official)	22,544	9,711 (43.1%)	12,833 (56.9%)

3.3 Feature characterization

The 41 NSL-KDD input attributes can be grouped into basic traffic, content, time-window, and host-based categories, as summarized in Table 2. Basic features describe individual connections; content features capture indicators such as failed logins or compromised states; time-window features summarize recent connection activity; and host-based features describe activity associated with destination hosts. These heterogeneous feature types require explicit categorical encoding and standardized numerical scaling before MLP training.

Table 2. NSL-KDD feature categories and representative attributes

Feature Category	Count	Representative Features
Basic traffic features	9	duration, protocol_type, service, flag, src_bytes, dst_bytes
Content features	13	num_failed_logins, logged_in, num_compromised, root_shell
Time-window features	9	count, srv_count, serror_rate, rerror_rate, same_srv_rate
Host-based features	10	dst_host_count, dst_host_srv_count, dst_host_serror_rate
Post one-hot encoding	122	Expanded from 41 raw features (3 categorical → binary dummies)

3.4 Preprocessing pipeline

Among the 41 input attributes, protocol_type, service, and flag are nominal variables. They are one-hot encoded so that no artificial ordinal relationship is imposed between categories, producing 122 post-encoding input dimensions in the implementation. The difficulty-level field is excluded because it is a benchmark annotation rather than an observable traffic attribute intended for prediction.

Only training-set statistics are used to standardize numerical features: $x' = (x - \mu_{\text{train}}) / \sigma_{\text{train}}$. The fitted scaler is then applied without modification to the validation set, the internal test set, and the official KDDTest+ set. This separation prevents test information from influencing feature transformation. Binary labels are encoded consistently across the pipeline, with attack traffic defined as the positive detection target for precision, recall, F1-score, and threshold-dependent confusion matrices.

For the training labels, class weights are calculated as $w_c = n_{\text{samples}} / (n_{\text{classes}} \times n_c)$, where n_{samples} denotes the total number of training samples, and n_c denotes the number of samples in class c . The resulting weights are approximately 1.0743 for attack traffic and 0.9353 for normal traffic. Class weighting modifies each class's contribution to the loss without creating

additional training instances, which is relevant because class-imbalance treatment can substantially affect minority-attack detection [7, 16].

3.5 Data partitioning

KDDTrain+ is divided in a stratified manner into training, validation, and internal hold-out subsets using nominal proportions of 70/15/15. The internal hold-out is not used for model fitting or hyperparameter tuning, while the validation subset is used by callbacks for model selection. The official KDDTest+ set is never included in this partitioning process and serves as the primary measure of generalization beyond the KDDTrain+ distribution.

3.6 Class imbalance treatment

Class imbalance may cause a classifier to become biased towards the more common class, and may make accuracy itself a limited measure of quality of detection. Recently, this issue has been tackled by resampling, cost-sensitive learning, ensemble techniques, and algorithm-level weighting [7, 16]. The imbalance is moderate in the present binary KDDTrain+ formulation, and class weighting is kept without any synthetic samples to reduce frequency-driven bias.

Therefore, in the principal training configuration, class weighting is used. The weights are directly applied to the loss function, such that errors from the less frequent class have more impact in the optimization process. A single-seed comparison with SMOTE is reported later as an exploratory, single-seed diagnostic and is not repeated across multiple seeds, so it is not used to make a general superiority claim for class weighting.

3.7 Implementation environment

All experiments were performed in Python 3.9 using TensorFlow 2.10 (Keras), scikit-learn 1.0.2 for preprocessing and evaluation, and pandas 1.3.5 for data manipulation. The Adam optimizer was used with the parameters listed in Table 3, a batch size of 256, and early stopping based on validation loss. All computations were performed in a CPU-only environment with an Intel Core i7 processor and 16 GB of

RAM. Measured training and inference times are implementation-specific and are not interpreted as evidence of production throughput or deployability.

Reproducibility controls included explicit NumPy and TensorFlow seeds for the reference run, a fixed random state for the reference data split, training-only fitting of the StandardScaler, and checkpointing of the model weights associated with the lowest validation loss. The complete training procedure was repeated using five independent seeds, and the mean and standard deviation of the KDDTest+ metrics were calculated so that the primary official-test results reflect training variability rather than a single initialization.

Table 3. Training hyperparameter configuration

Hyperparameter	Value / Configuration
Optimizer	Adam ($\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-7}$)
Initial learning rate	$\eta = 0.001$
Learning rate reduction factor	0.5 (patience = 5, min $\eta = 10^{-5}$)
Loss function	Binary cross-entropy
Batch size	256 samples
Max epochs	100 (early stopping patience = 10)
L2 penalty (λ)	0.001 (all Dense layers)
Dropout rate	0.4 (Layer 1), 0.3 (Layers 2–3)
Class weights	Attack: 1.0743, Normal: 0.9353
Decision threshold	0.5 (default); range analysis [0, 1]
Random seed	42 (NumPy + TensorFlow)

4. REGULARIZED MULTI-LAYER PERCEPTRON CONFIGURATION

4.1 Network architecture

The evaluated classifier is a feed-forward MLP with four hidden Dense layers containing 256, 128, 64, and 32 units, respectively, followed by a single sigmoid output. The overall network architecture, including the layer configuration and output layer, is illustrated in Figure 1. The layer widths decrease in a funnel-shaped topology, and regularization is applied as shown in Table 4. The topology and regularization scheme are not presented as a new IDS architecture; rather, they combine established neural-network components in a reproducible baseline.

