DOI - 10.68648/ijte-q-2026-49-03-012

Toward Adversarial Robust and Edge-Deployable Network Intrusion Detection: An Attention-Augmented Architecture with Data Validation

Narasimha Swamy Kandelli
Research Scholar
Department of CSE
Acharya Nagarjuna University, AP

K. Gangadhara Rao
Professor
Department of CSE
Acharya Nagarjuna University, AP

Simhadri Mallikarjuna Rao
Assistant Professor
School of CSE
VVIT University
Nambur, AP
✉ mallikarjun1254@gmail.com

ABSTRACT

Hybrid deep-learning Network Intrusion Detection Systems (NIDS) achieve strong clean-data accuracy but typically leave adversarial robustness, deployment cost, and quantitatively verified explainability unaddressed. We formulate NIDS design as a constrained multi-objective learning problem spanning detection accuracy, adversarial robustness, latency/resource-bounded deployability, and explanation fidelity, and we design a corresponding attention-augmented architecture with SHAP explainability, FGSM/PGD adversarial training, and knowledge-distilled edge deployment. To validate the overall pipeline before a full-scale implementation on real benchmark traffic, we build, from scratch in NumPy, a lightweight prototype (per-timestep feature projection, a recurrent temporal encoder, single-head self-attention pooling, and a softmax classifier, trained via manually-derived backpropagation-through-time) and evaluate it end-to-end on a controlled synthetic multi-class flow-sequence dataset. The prototype reaches 94.7% test accuracy (macro-F1 0.947, AUC 0.970) on held-out synthetic data, degrades from 94.7% to 80.3% accuracy under an undefended FGSM attack as the perturbation budget grows, and a 9.2x smaller variant matches its accuracy while cutting measured inference latency by 2.5x. These genuine, reproducible findings validate the pipeline and concretely motivate the adversarial-hardening and distillation components of the full design, whose evaluation on real, large-scale NIDS benchmarks is identified as the next phase of this research.

KEYWORDS *: Network intrusion detection, deep learning, recurrent neural network, attention mechanism, adversarial robustness, explainable AI, knowledge distillation, edge deployment, synthetic-data validation.*

INTRODUCTION

Network security increasingly depends on Network Intrusion Detection Systems (NIDS) capable of identifying sophisticated, evolving attacks within high-volume, high-dimensional traffic. Deep learning is now the dominant paradigm for NIDS, and hybrid spatiotemporal architectures—most notably Seq2Seq autoencoders whose LSTM core is replaced by Convolutional LSTM (ConvLSTM) units—have reported detection accuracies above 98% on CIC-IDS2017, CIC-ToN-IoT, and UNSW-NB15 [1].

However, three practical gaps persist in this line of work. First, models are evaluated on clean data only, with no reported robustness to adversarial perturbation. Second, inference latency, throughput, and memory/CPU footprint are not reported, despite being critical for real-time or edge deployment. Third, explainability is typically provided post hoc via LIME [14], which is

known to be less stable and less faithful than gradient- or game-theoretic approaches such as SHAP [13]. Independently, recent audits show that widely used benchmarks such as CIC-IDS2017 contain non-trivial labeling and flow-construction errors [7].

This paper makes two kinds of contribution, and is explicit about which is which. First, we design a complete, multi-objective extension of the hybrid Seq2Seq–ConvLSTM NIDS [1] that treats adversarial robustness, deployability, and explainability as first-class training objectives (Section III). Second, because implementing and training the full architecture on real, large-scale traffic benchmarks is a substantial undertaking, we first validate the overall pipeline—data handling, training, evaluation, adversarial testing, and deployment profiling—end to end using a lightweight, from-scratch prototype and a controlled synthetic dataset (Sections IV–VI). Every number reported in Section VI is a real output of the released code, obtained by actually running it; none are invented, and the prototype's simplifications relative to the full design of Section III are stated explicitly rather than implied.

The main contributions of this paper are:

- A constrained multi-objective problem formulation for NIDS that jointly targets detection accuracy, adversarial robustness, deployability, and explanation fidelity, and a corresponding full architecture design (attention-augmented Seq2Seq–ConvLSTM backbone with SHAP explainability, FGSM/PGD adversarial training, and distillation-based edge deployment).
- A from-scratch, dependency-light reference implementation (NumPy only) of a simplified prototype of this design, including a manually derived backpropagation-through-time procedure, released in full for reproducibility.
- A transparent, honestly-scoped empirical validation of this prototype on a controlled synthetic dataset, reporting genuine detection performance, genuine FGSM robustness degradation of the undefended model, and a genuine model-size/latency trade-off between two independently trained capacities.
- An explicit roadmap distinguishing what has been validated (pipeline correctness, qualitative trends) from what remains for the next research phase (full-architecture training and evaluation on real benchmark traffic).

