Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Dense-Autoencoder Anomaly Detection for 5G Network Traffic: A Reproducible Study of Detection Performance, Seed Sensitivity, and Reconstruction-Error Attribution

Prajwal V.
BMS College of Engineering
R. Ashok Kumar
ashokkumar.ise@bmsce.ac.in

BMS College of Engineering

Research Article

Keywords: 5G security, Dense Autoencoder, anomaly detection, reconstruction-error attribution, KernelSHAP, reproducibility

Posted Date: November 20th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11317159/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Dense-Autoencoder Anomaly Detection for 5G Network Traffic: A Reproducible Study of Detection Performance, Seed Sensitivity, and Reconstruction-Error Attribution

Prajwal V.[1] and R. Ashok Kumar[2*]

[1]Department of Information Science and Engineering (M.Tech CNE), BMS College of Engineering, Bengaluru, 560019, Karnataka, India.
[2*]Department of Computer Science and Business Systems, BMS College of Engineering, Bengaluru, 560019, Karnataka, India.

*Corresponding author(s). E-mail(s): ashokkumar.ise@bmsce.ac.in;
Contributing authors: prajwal.scn24@bmsce.ac.in;

Abstract

The decentralization of 5G network intelligence to Multi-access Edge Computing (MEC) nodes replaces a small number of centrally defended data centres with thousands of distributed, resource-constrained nodes, each requiring its own anomaly-detection capability rather than relying on centralized inspection. This shift demands detection methods that operate within strict per-node latency and compute budgets while remaining interpretable enough to support autonomous or operator-assisted decisions at the edge. We evaluate a Dense Autoencoder, trained exclusively on benign traffic, as a candidate for such distributed deployment, using the 5G-NIDD dataset and a fixed, pre-specified protocol across five independent random seeds. The detector sustains single-flow inference suitable for commodity edge hardware (mean latency 0.095 ms for detection alone and 0.178 ms including attribution on the primary machine, with a second machine showing the same qualitative pattern) and holds benign-flow false-positive rate near its 5% design target (5.05% $\pm$ 0.12%) with precision of 0.953 $\pm$ 0.097 (on a holdout that is 88.5% attack traffic). However, overall detection performance varies substantially across seeds (F1 = 0.561 $\pm$ 0.365), with per-class recall ranging from 0.9998 (ICMP Flood) to 0.220 (UDP Flood). We further compare two interpretability mechanisms suited to distributed, resource-limited nodes: a lightweight reconstruction-error attribution usable in real time, and an offline KernelSHAP explainer; the two agree on the dominant contributing feature in

only 15% of flagged flows, exposing a gap between fast and principled explanation methods for networked edge security. We report seed instability and attribution disagreement as central findings relevant to deploying such detectors across heterogeneous, distributed 5G edge infrastructure, and release the complete pipeline for independent verification.

Keywords: 5G security, Dense Autoencoder, anomaly detection, reconstruction-error attribution, KernelSHAP, reproducibility

1 Introduction

Cellular network security operated for two decades under a simple spatial logic: user traffic from 3G and 4G devices passed through a small number of heavily defended core data centres, and threat inspection could be concentrated at those choke points [3, 4]. This centralized model had scaling weaknesses, but its threat surface was bounded and well characterised.

The 5G transition dismantles this logic at its foundation. Ultra-Reliable Low-Latency Communication (URLLC) mandates end-to-end round-trip times that a centralised data centre cannot satisfy for devices at the network edge [8]. The only engineering resolution is to push computation outward, which motivates Multi-access Edge Computing (MEC) [23, 34], supported by the 5G system architecture introduced in 3GPP Release 15, alongside network slicing as the complementary mechanism for partitioning shared infrastructure into isolated, service-differentiated virtual networks [2, 12]. The consequence for network security is architectural, not merely operational: operators now defend not a handful of hardened facilities but potentially thousands of distributed nodes, each running on commodity hardware with its own compute, memory, and latency budget, and each needing local anomaly-detection capability rather than depending on centralized inspection [31].