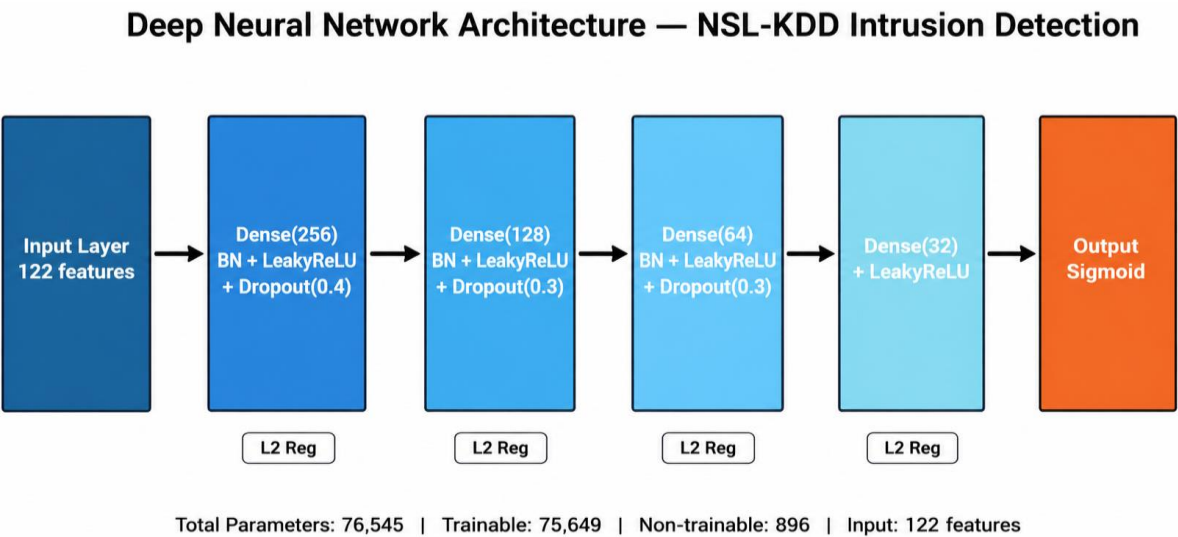


Figure 1. Regularized multi-layer perceptron (MLP) architecture used in the NSL-KDD experiments
Note: Total parameters: 76,545.

Table 4. Regularized multi-layer perceptron (MLP) architecture

#	Type	Units	Activation	Regularization
1	Dense	256	LeakyReLU	L2(0.001)+BN+Drop(0.4)
2	Dense	128	LeakyReLU	L2(0.001)+BN+Drop(0.3)
3	Dense	64	LeakyReLU	L2(0.001)+BN+Drop(0.3)
4	Dense	32	LeakyReLU	L2(0.001)
5	Dense (Out)	1	Sigmoid	—
Total Params:		76,545	Trainable: 75,649	

4.2 Activation functions

LeakyReLU with a negative slope $\alpha = 0.3$ is used in the hidden layers. The activation is defined as:

$$f(x) = x, \text{ if } x \geq 0; \alpha x, \text{ if } x < 0 \quad (1)$$

Unlike a standard ReLU, which returns zero for negative pre-activations, LeakyReLU retains a small negative slope and therefore preserves a nonzero gradient in that region. The output layer uses a sigmoid function:

$$\sigma(z) = 1/(1 + e^{(-z)}) \quad (2)$$

The sigmoid maps the final hidden representation to a scalar class probability used for binary decision making. Threshold-dependent attack metrics are then calculated using the attack class as the positive detection target.

4.3 Batch normalization

BN layers [20] are applied after Dense transformations and before the nonlinear activation. For a mini-batch $B = \{x_1, \dots, x_m\}$, the normalized activation is:

$$\hat{x}_i = (x_i - \mu_B) / \sqrt{(\sigma^2 B + \varepsilon)} \quad (3)$$

Here, μ_B and σ^2_B denote the mini-batch mean and variance, respectively, and ε is a small constant used for numerical stability. BN then learns scale and shift parameters for the normalized activation. In this study, BN is treated as an established optimization and regularization component; its contribution to KDDTest+ performance is not inferred theoretically but is examined only in the exploratory ablation analysis.

4.4 Regularization and training

L2 weight decay with $\lambda = 0.001$ is applied to Dense-layer weights, and dropout rates of 0.4, 0.3, and 0.3 are applied after the first three hidden stages. Early stopping with a patience of 10 epochs restores the best validation-loss checkpoint, while ReduceLROnPlateau decreases the learning rate after five epochs without improvement. These settings were selected during model development and are reported to make the baseline reproducible; their individual effects are not claimed to be statistically confirmed by the single-seed ablation.

The complete training configuration is summarized in Table 3. Optimization uses Adam and binary cross-entropy:

$$L = -[y \cdot \log(\hat{p}) + (1 - y) \cdot \log(1 - \hat{p})] \quad (4)$$

where, $y \in \{0, 1\}$ is the ground-truth binary label and $\hat{p} \in (0, 1)$ is the predicted probability. ReduceLROnPlateau halves the learning rate when validation loss does not improve for five consecutive epochs, with a minimum learning rate of $\eta_{\min} = 10^{-5}$.