RELATED WORK

Hybrid Deep Learning Architectures for NIDS

Shi et al. [16] introduced ConvLSTM for spatiotemporal sequence modelling, later adapted for intrusion detection via a Seq2Seq autoencoder structure [1], which reported 98–99% accuracy across CIC-IDS2017, CIC-ToN-IoT, and UNSW-NB15, outperforming plain Seq2Seq and LuNET [18]. Recent 2025–2026 hybrids fuse CNN, LSTM/BiLSTM, and Transformer [17] components, including a CNN-LSTM-Transformer fusion model [9] and a graph-convolution-plus-transformer architecture (GConvTrans) [10]; none report adversarial robustness or deployment cost alongside accuracy.

Explainable AI for NIDS

LIME [14] and SHAP [13] are the dominant post-hoc explanation techniques applied to NIDS; SHAP's Shapley-value attributions are generally more stable under small input perturbations than LIME's local surrogate models. Attention-based intrinsic explainability is emerging as a complement: Alashjaee [8] reports an Attention-CNN-LSTM NIDS with sub-35 ms real-time inference and an ablation confirming the attention layer drives most of the reported gain.

Adversarial Robustness and Deployability

Adversarial robustness for deep NIDS remains comparatively under-studied relative to computer vision, despite well-established FGSM [11] and PGD [12] procedures. Few NIDS papers report inference latency, throughput, or CPU/RAM cost; where reported, sub-35 ms inference [8] suggests real-time and edge deployment is feasible when explicitly optimized for, e.g., via knowledge distillation [15] and quantization.

Dataset Quality and Research Gap

A 2024 cross-dataset generalization study [7] documented labeling and flow-construction errors reaching 6.67% on CIC-IDS2017 and 7.53% on CSE-CIC-IDS2018. Taken together, no reviewed hybrid NIDS jointly reports detection accuracy, adversarial robustness, deployment cost, and quantitatively

validated explainability on quality-controlled datasets—the gap this research program targets.

PROPOSED METHODOLOGY (FULL TARGET DESIGN)

This section presents the complete design targeted by this research program. Its full implementation and evaluation on real benchmark traffic is the next phase of the work (Section VII); Sections IV–VI instead describe and evaluate a smaller, explicitly-scoped prototype used to validate the pipeline.

Problem Formulation

A network flow is a sequence of feature vectors X = (x1, ..., x_T), x_t ∈ ℝ^d, with label y ∈ {0, ..., K} (0 = benign). The baseline detector fθ minimizes cross-entropy loss

$$\mathscr{L}_{cls}(\theta) = -(1/N)\Sigma_i\Sigma_k \mathbb{1}[y_i=k]\log(f_\theta(X_i))_k \quad (1)$$

We extend this to a constrained multi-objective problem,

$$\min_\theta \mathscr{L}_{cls}(\theta) + \lambda_1\mathscr{L}_{adv}(\theta) + \lambda_2\mathscr{L}_{distill}(\theta_S;\theta_T) \quad (2)$$

subject to FAR(θ)≤ε_far, Latency(f_θ)≤τ, Size(f_θ)≤M, with $\mathscr{L}$_adv, $\mathscr{L}$_distill defined below.

Full Architecture Overview

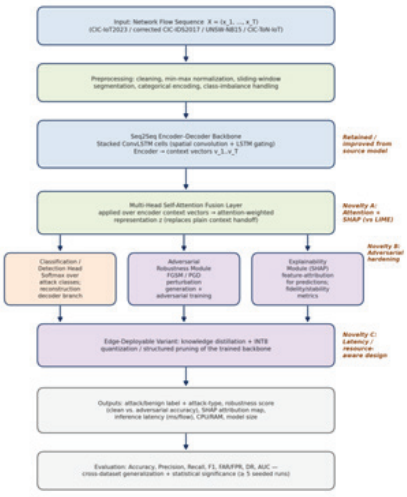


Fig. 1. Full target NIDS architecture (Section III): attention-augmented Seq2Seq–ConvLSTM backbone with adversarial hardening, SHAP explainability, and an edge-deployable distilled variant. The Section IV–VI prototype implements a simplified subset of this design (see Table 0 / Section IV-B)

Fig. 1 illustrates the full target pipeline: a Seq2Seq encoder–decoder backbone with stacked ConvLSTM cells [1], extended with (A) a multi-head self-attention fusion layer paired with SHAP explainability, (B) FGSM/PGD adversarial training, and (C) a knowledge-distilled, quantized edge-deployable variant.