This is, fundamentally, a distributed-systems problem before it is a machine-learning problem. A detector deployed to a MEC node must (i) operate within a strict per-node latency budget, since security processing cannot be allowed to violate the URLLC guarantees of the traffic it is inspecting; (ii) run on resource-constrained, heterogeneous commodity hardware without assuming GPU availability; and (iii) produce output that is interpretable enough for an operator – or, eventually, an autonomous orchestration layer – to act on with some causal basis, rather than a bare anomaly score [7]. These three constraints interact: a detector that is accurate but too slow for the per-node budget is not deployable, and a detector that is fast but opaque cannot support operator trust in automated network decisions of the kind that analytics functions such as the 3GPP NWDAF (TS 23.288) are intended to inform [1].

Signature-based detection, the long-standing alternative to learned models, can only flag threats that have already been catalogued; it is structurally blind to zero-day attacks and to fast-moving threats such as Mirai-class IoT botnets, whose variants are generated and distributed faster than signatures can be written [19]. Unsupervised anomaly detection is a natural fit for this setting because it does not require

labelled attack data, which is scarce and quickly stale against novel threats. Autoencoders trained on benign traffic alone, and flagging flows that reconstruct poorly, have been applied to network intrusion detection in prior work [9, 24], including ensemble variants aimed at capturing anomalous structure across network flows. What is less often reported for this setting is (a) how stable such detectors are across independent training runs under an identical protocol, which matters directly for whether a given deployment can be trusted without per-node retraining, and (b) how well fast, deployment-feasible explanation methods agree with more principled but computationally heavier methods such as KernelSHAP [22], since the choice between them is itself a resource/fidelity trade-off that a distributed edge deployment must make explicitly.

This paper reports an empirical study of a Dense Autoencoder for benign-only anomaly detection on the 5G-NIDD dataset [32], designed around the three constraints above. We make four contributions:

1. We report detection performance (F1, AUC-ROC, false-positive rate, recall) across five independent random seeds under a single, fixed protocol, and show that performance varies substantially between seeds despite identical training conditions – a reproducibility finding directly relevant to whether a detector trained once can be trusted for distributed deployment without per-site validation.
2. We measure real single-flow inference latency on commodity CPU hardware, for both detection alone and detection with per-feature attribution, to characterise what is actually deployable within a MEC node's compute budget.
3. We implement and compare two attribution mechanisms: a lightweight per-feature reconstruction-error method with no sampling overhead, and an offline KernelSHAP explainer that produces game-theoretically grounded attributions at materially higher computational cost. We report their quantitative agreement rate rather than assuming the two are interchangeable.
4. We provide a complete, openly available implementation and the full per-seed result set, so that every number reported here can be independently reproduced rather than taken on trust.

We report the seed-to-seed instability and the attribution disagreement as primary findings rather than as limitations to be minimised, because we consider them directly relevant to anyone planning to deploy a similar detector across distributed, heterogeneous 5G edge infrastructure.

2 Related Work

2.1 Classical anomaly-detection baselines

Before learned reconstruction-error methods, anomaly detection relied on classical, non-parametric approaches that remain common baselines today: Isolation Forest, which isolates anomalies via random recursive partitioning on the premise that outliers require fewer splits to isolate than normal points [21], and the Local Outlier Factor (LOF), which scores a point by the local density of its neighbourhood relative to its neighbours' densities [10]. Neither method learns a representation of normal

behaviour in the way an autoencoder does, and neither is evaluated in this study; we flag both explicitly as the comparison baselines our Limitations (Section 5.5) identifies as necessary before any claim of relative superiority for the Dense-AE approach.