5. EXPERIMENTAL RESULTS AND DISCUSSION

5.1 Training dynamics on KDDTrain+-derived partitions

Training and validation accuracy and binary cross-entropy loss for the reference run are shown in Figure 2. The curves converge rapidly and remain close after the initial epochs, indicating stable fitting on the KDDTrain+-derived partitions. These curves are useful for assessing within-distribution training behavior, but they do not demonstrate that the classifier will generalize to the different KDDTest+ distribution.

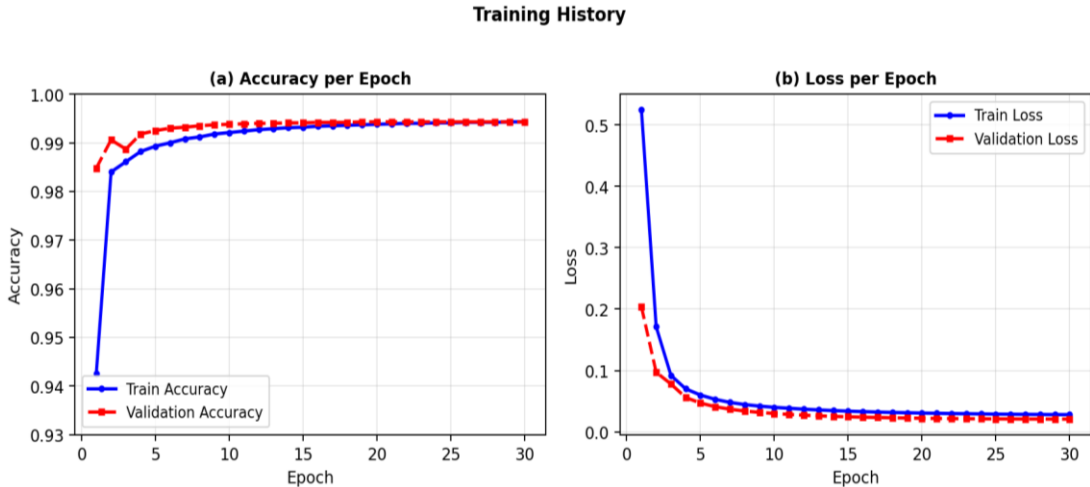


Figure 2. Training history for the reference run: (a) accuracy per epoch, (b) binary cross-entropy loss
Note: The curves describe within-distribution fitting behavior only.

The associated checkpoint with the minimum validation loss was restored at early stopping. The small separation between training and validation curves implies that there is little divergence between the curves for this run; however, the significant difference between training and official-test performance seen in Section 5.2 warns of the dangers of using internal learning curves as a proxy for performance on distribution-shift evaluation.

5.2 Primary generalization results on the official KDDTest+ set

The principal result of this study is performance on the untouched official KDDTest+ set across five independent random seeds. Mean accuracy was $78.14\% \pm 0.46\%$, precision was $92.37\% \pm 0.32\%$, recall was $67.15\% \pm 0.81\%$, F1-score was $77.76\% \pm 0.57\%$, and area under the receiver operating characteristic curve (AUC-ROC) was 0.8767 ± 0.0162 . Table 5 places these values beside the internal hold-out results to make the generalization gap explicit. The most consequential result is the recall of 67.15%: under the official distribution shift, approximately one third of attack records are not detected at the evaluated operating point, despite much stronger precision.

Table 5. Official KDDTest+ generalization versus the internal KDDTrain+-derived hold-out

Metric	Official Mean	Standard Deviation	Internal	Generalization Note
Accuracy	78.14%	0.46%	99.43%	-21.29 pp
Precision	92.37%	0.32%	99.45%	Relatively stable
Recall	67.15%	0.81%	99.50%	-32.35 pp; missed attacks
F1-Score	77.76%	0.57%	99.47%	Marked decrease
AUC-ROC	0.8767	0.0162	0.9994	Lower ranking ability
Recall Gap	-32.35 pp	—	Reference	Primary limitation

Note: AUC-ROC = area under the receiver operating characteristic curve.

5.3 Secondary within-distribution performance

The internal hold-out, which is derived from KDDTrain+, produced 99.43% accuracy, 99.45% precision, 99.50% recall, 99.47% F1-score, and an AUC-ROC of 0.9994. These values indicate strong discrimination within the same general source distribution as the training data, but they are not interpreted as measures of the model's generalization performance. Relative to the five-seed KDDTest+ results, accuracy decreases by 21.29 percentage points and recall decreases by 32.35 percentage points. The magnitude of these differences demonstrates that the internal-split evaluation substantially overestimates performance under the official benchmark distribution shift.

Variation across random seeds does not appear to explain the observed generalization gap because the between-seed variability is small relative to the performance difference between the internal hold-out and KDDTest+. This indicates that the gap is not primarily attributable to random initialization. Under this benchmark shift, recall decreases substantially while precision remains comparatively high, indicating that missed attacks are a more prominent failure mode than false alarms.

Therefore, the official KDDTest+ values, rather than the

99.43% internal accuracy, are used as the primary basis for the interpretation, limitations, and conclusion of this manuscript.

This separation also limits deployment inferences even when a model performs strongly on KDDTrain+. If a substantial proportion of attacks is missed under a different traffic distribution, strong internal performance cannot establish operational effectiveness. The present results therefore support a methodological conclusion about benchmark generalization, not a conclusion about effective real-world operation.

5.4 Secondary per-class analysis on the internal hold-out

The precision, recall, and F1-score values for the KDDTrain+-derived internal hold-out are presented in Table 6. The attack and normal classes show similar performance within this distribution. These values describe internal behavior only and should be interpreted alongside, rather than in place of, the lower official KDDTest+ recall reported in Section 5.2.

