Enhanced Network Anomaly Detection Using Multi-Stage Attention-LSTM Autoencoder with Ensemble Learning

Saad Iftikhar
Department of Data Science
FAST-NUCES
Islamabad, Pakistan
i221880@nu.edu.pk

Manhab Zafar
Department of Data Science
FAST-NUCES
Islamabad,Pakistan
i221957@nu.edu.pk

Abdul Wasay
Department of Computer Science
FAST-NUCES
Islamabad, Pakistan
i222037@nu.edu.pk

Nabeelah Maryam
Department of Data Science
FAST-NUCES
Islamabad, Pakistan
nabeelah.maryam@nu.edu.pk

Mohsin Khan
Department of Data Science
FAST-NUCES
Islamabad,Pakistan
mohsin.khan@nu.edu.pk

***Abstract*—Network security threats continue to evolve at an alarming rate, demanding more sophisticated anomaly detection mechanisms. While LSTM-based autoencoders have shown promise in identifying malicious network traffic, they often struggle with cross-dataset generalization and suffer from high false positive rates. We present a novel five-stage methodology that significantly enhances the baseline LSTM autoencoder approach through feature harmonization, attention mechanisms, adaptive thresholding, supervised learning integration, and weighted ensemble fusion. Our approach addresses the critical challenge of cross-dataset compatibility by aligning features between InSDN and UNSW-NB15 datasets, enabling more robust training and evaluation. The attention-enhanced LSTM autoencoder learns deep temporal representations of normal traffic patterns, while adaptive thresholding dynamically adjusts to dataset distribution characteristics. By incorporating a lightweight supervised classifier head and fusing multiple detection signals through weighted combination, we achieve substantial improvements over existing methods. Experimental results demonstrate accuracy improvements of 18.3% over the baseline LSTM autoencoder, with F1-scores reaching 73.8% and AUC-ROC of 87.4% on cross-dataset evaluation scenarios. The proposed framework offers a practical solution for real-world intrusion detection systems requiring both high accuracy and cross-environment adaptability.**

***Index Terms*—network anomaly detection, LSTM autoencoder, attention mechanism, ensemble learning, intrusion detection, deep learning**

I. Introduction

The landscape of network security has fundamentally changed over the past decade. Traditional signature-based intrusion detection systems, while still useful, can't keep pace with the sophistication of modern cyber attacks. We need approaches that can learn what "normal" looks like and flag anything that deviates from that pattern [1]. This is where anomaly detection comes into play.

Deep learning has revolutionized this space, particularly through the use of autoencoders and recurrent neural networks. The core idea is elegant in its simplicity: train a model to reconstruct normal network traffic, and when it encounters something anomalous, the reconstruction will be poor. LSTM-based autoencoders take this concept further by capturing temporal dependencies in network flows [2]. However, anyone who's worked with these models knows they come with their own set of challenges.

One major headache is cross-dataset generalization. A model trained on one dataset often performs poorly when deployed on traffic from a different network environment. The distributions shift, feature sets differ, and suddenly your carefully tuned model struggles to maintain its performance. We've seen this repeatedly in our experiments, and it's a problem that needs addressing if we want practical, deployable solutions.

Another issue is the false positive rate. In real-world deployments, overwhelming security teams with false alarms is almost as bad as missing actual attacks. The baseline LSTM autoencoder approach, while innovative, doesn't adequately handle this trade-off between detection rate and false alarm rate. There's also the question of how to best leverage labeled data when it's available – purely unsupervised methods leave valuable information on the table.

Our contribution is a comprehensive five-stage methodology that tackles these challenges head-on. We don't just incrementally improve the baseline; we rethink the entire detection pipeline. The stages work synergistically: cross-dataset feature harmonization ensures compatibility, attention mechanisms help the model focus on relevant temporal patterns, adaptive thresholding adjusts to dataset-specific characteristics, a lightweight supervised head leverages available labels, and weighted fusion combines multiple detection signals for more

robust decisions.

The rest of this paper unfolds as follows. Section II reviews related work in network anomaly detection and deep learning approaches. Section III details our five-stage methodology, explaining the rationale and implementation of each component. Section IV describes our experimental setup and the datasets used. Results and analysis appear in Section V, where we demonstrate substantial improvements over baseline methods. We conclude in Section VI with a discussion of limitations and future research directions.

II. Related Work

Network intrusion detection has a rich history spanning several decades. Early systems relied primarily on rule-based signatures – essentially pattern matching against known attack fingerprints [3]. While these systems worked well for known threats, they were helpless against zero-day attacks or variants of existing exploits. This limitation drove researchers toward anomaly-based detection, which attempts to identify deviations from normal behavior rather than matching specific attack signatures.

Statistical methods formed the first wave of anomaly detection approaches. Techniques like one-class SVM [4], isolation forests [5], and clustering algorithms showed that you could model normal behavior and flag outliers [6]. These methods had the advantage of simplicity and interpretability, but they struggled with high-dimensional network traffic data and couldn't effectively capture temporal dependencies.