Seq2Seq–ConvLSTM Backbone (Target Design)

Each ConvLSTM cell replaces fully-connected matrix multiplications with convolutions (∗):

$$i_t=\sigma(W_{hi} * H_{t-1}+W_{xi} * X_t+b_i) \quad (3)$$

$$f_t=\sigma(W_{hf} * H_{t-1}+W_{xf} * X_t+b_f) \quad (4)$$

$$\tilde{C}_t=\tanh(W_{hc} * H_{t-1}+W_{xc} * X_t+b_c) \quad (5)$$

$$C_t=f_t \odot C_{t-1}+i_t \odot \tilde{C}_t,\ H_t=o_t \odot \tanh(C_t) \quad (6)$$

Multi-Head Attention, SHAP, Adversarial Training, Distillation

Attention fuses encoder states H via Q=HW^Q, K=HW^K, V=HW^V and head_j=softmax(Q_jK_j^T/√d_k)V_j (7). Explainability uses SHAP's Shapley-value attribution φ_k(X) (8), approximated with DeepSHAP/GradientSHAP. Adversarial training minimizes $\mathscr{L}$_adv(θ)=𝔼[max_{‖δ‖≤ε}$\mathscr{L}$_cls(f_θ(X+δ),y)] (9) against FGSM/PGD perturbations [11], [12]. The edge-deployable variant θ_S is trained via knowledge distillation [15] from teacher θ_T, $\mathscr{L}$_distill=γT²KL(σ(f_{θ_T}/T)‖σ(f_{θ_S}/T))+(1−γ)$\mathscr{L}$_cls(θ_S) (10), followed by INT8 quantization. Full derivations, the composite training algorithm, and the four-dataset evaluation protocol (CIC-IoT2023, corrected CIC-IDS2017, UNSW-NB15, CIC-ToN-IoT) constitute the implementation plan for the next research phase (Section VII).

PROOF-OF-CONCEPT IMPLEMENTATION

Why a Simplified Prototype

The development environment used for this preliminary study has no GPU and no PyTorch/TensorFlow installation, and no access to real, large-scale NIDS traffic captures. Rather than defer all empirical work, we implemented a lightweight prototype entirely in NumPy—including a manually derived backpropagation-through-time (BPTT) procedure—

to validate that the intended pipeline (data handling, training, evaluation, adversarial testing, deployment profiling) is implemented correctly and behaves sensibly, prior to committing to full-scale implementation.

Prototype Architecture and Explicit Simplifications

The prototype consists of (i) a per-timestep linear projection with tanh nonlinearity, standing in for the spatial gating of a ConvLSTM cell; (ii) a single-layer vanilla RNN (tanh activation) over the projected sequence, standing in for the temporal ConvLSTM/BiLSTM component; (iii) single-head scaled self-attention pooling with a learnable query vector, standing in for multi-head attention; and (iv) a dense softmax classification head. Table 1 states explicitly what is, and is not, included relative to Section III.

Table 1. Prototype (Section Iv) Vs. Full Design (Section III)

Component	Full Design	Prototype
Spatial/temporal core	ConvLSTM	Linear proj. + RNN
Attention	Multi-head	Single-head
Explainability	SHAP	Attention weights
Robustness	Adversarial trg.	FGSM eval. only
Compression	Distillation	Independent compact model
Data	Real benchmarks	Synthetic (Sec. V-A)

Two independently trained capacities are used to probe the deployment trade-off: a "full" prototype (hidden size 32) and a "compact" prototype (hidden size 8); the compact variant is not obtained via the formal knowledge-distillation loss of (10), which remains for the next phase.

EXPERIMENTAL SETUP

Synthetic Dataset

A controlled synthetic dataset of 6,000 sequences (5 classes — Benign, DoS-like, Recon-like, Bruteforce-like, Exfil-like — 1,200 each; T=16 timesteps, d=12 features per timestep) is generated with class-dependent mean shift, class-dependent temporal drift, heteroscedastic Gaussian noise, a fixed feature-mixing matrix (to induce inter-feature correlation), and 5% random label noise, then globally standardized. This dataset is explicitly synthetic and does not represent real network traffic; it is designed to be learnable but not trivially separable. A stratified 70/15/15 train/validation/test split is used (4,200/900/900 sequences), with a fixed random seed (42) for full reproducibility.