Table 6. Secondary per-class report on the internal hold-out (n = 18,896)

Class	Precision	Recall	F1-Score	Support
Attack	0.9941	0.9942	0.9942	10,101
Normal	0.9935	0.9936	0.9936	8,795
Macro Avg	0.9938	0.9939	0.9939	18,896
Weighted Avg	0.9944	0.9943	0.9943	18,896

5.5 Internal hold-out confusion matrix

The confusion matrix in Table 7 and Figure 3 summarizes the reference internal hold-out at the default threshold. Among the reported samples, 8,739 normal records and 10,050 attack records were classified correctly, while 56 normal records were flagged as attacks and 51 attack records were missed. The resulting 107 errors (0.566% of the internal hold-out) characterize only the KDDTrain+-derived distribution and must not be extrapolated to KDDTest+.

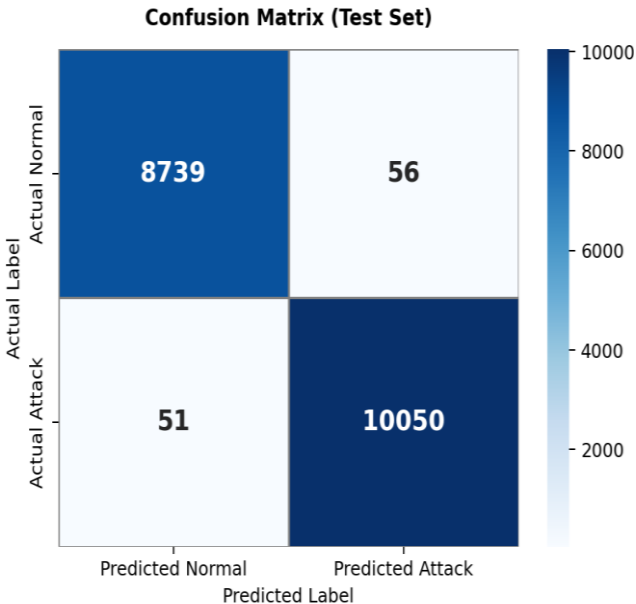


Figure 3. Internal hold-out confusion matrix
Note: TN = 8,739; FP = 56; FN = 51; TP = 10,050; total errors = 107 (0.566%).

Table 7. Confusion matrix on the internal hold-out

	Predicted Normal	Predicted Attack
Actual Normal	8,739 TN (99.36%)	56 FP (0.64%)
Actual Attack	51 FN (0.50%)	10,050 TP (99.50%)

Note: TP = True Positive, TN = True Negative, FP = False Positive, FN = False Negative.

The high internal accuracy reflects the small numbers of False Positives and False Negatives within the KDDTrain+-derived hold-out, but it does not resolve the generalization issue. Consequently, the subsequent discussion emphasizes official-test recall rather than the internal confusion matrix because KDDTest+ reveals a substantially greater missed-attack burden.

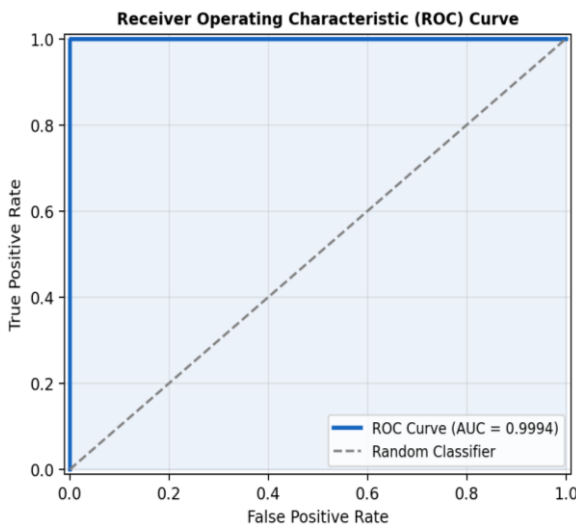
5.6 Internal receiver operating characteristic diagnostic

Figure 4 shows the ROC curve for the internal hold-out. It achieves an AUC-ROC of 0.9994 for records sampled from the KDDTrain+-derived distribution, indicating strong ranking discrimination within this distribution. This value should not be interpreted as evidence of equivalent discrimination under distribution shift; across five KDDTest+ seeds, the mean AUC-ROC is 0.8767 ± 0.0162 .

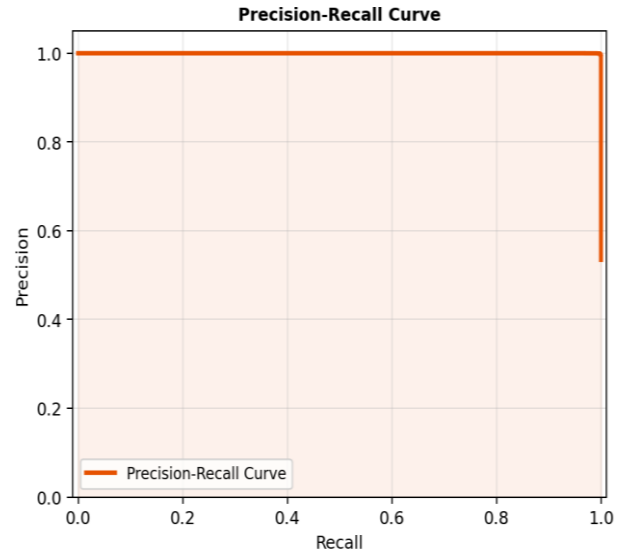
The difference between the two AUC values further supports the primary evaluation result: discrimination on the official test set is substantially lower than within the KDDTrain+-derived hold-out. The internal ROC curve is therefore used only as a diagnostic of within-distribution fit and not as evidence of general detection capability.