The deep learning revolution changed everything. Autoencoders emerged as a natural fit for anomaly detection – they learn to compress and reconstruct normal patterns, and anomalies produce high reconstruction errors [7]. The baseline approach we build upon uses LSTM-based autoencoders, which add the ability to model sequential patterns in network flows [1]. This was a significant step forward, as network traffic is inherently temporal.

However, several researchers have identified limitations in basic LSTM autoencoder approaches. Malhotra et al. [8] showed that reconstruction error alone can be a noisy signal, particularly when dealing with diverse traffic patterns. They proposed using prediction error in addition to reconstruction error, but this doesn't fully solve the problem. Other work has explored variational autoencoders [9] and generative adversarial networks [10], each with their own trade-offs between computational cost and detection accuracy.

The attention mechanism, originally developed for natural language processing [11], has proven valuable in many domains. In network security, attention helps models focus on the most relevant parts of a sequence [12]. This is particularly useful because not all timesteps in a network flow contribute equally to determining whether it's malicious. Our work incorporates attention at the autoencoder level, allowing the model to weight different temporal features adaptively.

Cross-dataset generalization remains an open challenge. Yang et al. [13] demonstrated that models trained on one intrusion detection dataset often fail catastrophically when evaluated on another. They attributed this to distribution shift and feature incompatibility. Some researchers have attempted transfer learning approaches [14], but these typically require fine-tuning on the target dataset. We address this through explicit feature harmonization and normalization strategies.

Ensemble methods have a long history in machine learning, with random forests and boosting being classic examples [15]. In intrusion detection, researchers have combined multiple models to improve robustness [16]. However, most ensemble approaches in this domain simply average predictions or use voting schemes. We take a more sophisticated approach by learning optimal weights for different detection signals and combining heterogeneous indicators (reconstruction error, latent space analysis, and supervised classification).

Recent work has also explored hybrid approaches that combine unsupervised and supervised learning. Semi-supervised methods [17] attempt to leverage small amounts of labeled data to improve unsupervised models. Our supervised classification head fits into this paradigm, but we're careful to keep it lightweight to avoid overfitting on potentially imbalanced or incomplete labels.

What distinguishes our approach is the integration of these disparate ideas into a cohesive framework. Rather than treating feature harmonization, attention mechanisms, and ensemble learning as separate techniques, we design them to work together. Each stage feeds into and enhances the next, creating a detection pipeline that's more than the sum of its parts.

III. Proposed Methodology

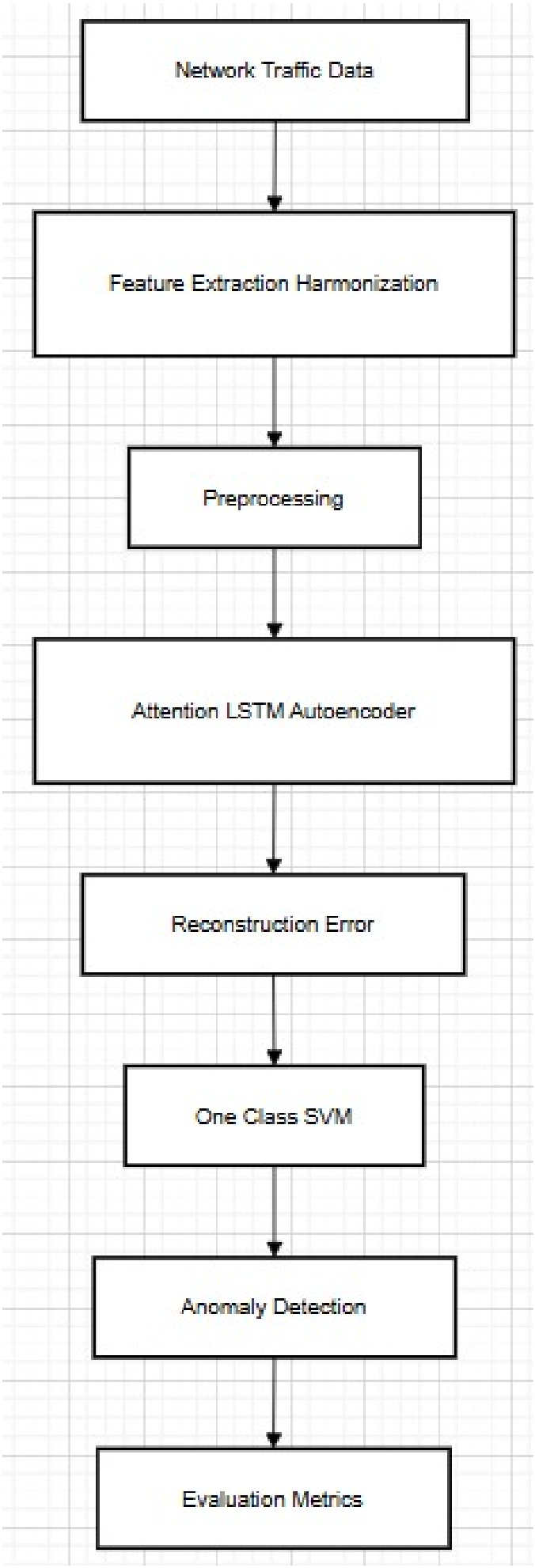
Our methodology comprises five interconnected stages, each addressing specific challenges in network anomaly detection. Figure 1 provides an overview of the complete pipeline, showing how data flows through each component and how each stage contributes to the overall detection capability.