2.2 Benchmark datasets for network intrusion detection

5G-NIDD [32] sits within a longer lineage of network intrusion-detection benchmarks. Earlier general-purpose benchmarks include the KDD Cup 99 data set and its corrected successor NSL-KDD [35], CICIDS2017 [30], and UNSW-NB15 [25]; none of these capture 5G-specific protocol fields or a 3GPP-compliant core, which is the gap 5G-NIDD was constructed to close. We do not evaluate on any of these older benchmarks in this study, but note that cross-dataset evaluation is a natural extension given the domain-shift concerns raised by Sommer and Paxson [33] and Apruzzese et al. [6].

2.3 Reconstruction-error anomaly detection for network traffic

Reconstruction-error anomaly detection was applied to system telemetry using dense autoencoders by Borghesi et al. in the context of High-Performance Computing monitoring [9], and extended to real-time network intrusion detection by Mirsky et al. through Kitsune, an ensemble of lightweight autoencoders [24]. We adopt a 95th-percentile reconstruction-error rule for the detection threshold (Section 3); this is a pragmatic, widely used operating point rather than one we validate here against alternative thresholding strategies. To our knowledge, neither of these works reports cross-seed stability of the resulting detector, which is a precondition for deploying a single trained model across many distributed edge nodes without per-site retraining.

2.4 Explaining autoencoder-based anomalies

Lundberg and Lee established SHAP as a theoretically grounded, game-theoretic approach to post-hoc model explanation [22]; LIME, a complementary local surrogate-model approach to explaining individual predictions, was proposed around the same time by Ribeiro et al. [28]. Antwarg et al. specifically applied SHAP attribution to autoencoder reconstruction error on tabular data [5], and Nguyen et al. proposed GEE, a gradient-based explainer purpose-built for variational-autoencoder network-anomaly detectors [26] – the closest prior work, to our knowledge, to the explainable-autoencoder setting evaluated here, though GEE targets a variational autoencoder [18] rather than the deterministic Dense Autoencoder we use. Arrieta et al. provide a broader taxonomy of explainable-AI methods within which KernelSHAP sits [7], and Doshi-Velez and Kim argue more fundamentally that interpretability claims require rigorous, task-grounded evaluation rather than intuitive appeal [11]; Rudin goes further, arguing that high-stakes decisions should favour inherently interpretable models over post-hoc explanations of black boxes [29], a position our own finding of low cross-method attribution agreement (Section 4.6) does not contradict. A recurring limitation noted in this literature is that full Shapley-value computation carries combinatorial cost in the number of input features, which has motivated lightweight, sampling-free alternatives such as raw per-feature reconstruction error for settings with tight latency budgets. What is not well characterised in prior work is *how*

much a lightweight attribution method and a principled KernelSHAP explainer actually agree on the same flagged flow – a question we address directly, since the choice between the two is a real resource/fidelity trade-off for a distributed edge deployment rather than a purely academic one. Recent work by Krishna et al. documents this kind of cross-method disagreement as a general, practitioner-relevant phenomenon across explainable-ML methods broadly [20]; our result is, to our knowledge, among the first to quantify it specifically for network-anomaly attribution.

2.5 Domain shift and reproducibility in network security ML

Sommer and Paxson identified the closed-world assumption – the implicit belief that deployment conditions statistically resemble training conditions – as a structural problem in network-security machine learning [33], and Apruzzese et al. quantified its impact more recently across several ML-based intrusion detection systems [6]. This body of work motivates caution about a single benchmark result generalising to deployment; our contribution is complementary and more basic: we show that even *within* a single, fixed benchmark and a single, fixed training protocol, the detector's own performance is not stable across independent random initialisations, which is a prerequisite concern to domain shift and, to our knowledge, is not separately quantified in prior reconstruction-error anomaly-detection studies on 5G-NIDD.