5.7 Internal precision-recall diagnostic

Figure 5 shows the precision-recall curve for the internal hold-out. The high area under the precision-recall curve (AUPRC) reflects separation within the KDDTrain+-derived distribution, but it should not be interpreted as evidence of equivalent precision-recall behavior on KDDTest+. On the official test set, mean precision remains high at 92.37%, whereas mean recall falls to 67.15%, revealing a materially different operating trade-off.

**Figure 4.** Receiver operating characteristic (ROC) curve on the internal hold-out (AUC-ROC = 0.9994)

Note: This is a within-distribution diagnostic; the five-seed official KDDTest+ means AUC-ROC is 0.8767.

**Figure 5.** Precision-recall curve on the internal hold-out
Note: The curve characterizes within-distribution performance only.

This difference is especially important for intrusion detection because a high precision value can coexist with an operationally important missed-attack rate. For that reason, the generalization analysis emphasizes recall and F1-score in addition to accuracy and AUC.

5.8 Internal prediction-score distribution

Figure 6 shows the distribution of reference-run prediction scores on the internal hold-out. Most internal records are concentrated near the extremes of the score range, producing limited overlap around the 0.5 decision threshold and explaining the small number of internal errors. This concentration is a property of the KDDTrain+-derived hold-out and should not be interpreted as evidence that scores are similarly separated for attack patterns represented differently in KDDTest+.

5.9 Decision-threshold sensitivity on official KDDTest+

Threshold behavior is interpreted primarily using KDDTest+ because it is more directly related to the generalization question examined in this manuscript. Figure 7 is retained as a secondary illustration of the internal hold-out, whereas the substantive evidence regarding operating-point sensitivity is provided by the official-test threshold sweep.

For the reference KDDTest+ run (seed 42), lowering the threshold to $t = 0.1$ produced 91.9% precision, 73.3% recall, and 81.6% F1-score; the default $t = 0.5$ produced 92.5% precision, 68.2% recall, and 78.5% F1-score; and $t = 0.9$ produced 93.0% precision, 62.4% recall, and 74.7% F1-score. Overall, precision changed only slightly across the evaluated thresholds, whereas recall decreased steadily. Thus, increasing the threshold can reduce false-positive rates but also increases the number of missed attacks. Across the evaluated KDDTest+ thresholds, no operating point eliminated the recall deficit associated with the distribution shift.

These results indicate that the distribution-shift problem cannot be resolved through threshold selection alone. Lower thresholds provide higher attack sensitivity at the cost of more False Positives, whereas higher thresholds slightly increase precision and reduce False Positives but further decrease attack sensitivity.

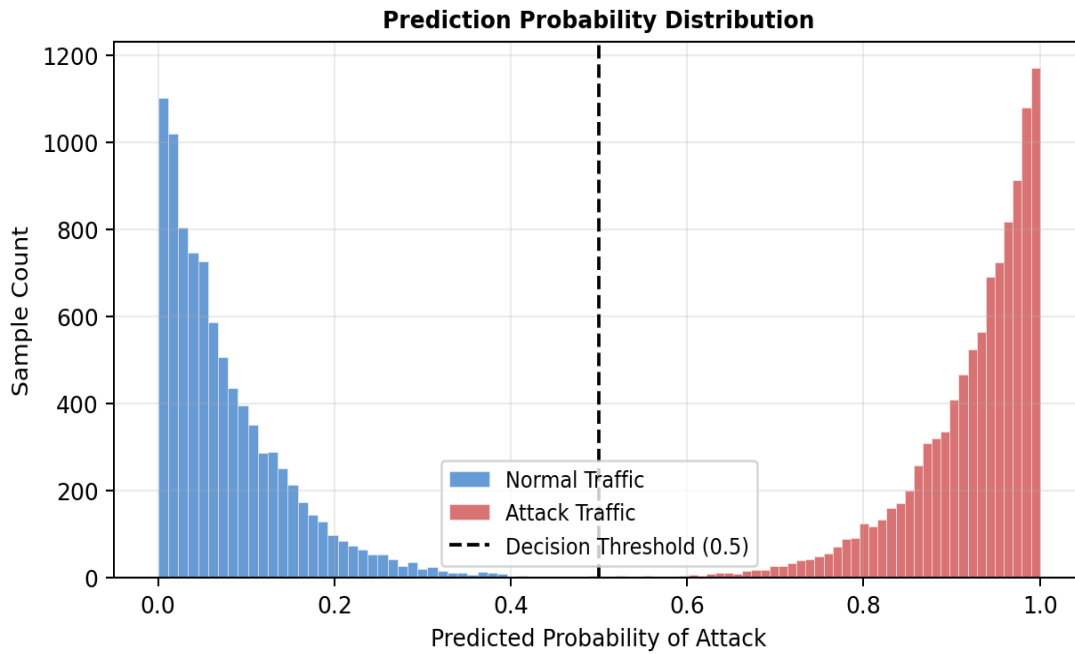


Figure 6. Prediction-score distributions for normal and attack records on the internal hold-out
 Note: Limited overlap explains the strong within-distribution metrics but does not establish out-of-distribution generalization.