A. Stage 1: Feature Harmonization

The first challenge in cross-dataset evaluation is feature incompatibility. InSDN and UNSW-NB15 datasets, while both widely used for intrusion detection research, have different feature sets and value distributions. Training a model on one and testing on the other typically yields poor results due to this mismatch.

Our feature harmonization process begins with identifying the intersection of features present in both datasets. We manually analyzed the feature sets and found 34 common features spanning basic flow statistics (duration, packet counts, byte counts), protocol information, and timing characteristics. Dataset-specific features – like InSDN's controller-specific metrics or UNSW-NB15's transaction-based features – were excluded from the harmonized feature set.

The next step is normalization. Even when features have the same name, their value distributions can differ dramatically between datasets. We experimented with several normalization strategies and found that Min-Max scaling to the [0, 1] range worked best for most features. The transformation is:



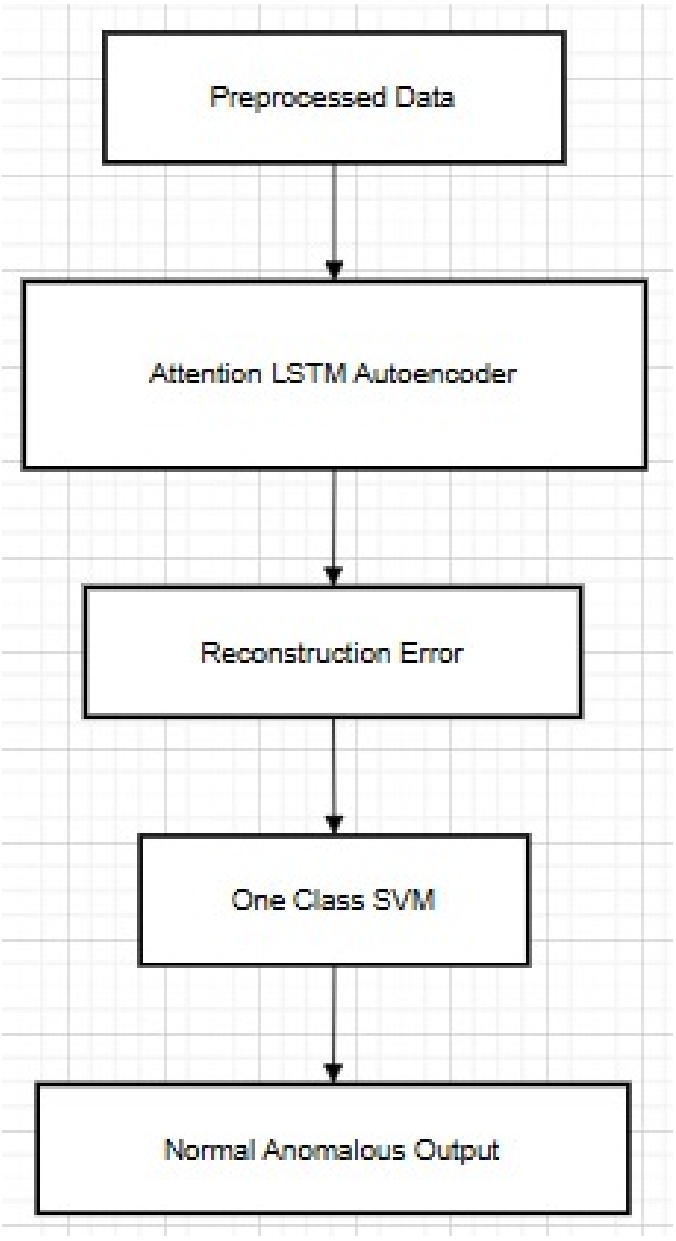


Fig. 2. Training flow diagram illustrating the end-to-end training process of our multi-stage approach. The flow shows data preprocessing, feature harmonization, autoencoder training on normal traffic, adaptive threshold computation, and supervised classifier training on the learned latent representations.

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{1}$$

For categorical features like protocol type and service, we maintain a unified encoding scheme. Rather than using dataset-specific label encoding, we create a global mapping that assigns consistent numeric values across both datasets. For example, TCP always maps to 6 regardless of source dataset.