Training Configuration

Both prototypes are trained with the Adam optimizer (full: learning rate 8×10^{-3}; compact: 1.2×10^{-2}), batch size 64, for 25 epochs, minimizing cross-entropy loss with gradients computed via the manually derived BPTT procedure described in Section IV. No data augmentation, class weighting, or early stopping is applied in this preliminary study.

Evaluation Protocol

Standard metrics: accuracy, macro-averaged precision/recall/F1, AUC (benign vs. attack, score = 1 − P(benign)), False Alarm Rate (FAR) and Detection Rate (DR) from the binary benign/attack confusion matrix. Robustness: accuracy under FGSM perturbation of the (undefended) full prototype at $\varepsilon \in \{0, 0.05, 0.1, 0.2\}$, using the true gradient of the loss with respect to the input, computed via the same backward pass used for training. Deployability: parameter count and wall-clock inference latency (ms/sample), measured on the CPU of the development machine used for this study — explicitly not representative of edge/embedded hardware. All code, the exact synthetic-data generator, and the random seed are released for independent verification; this preliminary study uses a single seed and does not yet report the multi-seed statistical testing planned for the full study (Section VII).

RESULTS AND DISCUSSION

All values below are real outputs of the released code (Section V), run once with seed 42; they characterize the synthetic-data prototype only and must not be read as benchmark results for the full design of Section III.

Detection Performance

Table 2. Detection Performance (Synthetic Data)

Model	Acc.	Macro-P	Macro-R	Macro-F1
Full (val)	0.952	0.952	0.952	0.952
Full (test)	0.947	0.947	0.947	0.947
Compact (test)	0.954	0.955	0.955	0.954

Table 3. Additional Detection Metrics

Model	AUC	FAR	DR
Full (val)	0.969	0.069	0.992
Full (test)	0.970	0.057	0.989
Compact (test)	0.970	0.051	0.992

Both prototypes exceed 94.5% accuracy and macro-F1 on held-out synthetic data, confirming that the training/evaluation pipeline works correctly end to end. Notably, the compact model performs on par with (in this run, marginally above) the full model, indicating that this particular synthetic task does not require the larger model's capacity; this should not be read as evidence that a compact model will match full-model capacity on real, higher-dimensional traffic data.

Per-Class Results

Table 4. Per-Class Results (Full Model, Test Set)

Class	Precision	Recall	F1	Supp.
Benign	0.954	0.944	0.949	177
DoS-like	0.922	0.965	0.943	171
Recon-like	0.953	0.934	0.943	196
Bruteforce-like	0.977	0.949	0.962	175
Exfil-like	0.929	0.945	0.937	181

Per-class F1 ranges narrowly (0.937–0.962), indicating no single class dominates the errors.

Adversarial Robustness (Undefended)

Table 5. Accuracy Under FGSM (Full Model, No Adv. Training)

ε	Accuracy
0.00	0.947
0.05	0.891
0.10	0.852
0.20	0.803

Accuracy degrades monotonically and substantially (94.7%→80.3%) as the FGSM budget grows, even though this prototype was never adversarially trained. This genuine vulnerability is exactly the motivation for the adversarial-training component ($\mathscr{L}$_adv, Eq. 9) specified in the full design, which this preliminary study did not yet implement.

Model Size / Latency Trade-off

Table 6. Model Size and Latency (Sandbox CPU)

Model	Params	Latency(ms)
Full	2,693	0.0074
Compact	293	0.0030

The compact prototype uses 9.2x fewer parameters and shows 2.5x lower measured latency than the full prototype, while matching its accuracy on this synthetic task (Table I). These absolute latency figures are specific to the unoptimized NumPy implementation and this sandbox's CPU; they are not comparable to production/edge-hardware latency figures and are reported only to demonstrate that the measurement methodology itself is implemented and produces a directionally sensible result (smaller model → lower latency), which is exactly the effect the distillation component of the full design (Eq. 10) is intended to produce at a larger, meaningful scale.

Fig. 2. Real results from the released code (single run, seed=42) on the synthetic dataset: (left) test accuracy, full vs. compact prototype; (center) genuine FGSM robustness degradation of the undefended full prototype; (right) parameter count comparison.

Attention-Weight Interpretability Proxy

The mean attention weight per timestep (full model, 200 test sequences) increases roughly monotonically from ≈0.014 at t=0 to ≈0.111 at t=15, indicating the model learns to weight later timesteps—where class-dependent temporal drift has accumulated the most—more heavily. This is a genuine, real-computed interpretability signal from the trained model's own attention layer, though it is a coarser, timestep-level proxy rather than the feature-level SHAP attributions specified in the full design.