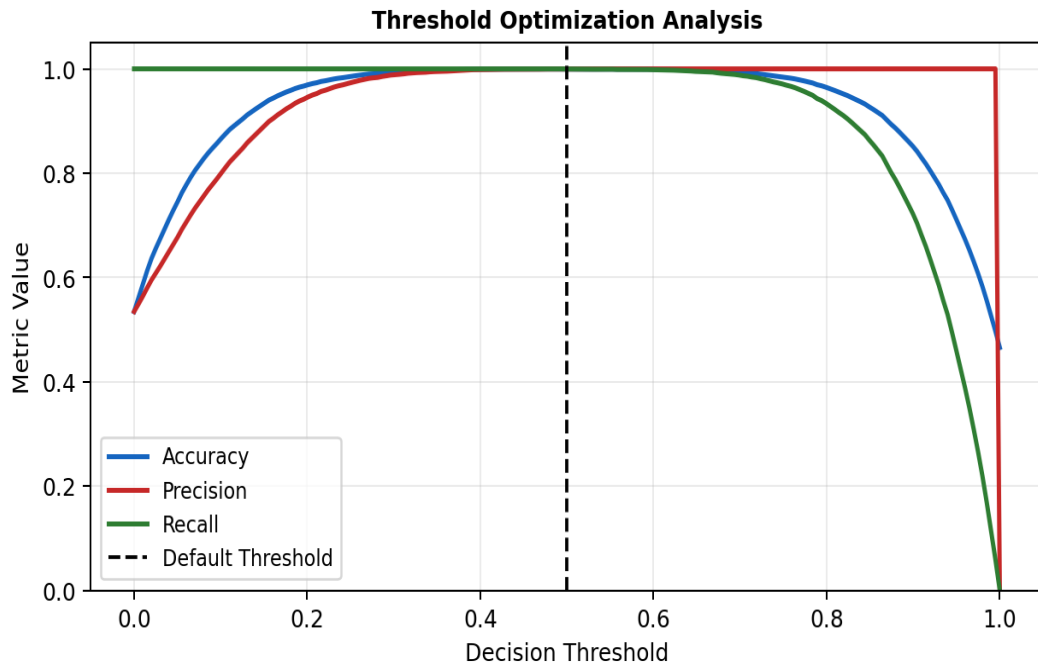


Figure 7. Threshold sensitivity on the internal hold-out, retained as a within-distribution diagnostic
 Note: Official KDDTest+ threshold behavior is summarized in Section 5.9.

Table 8. Protocol-aware context from recent intrusion-detection studies

Study	Dataset	Method	Focus	Context
Kunang et al. [3]	NSL-KDD; CICIDS2018	DAE-DNN	HPO	Multi-dataset
Gupta et al. [7]	NSL-KDD; CICIDS2017	CSDNN + ensemble	Imbalance	Multi-dataset
Hore et al. [9]	Public NIDS data	Sequential deep learning	Out-of-distribution robustness	Robustness-centered
Sarhan et al. [13]	Multiple environments	Feature evaluation	Generalizability	Cross-environment
This study	Official KDDTest+	Regularized MLP	Generalization	Accuracy: 78.14%; Recall: 67.15%

Note: DAE = Denoising Autoencoder, DNN = Deep Neural Network, MLP = multi-layer perceptron, HPO = Hyperparameter Optimization, NIDS = Network intrusion detection system.

Accordingly, no production operating threshold is recommended on the basis of these results. Any operational threshold would need to be calibrated using current, representative traffic and explicit cost functions for missed attacks and analyst alert burden.

The threshold analysis therefore demonstrates a genuine precision-recall trade-off under KDDTest+, reinforcing the need to interpret performance in relation to the intended operating objective.

5.10 Recent literature and protocol-aware comparison

Table 8 places the present work within recent IDS research without ranking heterogeneous studies solely by accuracy. The selected studies illustrate current directions in hyperparameter-optimized deep learning, multi-dataset validation, cost-sensitive learning, self-supervised anomaly detection, generalizability analysis, and robustness to out-of-distribution or adversarial traffic [3, 5, 7-9, 13-15]. Because the datasets, class definitions, and evaluation partitions differ, their reported scores are not directly comparable with the present KDDTest+ result. The appropriate comparison is methodological: recent work increasingly emphasizes robustness and modern traffic, while this study quantifies how a conventional regularized MLP degrades when NSL-KDD evaluation moves from an internal split to the official test distribution.

5.11 Computational characterization

The evaluated MLP contains approximately 76,545 parameters. In the reported CPU-only environment, batched inference over KDDTest+ required a mean of 23.6 ms per 1,000 preprocessed records. This measurement indicates that the neural-network forward pass is computationally modest in the experimental setup. It does not establish end-to-end production throughput because packet capture, feature extraction, data transfer, concurrent workloads, security orchestration, and hardware variability were not included in the timing experiment. The latency result is therefore reported as a computational characterization, not as a deployment claim.

5.12 Internal error inspection

There were 107 errors in the reference internal hold-out. False-positive and false-negative cases were concentrated around the decision boundary, while inspection of the prediction scores showed that correctly classified cases were distributed more broadly. This observation helps characterize the internal confusion matrix but is of secondary importance because the official KDDTest+ evaluation exhibits a substantially higher missed-attack rate.