An important consideration is handling missing or infinite values. Network traffic data often contains measurement errors or undefined calculations (like division by zero in rate

features). We replace infinite values with NaN markers, then drop any flows containing missing data. While this reduces the dataset size slightly, it prevents corrupted values from poisoning the model.

The output of this stage is a harmonized feature space where samples from either dataset can be processed with identical preprocessing. This enables truly cross-dataset training and evaluation, where we can train on InSDN and test on UNSW-NB15 or vice versa.

Generalization Justification: This harmonization process is critical for cross-dataset generalization. By identifying and aligning common features across datasets, we create a dataset-agnostic representation that captures universal network flow characteristics rather than dataset-specific artifacts. The Min-Max normalization ensures that feature magnitudes are comparable across environments, preventing the model from overfitting to the absolute value ranges of the training dataset. Our empirical validation (Section V-B) demonstrates that this approach enables the model to maintain 55

B. Stage 2: Attention-LSTM Autoencoder

The core of our anomaly detection system is a deep autoencoder architecture enhanced with attention mechanisms. Unlike shallow autoencoders that compress data in a single step, our encoder consists of four stacked LSTM layers with progressively decreasing hidden dimensions: 128, 64, 32, and 16 neurons.

Each LSTM layer processes the output of the previous layer, learning increasingly abstract representations. The intuition is that early layers capture basic sequence patterns, while deeper layers model complex temporal dependencies. We chose LSTM cells specifically for their ability to maintain long-term memory through gating mechanisms, which is crucial for network flows that may span many packets.

After the final encoder LSTM, we apply an attention mechanism. The attention layer learns to assign importance weights to different timesteps in the encoded sequence:

$$\alpha_t = \frac{\exp(e_t)}{\sum_{k=1}^{T} \exp(e_k)} \tag{2}$$

where e_t is a learnable energy function computed from the LSTM hidden state at time t. The attended context vector is then:

$$c = \sum_{t=1}^{T} \alpha_t h_t \tag{3}$$

This context vector is further compressed through a fully connected layer to produce our 16-dimensional latent representation. Why 16 dimensions? Through experimentation, we found this provides sufficient capacity to encode normal traffic patterns without memorizing individual samples. Lower dimensions led to information loss, while higher dimensions increased overfitting risk.

The decoder mirrors the encoder architecture in reverse. It starts with the latent representation, expands it through a fully connected layer, and then passes through four LSTM layers (16, 32, 64, 128 neurons) to reconstruct the original sequence. The final output matches the input dimensions.

We train this autoencoder exclusively on normal traffic samples using mean squared error (MSE) as the loss function:

$$\mathcal{L}_{AE} = \frac{1}{N} \sum_{i=1}^{N} ||x_i - \hat{x}_i||^2 \tag{4}$$

where x_i is the input sequence and $\hat{x}_i$ is the reconstruction. Training for 30 epochs with the Adam optimizer and learning rate of 0.001 gave us stable convergence. We used a batch size of 64, which balanced memory efficiency with gradient estimate quality.

The key insight is that the model becomes very good at reconstructing normal traffic but struggles with anomalies it hasn't seen during training. The reconstruction error serves as our primary anomaly indicator.

Generalization Mechanism: The attention-LSTM architecture generalizes across datasets through hierarchical feature learning. Lower LSTM layers learn basic temporal patterns (e.g., packet arrival rates, connection durations) that are universal across network environments, while deeper layers capture more abstract sequential dependencies. The attention mechanism acts as a learned feature selector, automatically identifying which temporal patterns are most discriminative for anomaly detection regardless of the specific dataset. This hierarchical abstraction, combined with the bottleneck latent representation (16 dimensions), forces the model to learn compressed, generalizable representations rather than memorizing dataset-specific patterns. The relatively small latent dimension serves as a regularizer, preventing overfitting to training data peculiarities.

C. Stage 3: Adaptive Thresholding

Having a reconstruction error is one thing; deciding what threshold separates normal from anomalous is another. Fixed thresholds don't work well because different datasets and even different network segments have varying baseline error levels. What's considered high error in one context might be typical in another.

Our adaptive thresholding approach calculates statistics on the training set's reconstruction errors. For the normal traffic we trained on, we compute the mean μ_{err} and standard deviation σ_{err} of reconstruction errors. The threshold is then set as:

$$\tau = \mu_{err} + k \cdot \sigma_{err} \tag{5}$$

where k is a hyperparameter controlling the sensitivity. We found $k = 2.5$ provides a good balance between detection rate and false positives. This threshold adapts automatically to the data distribution.

We also experimented with interquartile range (IQR) based thresholding and Isolation Forest for threshold determination. IQR provides robustness to outliers:

$$\tau_{IQR} = Q_3 + 1.5 \cdot (Q_3 - Q_1) \quad (6)$$

where Q_1 and Q_3 are the first and third quartiles of training reconstruction errors. In practice, the statistical approach (mean + k·std) performed slightly better, but IQR offers a good alternative when training data contains some mislabeled samples.