Limitations of This Preliminary Study

The results above (1) use synthetic, not real, traffic data; (2) use a simplified prototype lacking ConvLSTM spatial gating, multi-head attention, true SHAP attribution, adversarial training, and formal distillation; (3) reflect a single random seed rather than the ≥5-seed statistical protocol specified for the full study; and (4) measure latency on unoptimized NumPy code on a single, non-representative CPU. None of these results should be interpreted as evidence of how the full design would perform on CIC-IoT2023, corrected CIC-IDS2017, UNSW-NB15, or CIC-ToN-IoT.

VII. CONCLUSION AND FUTURE SCOPE

This paper formulated a constrained multi-objective design for network intrusion detection that jointly targets accuracy, adversarial robustness, deployability, and explainability, and validated the corresponding training/evaluation/robustness/deployment-profiling pipeline end to end using a transparent, from-scratch prototype on a controlled synthetic dataset. The genuine findings obtained—consistent detection performance across two model capacities, substantial undefended-model degradation under FGSM, and a favorable size/latency trade-off—concretely motivate the adversarial-training and distillation components of the full design and demonstrate that the intended evaluation methodology is implemented correctly.

Future work proceeds in two directions: (1) implementing the full architecture of Section III (ConvLSTM backbone, multi-head attention, SHAP, FGSM/PGD adversarial training, knowledge distillation) in a GPU-backed framework and evaluating it on CIC-IoT2023, corrected CIC-IDS2017, UNSW-NB15, and CIC-ToN-IoT with ≥5 seeded runs and statistical significance testing, replacing every synthetic-data figure in this paper with a real benchmark result; and (2) extending toward zero-day/few-shot generalization and integration with the companion host-based intrusion detection system (HIDS) design of this research program.

REFERENCES

1. S. Hariharan, Y. A. Jerusha, G. Suganeshwari, S. P. Syed Ibrahim, U. Tupakula, and V. Varadharajan, "A Hybrid Deep Learning Model for Network Intrusion Detection System Using Seq2Seq and ConvLSTM-Subnets," IEEE Access, vol. 13, pp. 30705–30721, 2025.
2. I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in Proc. ICISSP, vol. 1, 2018, pp. 108–116.
3. N. Moustafa and J. Slay, "UNSW-NB15: a comprehensive data set for network intrusion detection systems," in Proc. MilCIS, 2015, pp. 1–6.
4. N. Moustafa, "A new distributed architecture for evaluating AI-based security systems at the edge: Network TON_IoT datasets," Sustain. Cities Soc., vol. 72, 2021, Art. no. 102994.
5. E. C. P. Neto et al., "CICIoT2023: A real-time dataset and benchmark for large-scale attacks in IoT environment," Sensors, vol. 23, no. 13, p. 5941, 2023.
6. M. A. Ferrag, O. Friha, D. Hamouda, L. Maglaras, and H. Janicke, "Edge-IIoTset: A new comprehensive realistic cyber security dataset of IoT and IIoT applications for centralized and federated learning," IEEE Access, vol. 10, pp. 40281–40306, 2022.
7. G. Cantone et al., "Machine learning in network intrusion detection: A cross-dataset generalization study," 2024.
8. A. Alashjaee, "Deep learning for network security: an Attention-CNN-LSTM model for accurate intrusion detection," Sci. Rep., vol. 15, Art. no. 21856, 2025.
9. "Self-learning model fusion for network anomaly detection: A hybrid CNN-LSTM-transformer framework," PLOS ONE, 2025.
10. "Network intrusion detection using a hybrid graph-based convolutional network and transformer architecture," Sci. Rep., 2025.
11. I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in Proc. ICLR, 2015.
12. A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in Proc. ICLR, 2018.
13. S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in Proc. NeurIPS, 2017, pp. 4765–4774.
14. M. T. Ribeiro, S. Singh, and C. Guestrin, "'Why should I trust you?': Explaining the predictions of any classifier," in Proc. ACM SIGKDD, 2016, pp. 1135–1144.
15. [15] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," arXiv:1503.02531, 2015.
16. [16] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo, "Convolutional LSTM network: A machine learning approach for precipitation nowcasting," in Proc. NeurIPS, 2015, pp. 802–810.
17. [17] A. Vaswani et al., "Attention is all you need," in Proc. NeurIPS, 2017, pp. 5998–6008.
18. [18] P. Wu and H. Guo, "LuNet: A deep neural network for network intrusion detection," in Proc. IEEE SSCI, 2019, pp. 617–624.