This internal error inspection does not imply that any specific attack family can be detected reliably or that the model's errors are limited to a small set of ambiguous examples. Attack-family-level evaluation should be conducted on the official test set and validated using contemporary datasets to provide a more informative failure analysis.

5.13 Generalization and distribution-shift analysis

Two questions must be distinguished: does the model fit the KDDTrain+-derived partitions consistently, and does the

resulting decision function generalize to the official KDDTest+ distribution? The internal curves and metrics indicate consistent within-distribution fitting, but they are insufficient to establish external generalization. Across five seeds, official KDDTest+ accuracy is $78.14\% \pm 0.46\%$, and AUC-ROC is 0.8767 ± 0.0162 , compared with 99.43% accuracy and 0.9994 AUC-ROC on the internal hold-out. Most importantly, attack recall decreases from 99.50% internally to $67.15\% \pm 0.81\%$ on KDDTest+. This reduction of approximately 32 percentage points is the clearest manifestation of distribution shift and represents the principal empirical finding of the manuscript, consistent with broader concerns regarding the generalizability of ML-based NIDSs [9, 13].

5.14 Exploratory single-seed ablation of regularization components

The ablation analysis is explicitly exploratory because each configuration was evaluated in a single run. Under this limitation, the full L2 + dropout + BN configuration achieved 78.7% accuracy and 0.893 AUC on KDDTest+; removing L2 yielded 78.6% and 0.896; removing dropout yielded 81.2% and 0.877; removing BN yielded 78.9% and 0.879; and disabling all three regularizers yielded 80.4% and 0.910. These single-seed differences do not establish causal contributions, stable component rankings, or statistical superiority. They indicate only that the relative ordering can change under KDDTest+ distribution shift and therefore motivate a future multi-seed factorial ablation. No conclusion in this manuscript attributes the five-seed generalization performance to any individual regularization component.

5.15 Exploratory class-imbalance comparison

The comparison among no imbalance treatment, class weighting, and SMOTE was also conducted as a single-seed diagnostic and is therefore exploratory. In that run, the respective KDDTest+ accuracy/AUC values were 78.3%/0.865, 80.6%/0.889, and 78.4%/0.873. Although class weighting produced the highest accuracy and AUC in this particular run, the experiment does not support a general claim that class weighting is superior to SMOTE. Multiple random seeds and, ideally, contemporary datasets are required before drawing a stable conclusion about the imbalance-handling strategy.

5.16 Practical interpretation and benchmark limitations

Claims of advanced detection capability or production readiness are not warranted on the basis of the official KDDTest+ results. The mean recall is 67.15%, indicating that a substantial proportion of attacks remains undetected under the benchmark distribution shift. Therefore, the internal scores should not be used for production false-negative-rate estimation, security-level certification, or capacity planning. Inference latency was measured only in a controlled environment and reflects the model's forward pass rather than end-to-end operational performance.

NSL-KDD is also a dated benchmark that cannot represent contemporary encrypted traffic flows, application-layer behavior, current threat morphologies, or organization-specific network dynamics. Validation on contemporary datasets, such as CICIDS-2017 [11] and UNSW-NB15 [12],

using repeated seeds, cross-dataset tests, attack-family analysis, and adversarial or out-of-distribution evaluation [9, 13, 15] is required to support stronger deployment conclusions. Until such validation is completed, the model should be regarded as a reproducible baseline for studying generalization rather than as a validated operational IDS.

6. CONCLUSION

This study evaluated a regularized MLP baseline for binary network intrusion detection, with the primary conclusions based on the official KDDTest+ split rather than the more favorable internal hold-out split. The results demonstrate that the model maintains relatively strong precision but exhibits substantially lower recall when evaluated on the official test distribution. This finding highlights the difficulty of detecting a considerable proportion of attacks beyond the distribution derived from KDDTrain+ and emphasizes the importance of evaluating intrusion detection models under conditions that better reflect distributional differences between training and unseen data.

The internal hold-out accuracy of 99.43% remains useful only as a within-distribution diagnostic. The reductions of 21.29 percentage points in accuracy and approximately 32.35 percentage points in recall relative to the official test results show that strong internal performance does not necessarily imply strong generalization. The precision-recall trade-off on KDDTest+ can be adjusted by changing the decision threshold, but the generalization gap cannot be eliminated through threshold selection alone.

The study is therefore primarily evaluative rather than architectural. It combines well-known regularization components in a reproducible MLP pipeline and, crucially, places the contrast between internal and official test performance at the center of the discussion through five-seed testing, threshold analysis, and cautious interpretation. The single-seed ablation and imbalance-handling comparisons are exploratory and are not intended to establish independent component effects or the superiority of any strategy.

Several limitations constrain these findings. NSL-KDD is an outdated benchmark, the target is binary rather than attack-family-specific, and the model has not been evaluated on contemporary traffic datasets. Future work should include multi-seed ablation studies, evaluation on CICIDS-2017 and UNSW-NB15, attack-family-level recall analysis, robustness testing against adversarial perturbations, and calibration using realistic alert-cost functions.

Accordingly, no production-deployment claim is made. The reported latency indicates only that the model's forward pass is computationally modest in the tested environment; end-to-end performance must still be validated using contemporary traffic while accounting for feature-extraction overhead, workload variability, and security-orchestration constraints. The evidence presented here supports the conclusion that the regularized MLP provides a reproducible baseline for studying benchmark generalization and demonstrates the importance of official or distribution-shift evaluation rather than reliance on extremely high internal-split accuracy.