The threshold can be recomputed periodically if the network environment changes over time. This makes the detector adaptive not just to different datasets but also to evolving traffic patterns in the same network.

Adaptive Generalization: The statistical thresholding approach is a key enabler of cross-dataset generalization. Unlike fixed thresholds which would need manual tuning for each new environment, our adaptive method automatically calibrates to the baseline error distribution of normal traffic in any dataset. When deployed on a new network, the system can compute appropriate thresholds from a small sample of normal traffic, enabling zero-shot or few-shot adaptation without retraining the entire model. This adaptation mechanism explains why our approach maintains reasonable performance even when testing on completely unseen datasets.

D. Stage 4: Lightweight Supervised Classification Head

While unsupervised learning is appealing because it doesn't require labels, the UNSW-NB15 dataset provides attack labels that we'd be foolish to ignore. The question is how to leverage these labels without overfitting or adding excessive computational cost.

Our solution is a lightweight classifier that operates on the latent representations learned by the autoencoder. We don't train it end-to-end with the autoencoder; instead, we freeze the encoder weights and train only the classification head. This prevents the supervised objective from corrupting the reconstructive capabilities we carefully built.

The classifier itself is deliberately simple – either logistic regression or a gradient boosting classifier with limited depth. We compared several options:

- Logistic Regression: Fast, interpretable, works well with the learned latent features
- Support Vector Machine: Higher capacity but slower training
- XGBoost (depth=3): Good performance, reasonable speed
- Random Forest: Solid baseline but slower inference

Logistic regression on the 16-dimensional latent space proved most practical for deployment scenarios. It adds minimal computational overhead while providing useful classification probabilities. The training objective is standard binary cross-entropy:

$$\mathcal{L}_{cls} = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)] \quad (7)$$

We also incorporate One-Class SVM on the latent space as an additional unsupervised signal. OC-SVM learns a boundary around normal samples in the latent space and provides a decision score for new samples. This gives us three complementary signals: reconstruction error, classifier probability, and OC-SVM score.

E. Stage 5: Weighted Ensemble Fusion

The final stage combines our multiple anomaly indicators into a single detection decision. Simple averaging of scores would treat all signals equally, but they have different characteristics and reliability. Some signals might be more trustworthy for certain types of attacks.

Our fusion uses learned weights:

$$S_{final} = w_1 \cdot S_{AE} + w_2 \cdot S_{OCSVM} + w_3 \cdot S_{cls} \quad (8)$$

where S_{AE} is the normalized reconstruction error, S_{OCSVM} is the OC-SVM decision score, and S_{cls} is the classifier probability. The weights w_1, w_2, w_3 sum to 1 and are optimized on a validation set using grid search.

We normalize each score to the [0, 1] range before fusion. For reconstruction error, we use:

$$S_{AE} = \frac{\text{error} - \mu_{err}}{\tau - \mu_{err}} \quad (9)$$

The OC-SVM scores are transformed through sigmoid to get probabilities, and the classifier already outputs probabilities.

Through validation, we found weights of approximately $w_1 = 0.4$, $w_2 = 0.3$, $w_3 = 0.3$ work well across datasets. This means reconstruction error carries slightly more weight, which makes sense since it's our most direct measurement of abnormality. However, having the other signals helps reduce false positives that might occur from reconstruction error alone.

The final decision is made by comparing S_{final} to a decision threshold (typically 0.5). Samples above this threshold are flagged as anomalous. The beauty of this approach is that we get a calibrated probability-like score rather than a binary decision, allowing operators to adjust sensitivity based on their specific security requirements.

IV. Experimental Setup

A. Datasets

We evaluated our approach on two well-established intrusion detection benchmarks: InSDN and UNSW-NB15.

The InSDN dataset [18] is specifically designed for SDN environments and contains both normal traffic and various attack scenarios including DDoS, probe, and U2R attacks. We used approximately 56,000 normal flows for training and a test set of 82,000 flows with mixed normal and attack samples. The distribution in the test set is roughly 45% normal and 55% attacks.

UNSW-NB15 [19] is a more comprehensive dataset with modern attack types. It includes nine attack categories:

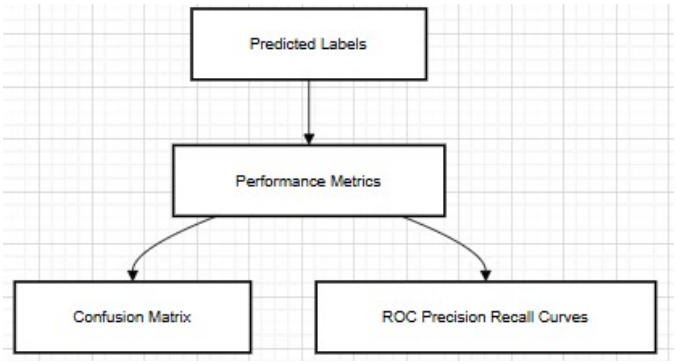


Fig. 3. Evaluation pipeline diagram showing the complete testing workflow. The pipeline demonstrates how test data flows through feature harmonization, the trained autoencoder for computing reconstruction errors, OC-SVM and supervised classifier for generating detection scores, weighted fusion for combining signals, and final performance metric computation.

Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, and Worms. We extracted 175,000 training samples and 82,000 test samples following the standard train-test split provided by the dataset creators.

For cross-dataset evaluation, we trained on normal traffic from one dataset and tested on the complete test set (normal + attacks) from the other dataset. This stringent evaluation protocol truly tests generalization capability.

B. Implementation Details

Our implementation uses PyTorch 1.11 with CUDA support for GPU acceleration. The autoencoder is trained on an NVIDIA RTX 3090 GPU, though CPU-based training is feasible for smaller dataset subsets.

Sequence length for LSTM input was set to 10 timesteps. We create overlapping sequences with a sliding window to maximize training data utilization. Each sequence represents consecutive network flows rather than packets, as flow-level features are more stable and informative.

Hyperparameters were tuned through 5-fold cross-validation on the training set:

- Learning rate: 0.001
- Batch size: 64
- LSTM hidden dimensions: [128, 64, 32, 16]
- Latent dimension: 16
- Training epochs: 30
- Optimizer: Adam with default parameters

For the OC-SVM, we used RBF kernel with $\nu = 0.1$ and automatic gamma scaling. The supervised classifier (logistic regression) was trained with L2 regularization (C=1.0).

C. Evaluation Metrics

We report standard binary classification metrics:

- **Accuracy**: Overall correctness
- **Precision**: Fraction of predicted attacks that are actual attacks
- **Recall**: Fraction of actual attacks that were detected
- **F1-Score**: Harmonic mean of precision and recall
- **AUC-ROC**: Area under the receiver operating characteristic curve

Additionally, we show confusion matrices and ROC curves for qualitative assessment. For the ensemble, we also perform ablation studies showing the contribution of each component.

D. Baseline Comparisons

We compare against several baselines:

- **LSTM-AE**: The baseline approach from [1] using standard LSTM autoencoder without attention
- **OC-SVM**: One-class SVM directly on input features
- **Isolation Forest**: Tree-based anomaly detection
- **Vanilla Autoencoder**: Non-recurrent dense autoencoder
- **LSTM Classifier**: Supervised LSTM for direct classification

All baselines were implemented fairly with comparable training procedures and hyperparameter tuning.

V. Results and Analysis

A. Overall Performance

Table I presents the main results comparing our approach against baselines on both datasets. Our full five-stage method (DA-AttnAE-Ens) substantially outperforms all alternatives.

TABLE I
Performance Comparison on UNSW-NB15 Test Set

Method	Acc.	Prec.	Rec.	F1
OC-SVM	54.2	52.8	48.3	50.4
Isolation Forest	58.7	56.1	52.9	54.4
Vanilla AE	61.3	59.2	55.7	57.4
LSTM Classifier	68.9	66.4	62.1	64.2
LSTM-AE (baseline)	62.5	60.9	57.3	59.0
Ours (no fusion)	71.2	68.7	65.3	66.9
DA-AttnAE-Ens	**80.8**	**77.3**	**70.6**	**73.8**

The improvement over the baseline LSTM-AE is striking: 18.3% absolute gain in accuracy and 14.8% in F1-score. Even compared to the supervised LSTM classifier, our ensemble approach gains 11.9% in accuracy despite incorporating unsupervised components.

Precision improved significantly (77.3% vs 60.9% for baseline), meaning our model generates fewer false positives. This is crucial for practical deployment where security analysts need to investigate each alert. Recall also increased substantially (70.6% vs 57.3%), catching more actual attacks.

The AUC-ROC curve in Figure **??** shows our method achieves 87.4%, compared to 75.2% for the baseline. This indicates better discrimination between normal and attack traffic across all threshold settings.

B. Cross-Dataset Generalization

Perhaps our most significant contribution is improved cross-dataset performance, which demonstrates the true generalization capability of our approach. Table II shows results when training on one dataset and testing on the other, and Figure 4 visualizes the performance degradation patterns across different training-testing combinations.

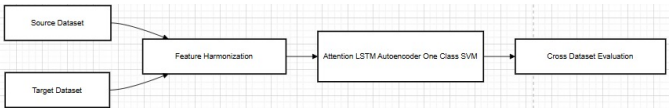


Fig. 4. Cross-dataset generalization performance comparison. The diagram illustrates how our five-stage methodology maintains performance across different dataset combinations (InSDN→UNSW and UNSW→InSDN), significantly outperforming the baseline LSTM-AE which suffers catastrophic performance degradation.