3 Methods

3.1 System overview

Figure 1 summarises the full pipeline evaluated in this study, in two stages. Stage A (offline, run once per random seed) prepares the data, fits the scaler, trains the Dense Autoencoder on benign traffic only, and calibrates the detection threshold τ on held-out validation-benign data. Stage B (per-flow inference) is what would run on a deployed edge node: an incoming flow is scaled with the already-fitted scaler, passed through the trained autoencoder to obtain a reconstruction error, and compared against τ. Flows below threshold require no further action; flagged flows can be explained by either of the two attribution paths evaluated in this study – the fast, synchronous reconstruction-error method, or the slower, offline KernelSHAP method – which we compare directly in Section 4.6. This two-path structure is intended to make explicit that the detection decision itself never waits on the explanation method chosen, which is the property that makes the slower, more principled KernelSHAP explainer usable at all in a latency-constrained setting.

3.2 Dataset and feature set

We use the 5G-NIDD dataset [32], collected on the 5G Test Network (5GTN) at the University of Oulu, a functional non-standalone (option 3a) 5G network in which attacker nodes and real mobile devices generating benign traffic were attached to two base stations and the captured traffic was converted to flow records. The combined dataset comprises 1,215,890 flow records, of which 477,737 are labelled Benign and

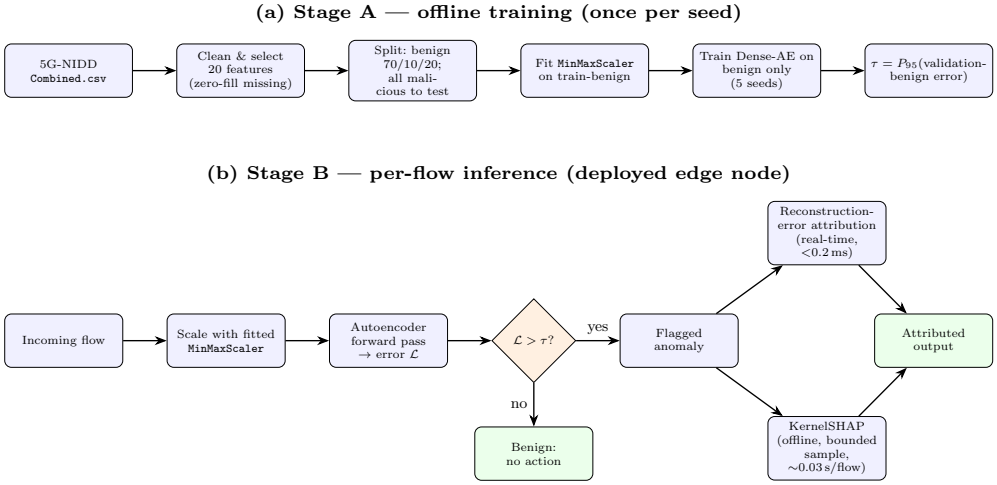

Fig. 1 System pipeline. (a) Stage A is run once per random seed, offline. (b) Stage B is the per-flow path that would execute on a deployed edge node. The threshold τ computed at the end of Stage A is fixed and carried forward unchanged into every Stage B decision.

738,153 are labelled Malicious across eight attack types (UDP Flood, HTTP Flood, Slowrate DoS, TCP Connect Scan, SYN Scan, UDP Scan, SYN Flood, ICMP Flood).

We use a fixed 20-feature subset of the 5G-NIDD flow-record schema: `Dur`, `Proto`, `SrcPkts`, `DstPkts`, `SrcBytes`, `DstBytes`, `SrcLoad`, `DstLoad`, `SrcLoss`, `DstLoss`, `sMeanPktSz`, `dMeanPktSz`, `sTtl`, `dTtl`, `sHops`, `dHops`, `sVid`, `Offset`, `Max`, `Min`. `Proto` is mapped to an integer code using a fixed alphabetical table over the eight protocol values present in the dataset, so the encoding is identical across runs. We note for transparency that `sVid` is blank (missing) in 90.6% of all rows, and `dTtl`/`dHops` are blank in 77.6%; all missing values across the feature set are filled with zero, applied identically to training, validation, and test data. This is a simplifying, documented modelling choice rather than a claim that these fields carry a specific domain meaning when absent.