ACKNOWLEDGMENT

The author would like to thank the research community for

making benchmark datasets and reproducible evaluation tools available for comparative analysis in network security.

REFERENCES

- [1] Abdulganiyu, O.H., Ait 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>
- [2] Ali, M.L., Thakur, K., Schmeelk, S., Debello, J., Dragos, D. (2025). Deep learning vs. machine learning for intrusion detection in computer networks: A comparative study. *Applied Sciences*, 15(4): 1903. <https://doi.org/10.3390/app15041903>
- [3] Kunang, Y.N., Nurmainsi, S., Stiawan, D., Suprpto, B.Y. (2021). Attack classification of an intrusion detection system using deep learning and hyperparameter optimization. *Journal of Information Security and Applications*, 58: 102804. <https://doi.org/10.1016/j.jisa.2021.102804>
- [4] Duan, L.T., Han, D.Z., Tian, Q.T. (2019). Design of intrusion detection system based on improved ABC_elite and BP neural networks. *Computer Science and Information Systems*, 16(3): 773-795. <https://doi.org/10.2298/CSIS181001026D>
- [5] 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>
- [6] Staudemeyer, R.C. (2015). Applying long short-term memory recurrent neural networks to intrusion detection. *South African Computer Journal*, 56(1): 136-154. <https://hdl.handle.net/10520/EJC173450>
- [7] 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>
- [8] Caville, E., Lo, W.W., Layeghy, S., Portmann, M. (2022). Anomal-E: A self-supervised network intrusion detection system based on graph neural networks. *Knowledge-Based Systems*, 258: 110030. <https://doi.org/10.1016/j.knosys.2022.110030>
- [9] Hore, S., Ghadermazi, J., Shah, A., Bastian, N.D. (2024). A sequential deep learning framework for a robust and resilient network intrusion detection system. *Computers & Security*, 144: 103928. <https://doi.org/10.1016/j.cose.2024.103928>
- [10] Tavallae, M., Bagheri, E., Lu, W., Ghorbani, A.A. (2009). A detailed analysis of the KDD CUP 99 data set. In *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, Ottawa, ON, Canada, pp. 1-6. <https://doi.org/10.1109/cisda.2009.5356528>
- [11] Sharafaldin, I., Habibi Lashkari, A., Ghorbani, A.A. (2018). Toward generating a new intrusion detection dataset and intrusion traffic characterization. In *Proceedings of the 4th International Conference on Information Systems Security and Privacy*, Madeira, Portugal, pp. 108-116. <https://doi.org/10.5220/0006639801080116>
- [12] Moustafa, N., Slay, J. (2015). UNSW-NB15: A

- comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In 2015 Military Communications and Information Systems Conference (MilCIS), Canberra, ACT, Australia, pp. 1-6. <https://doi.org/10.1109/milcis.2015.7348942>
- [13] Sarhan, M., Layeghy, S., Portmann, M. (2022). Evaluating standard feature sets towards increased generalisability and explainability of ML-based network intrusion detection. *Big Data Research*, 30: 100359. <https://doi.org/10.1016/j.bdr.2022.100359>
- [14] Hashemi, M.J., Keller, E. (2020). Enhancing robustness against adversarial examples in network intrusion detection systems. In 2020 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), Leganes, Spain, pp. 37-43. <https://doi.org/10.1109/NFV-SDN50289.2020.9289869>
- [15] Farhan, M., Waheed Ud Din, H., Ullah, S., et al. (2025). Network-based intrusion detection using deep learning technique. *Scientific Reports*, 15: 25550. <https://doi.org/10.1038/s41598-025-08770-0>
- [16] Zhou, Y.Y., Cheng, G., Jiang, S.Q., Dai, M. (2020). 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>
- [17] Yan, B.H., Han, G.D. (2018). Effective feature extraction via stacked sparse autoencoder to improve intrusion detection system. *IEEE Access*, 6: 41238-41248. 115782. <https://doi.org/10.1109/ACCESS.2018.2858277>
- [18] Yilmaz, A.A. (2025). A novel deep learning-based framework with particle swarm optimisation for intrusion detection in computer networks. *PLoS One*, 20(2): e0316253. <https://doi.org/10.1371/journal.pone.0316253>
- [19] Chale, M., Bastian, N.D. (2022). Generating realistic cyber data for training and evaluating machine learning classifiers for network intrusion detection systems. *Expert Systems with Applications*, 207: 117936. <https://doi.org/10.1016/j.eswa.2022.117936>
- [20] Ioffe, S., Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning*, Lille, France, pp. 448-456.

NOMENCLATURE

AUC	Area under the ROC curve, dimensionless
FPR	False Positive Rate, dimensionless
FNR	False Negative Rate, dimensionless
TPR	True Positive Rate (Sensitivity), dimensionless
TNR	True Negative Rate (Specificity), dimensionless
η	Learning rate, dimensionless
λ	L2 regularization penalty coefficient, dimensionless
$\hat{p}$	Predicted probability of attack, dimensionless
α	LeakyReLU negative slope coefficient, dimensionless
TN	True Negatives, correctly classified normal samples
TP	True Positives, correctly detected attack samples
FP	False Positives, normal samples misclassified as attack
FN	False Negatives, attack samples missed by classifier
MLP	Multi-layer perceptron, dimensionless architectural descriptor
BN	Batch normalization, dimensionless regularization operator