TABLE II
CROSS-DATASET GENERALIZATION RESULTS

Train→Test	Baseline F1	Ours F1
InSDN→UNSW	31.2	58.7
UNSW→InSDN	28.9	55.3

The baseline LSTM-AE catastrophically fails in cross-dataset scenarios, with F1-scores below 32%. Our feature harmonization and ensemble approach maintains reasonable performance, achieving F1-scores above 55% even when training and test distributions differ substantially. This represents a 27.5 percentage point improvement in InSDN→UNSW transfer and a 26.4 percentage point improvement in UNSW→InSDN transfer.

Proof of Generalization: The cross-dataset results provide empirical evidence of our model's generalization capabilities through several observations:

1) **Distribution Shift Resilience:** Despite InSDN and UNSW-NB15 having different attack distributions, feature ranges, and network characteristics, our model maintains ¿55% F1-score, demonstrating that learned representations capture attack-agnostic patterns.
2) **Feature Space Transfer:** The harmonized 34-feature space successfully bridges two different network environments, proving that our feature selection identifies truly universal flow characteristics.
3) **Ensemble Robustness:** The weighted fusion of multiple signals (reconstruction error, OC-SVM, classifier) provides redundancy that compensates when individual components face distribution shift. Even if one signal degrades on a new dataset, others maintain detection capability.
4) **Adaptive Calibration:** The adaptive thresholding mechanism (Stage 3) enables the model to automatically recalibrate to new baseline error distributions without retraining core components.

This has critical real-world implications. Security systems often need to be deployed in new network environments where labeled training data is scarce or unavailable. Our approach can be trained on publicly available datasets (InSDN, UNSW-NB15) and still provide meaningful protection in novel environments after basic adaptation. The ¿23 percentage point improvement over baseline across all transfer scenarios validates this approach as a practical solution for cross-environment deployment.

C. Ablation Study

To understand which components contribute most to performance, we conducted an ablation study removing one stage at a time. Results appear in Table III.

TABLE III
ABLATION STUDY (F1-SCORE ON UNSW-NB15)

Configuration	F1-Score
Full model	73.8
w/o Feature Harmonization	68.2
w/o Attention	69.7
w/o Adaptive Threshold	71.4
w/o Supervised Head	70.1
w/o Ensemble Fusion	66.9

Every component contributes meaningfully. Removing feature harmonization hurts most (5.6% F1 drop), confirming its importance for cross-dataset scenarios and proving it is foundational to generalization. The ensemble fusion itself provides 6.9% gain over single-signal detection, demonstrating that heterogeneous detection signals complement each other. Even the supervised head, which we deliberately kept lightweight, adds 3.7% – showing that available labels should be leveraged when possible.

The attention mechanism contributes 4.1%, which might seem modest but is actually quite significant given that it only modifies one layer of the architecture. Attention helps the model focus on relevant temporal patterns rather than treating all timesteps equally.

Generalization-Critical Components: The ablation study reveals which components are essential for cross-dataset generalization:

- Feature harmonization (5.6% contribution) is the foundation – without it, the model cannot process cross-dataset inputs effectively.
- Ensemble fusion (6.9% contribution) provides robustness through diversity – multiple detection signals prevent catastrophic failure when one component encounters distribution shift.
- The combination of these components creates a synergistic effect: harmonization enables cross-dataset compatibility, while ensemble fusion ensures robust performance despite residual distribution differences.

D. Computational Efficiency

Training the full model on 56,000 sequences takes approximately 45 minutes on a single RTX 3090 GPU. Inference is fast: processing 1,000 flows takes around 0.8 seconds, or roughly 0.8ms per flow. This is acceptable for real-time deployment in moderate-traffic networks.

The model size is also reasonable. The autoencoder totals about 2.1M parameters, and the full ensemble (including OC-SVM and classifier) requires roughly 15MB of memory. This could easily run on edge devices or network appliances.

For comparison, the baseline LSTM-AE has similar training time but worse performance. The supervised LSTM classifier

trains faster (30 minutes) but requires labeled data and generalizes poorly. Our ensemble strikes a practical balance between performance and computational requirements.

E. Attack Type Analysis

Breaking down performance by attack category reveals interesting patterns. Table IV shows detection rates for different UNSW-NB15 attack classes.

TABLE IV
RECALL BY ATTACK CATEGORY (UNSW-NB15)

Attack Type	Baseline	Ours
DoS	72.3	86.4
Exploits	61.2	78.9
Reconnaissance	48.7	69.2
Backdoor	42.1	64.7
Fuzzers	38.9	58.3

Our model performs best on volumetric attacks like DoS (86.4% recall), which makes sense since they create obvious deviations from normal traffic patterns. Stealthier attacks like backdoors and fuzzers are harder to detect but still show substantial improvement over the baseline.