3.3 Train/validation/test split

The autoencoder is trained exclusively on benign traffic, so no attack flow is ever needed for training or model selection. We therefore split benign flows 70%/10%/20% into training, validation, and test partitions, and assign *all* malicious flows to the test partition, since withholding attack examples from evaluation would serve no purpose. This yields, per random seed, 334,415 benign training flows, 47,774 benign validation flows, and a held-out test set of 833,701 flows (95,548 benign and 738,153 malicious; attack-type composition in Table 1). A `MinMaxScaler` is fit on the training partition only and applied unchanged to validation and test data.

Table 1 Holdout test-set composition (833,701 flows total). Benign-flow count reflects the 20% benign test split and varies by at most one flow across the five seeds reported here.

Traffic class	Flow count	% of holdout
Benign	95,548	11.46
UDP Flood	457,340	54.86
HTTP Flood	140,812	16.89
Slowrate DoS	73,124	8.77
TCP Connect Scan	20,052	2.41
SYN Scan	20,043	2.40
UDP Scan	15,906	1.91
SYN Flood	9,721	1.17
ICMP Flood	1,155	0.14
Total	**833,701**	**100.00**

3.4 Model architecture

The detector is a Dense Autoencoder with the topology in Table 2: a 20-dimensional input is compressed through two ReLU-activated hidden layers to an 8-dimensional latent representation, then reconstructed through a mirrored decoder with a sigmoid output layer. The 8-dimensional bottleneck (a 2.5:1 compression ratio against the 20-dimensional input) was chosen as a conventional, moderate compression factor for a feature set of this size; we did not sweep this hyperparameter, and a narrower or wider bottleneck is a reasonable direction for follow-up tuning rather than a setting we claim to be optimal.

Table 2 Dense-Autoencoder topology.

Layer	Activation	Output dim.	Role
Input vector	–	20	Raw features
Dense encoder 1	ReLU	32	Feature extraction
Dense encoder 2	ReLU	16	Compression
Dense bottleneck	ReLU	8	Latent representation
Dense decoder 1	ReLU	16	Expansion
Dense decoder 2	ReLU	32	Recovery
Output layer	Sigmoid	20	Reconstruction

3.5 Training protocol

Training minimises mean squared reconstruction error on benign traffic only (Eq. 1), the standard autoencoder training objective [14]; no attack label is used at training time.

$$\mathcal{L}_{\text{MSE}} = \frac{1}{T}\sum_{t=1}^{T}\|x_t - \hat{x}_t\|^2 \tag{1}$$

We use Adam [17] ($\alpha = 10^{-3}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$), mini-batch size 256, for up to 50 epochs with early stopping (patience 5 epochs on validation loss). The model weights corresponding to the lowest validation loss observed during training are restored before evaluation, as summarised in Algorithm 1. The anomaly threshold τ is fixed as the 95th percentile of *validation*-set (benign-only) reconstruction error, computed once after training and never adjusted on test data (Eq. 2).

$$\tau = \text{Percentile}_{95}\big(\mathcal{L}_{\text{MSE}}(x_i) : x_i \in X_{\text{val-benign}}\big) \tag{2}$$

Algorithm 1 Offline training and threshold calibration

Require: Benign training set X_{train}, benign validation set X_{val}, max epochs $E = 50$, patience $P = 5$
Ensure: Weights Θ, scaler S, threshold τ