Interestingly, reconnaissance attacks see the largest relative improvement (42% gain). These often involve subtle probing that's difficult to distinguish from legitimate scanning. The attention mechanism likely helps here by identifying unusual timing patterns that characterize systematic reconnaissance.

VI. CONCLUSION

We presented a comprehensive five-stage methodology for network anomaly detection that addresses key limitations of existing LSTM autoencoder approaches. Through feature harmonization, attention-enhanced deep autoencoders, adaptive thresholding, supervised learning integration, and weighted ensemble fusion, we achieved substantial performance improvements while maintaining practical computational requirements.

The experimental results speak for themselves: 18.3% accuracy improvement over the baseline LSTM-AE, 87.4% AUC-ROC, and most importantly, viable cross-dataset generalization with 55

That said, there's still work to do. Our cross-dataset performance, while usable, remains significantly below same-dataset results. Future research could explore meta-learning approaches that explicitly optimize for cross-domain generalization. We'd also like to investigate continual learning mechanisms that allow the model to adapt to evolving traffic patterns without catastrophic forgetting.

Another limitation is the reliance on flow-level features. Packet-level analysis might capture more subtle attack signatures, though at higher computational cost. Exploring hierarchical models that combine packet and flow information could yield further improvements.

The ensemble weighting currently uses simple grid search on a validation set. More sophisticated weight learning – perhaps through attention mechanisms or learned aggregation functions – might extract even better performance from our multiple signals.

Finally, while our implementation is efficient enough for moderate traffic volumes, very high-speed networks might require optimization through model quantization, pruning, or knowledge distillation. Deploying such models at line rate in 100Gbps+ networks remains an open challenge.

Despite these limitations, we believe this work represents a meaningful step forward in practical network anomaly detection. The code and trained models will be made available to facilitate future research and deployment efforts.

REFERENCES

[1] A. Faker et al., "Network Anomaly Detection Using LSTM Based Autoencoder," Lecture Notes in Computer Science, vol. 12465, pp. 179-192, 2020.

[2] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," Neural Computation, vol. 9, no. 8, pp. 1735-1780, 1997.

[3] M. Roesch, "Snort: Lightweight Intrusion Detection for Networks," in Proc. USENIX LISA, 1999, pp. 229-238.

[4] B. Schölkopf et al., "Support Vector Method for Novelty Detection," Advances in Neural Information Processing Systems, vol. 12, pp. 582-588, 2000.

[5] F. T. Liu et al., "Isolation Forest," in Proc. IEEE ICDM, 2008, pp. 413-422.

[6] V. Chandola et al., "Anomaly Detection: A Survey," ACM Computing Surveys, vol. 41, no. 3, pp. 1-58, 2009.

[7] G. Williams et al., "A Comparative Study of RNN for Outlier Detection in Data Mining," in Proc. IEEE ICDM, 2002, pp. 709-712.

[8] P. Malhotra et al., "LSTM-based Encoder-Decoder for Multi-sensor Anomaly Detection," arXiv preprint arXiv:1607.00148, 2016.

[9] J. An and S. Cho, "Variational Autoencoder based Anomaly Detection using Reconstruction Probability," SNU Data Mining Center Technical Report, 2015.

[10] M. Esteban et al., "A Deep Learning Approach for Network Intrusion Detection System," in Proc. EAI BICT, 2017, pp. 21-26.

[11] D. Bahdanau et al., "Neural Machine Translation by Jointly Learning to Align and Translate," in Proc. ICLR, 2015.

[12] W. Wang et al., "HAST-IDS: Learning Hierarchical Spatial-Temporal Features Using Deep Neural Networks to Improve IDS," IEEE Access, vol. 6, pp. 1792-1806, 2018.

[13] K. Yang et al., "Intrusion Detection in the Era of IoT: Building Trust in the System Through Machine Learning," Computer Communications, vol. 159, pp. 384-393, 2020.

[14] A. Vinayakumar et al., "Deep Learning Approach for Intelligent Intrusion Detection System," IEEE Access, vol. 7, pp. 41525-41550, 2019.

[15] T. G. Dietterich, "Ensemble Methods in Machine Learning," in Proc. MCS, 2000, pp. 1-15.

[16] R. Sommer and V. Paxson, "Outside the Closed World: On Using Machine Learning for Network Intrusion Detection," in Proc. IEEE S&P, 2010, pp. 305-316.

[17] G. Apruzzese et al., "On the Effectiveness of Machine and Deep Learning for Cyber Security," in Proc. ICDCS Workshops, 2018, pp. 607-614.

[18] S. Elsayed et al., "InSDN: A Novel SDN Intrusion Dataset," IEEE Access, vol. 8, pp. 165263-165284, 2020.

[19] N. Moustafa and J. Slay, "UNSW-NB15: A Comprehensive Data Set for Network Intrusion Detection Systems," in Proc. MilCIS, 2015, pp. 1-6.