$S \leftarrow \text{MinMaxScaler.fit}(X_{\text{train}})$
Initialise Θ randomly; best_loss $\leftarrow \infty$; bad $\leftarrow 0$
for $e = 1$ to E **do**
  for each mini-batch $B \subset S(X_{\text{train}})$ **do**
    $\hat{B} \leftarrow \text{Autoencoder}(B; \Theta)$; $\mathcal{L} \leftarrow \text{MSE}(\hat{B}, B)$
    $\Theta \leftarrow \Theta - \alpha\nabla_\Theta\mathcal{L}$ ▷ Adam, $\alpha = 10^{-3}$
  end for
  $v \leftarrow \text{MSE}(\text{Autoencoder}(S(X_{\text{val}}); \Theta), S(X_{\text{val}}))$
  if $v <$ best_loss **then**
    best_loss $\leftarrow v$; store a copy of Θ as Θ^*; bad $\leftarrow 0$
  else
    bad $\leftarrow$ bad $+ 1$; **if** bad $\geq P$ **then break**
  end if
end for
$\Theta \leftarrow \Theta^*$ ▷ restore best-validation weights
$\tau \leftarrow \text{Percentile}_{95}(\mathcal{L}_{\text{MSE}}(S(X_{\text{val}})))$
return Θ, S, τ

Five independent trials are run with fixed random seeds $\{42, 123, 256, 789, 1024\}$, each repeating the full protocol above (independent weight initialisation, independent mini-batch ordering) on an identical data split and identical hyperparameters. Five seeds is a practical choice given the per-seed training cost on commodity CPU hardware rather than a statistically motivated minimum; we report the 95% confidence intervals that follow from $n = 5$ explicitly throughout, and note in Section 5 that a larger seed count would tighten these intervals and is a direct, low-cost extension of this work.

3.6 Explanation methods

3.6.1 Reconstruction-error attribution

For a flagged flow, the root-cause feature is identified directly from the per-feature contribution to reconstruction error (Eq. 3), computed synchronously in the same forward pass used for detection, with no sampling, no background dataset, and no coalition enumeration.

$$\text{RootCause} = \arg\max_i (x_i - \hat{x}_i)^2 \tag{3}$$

3.6.2 KernelSHAP

For comparison, we compute KernelSHAP attributions [22] on the scalar reconstruction-error function $f(x) = \|x - \hat{x}\|^2$, using a background set of 100 randomly sampled benign training flows and 500 coalition samples per explained flow. Because this computation is materially more expensive than the reconstruction-error method, it is run offline, on a bounded sample of flagged flows (up to 10 per attack class, fixed random selection), rather than on every flow – consistent with our framing of this as an asynchronous, non-real-time explanation path rather than a replacement for the fast attribution method on the detection critical path.

3.7 Latency measurement

Single-flow inference latency is measured on a single CPU thread, over 5,000 randomly sampled holdout flows (200 warm-up calls discarded), for two configurations: detection alone (reconstruction-error forward pass only), and detection with attribution (forward pass plus the per-feature attribution in Eq. 3). We report this separately from KernelSHAP latency, which is not intended for the per-flow critical path.

3.8 Statistical analysis

Precision, recall, F1-score (binary, with attack as the positive class and benign as the negative class), AUC-ROC, and false-positive rate (FPR $= FP/(FP+TN)$, computed over benign holdout flows only) are computed per seed. We report the mean and 95% confidence interval across the five seeds using the t-distribution ($n = 5$). Per-attack-class recall is computed and aggregated the same way. The threshold τ is fixed prior to test-set evaluation for every seed and is never tuned on test data.

3.9 Reproducibility

The complete implementation, including the data-preparation, training, evaluation, attribution, and latency-measurement scripts, and the full per-seed reference results, is openly available at https://github.com/PrajwalPrasana/IEI-5G-DenseAE/releases/tag/v1.0-ijndc-submission. The experiments reported here were independently executed on two different machines (a single-core Linux environment and a 14-core Windows laptop), with the same qualitative pattern reproduced on both; the Windows

run is reported as the primary result set, as it reflects more representative commodity hardware.

4 Results

4.1 Overall detection performance

Table 3 reports per-seed results and Table 4 the aggregate across all five seeds. The false-positive rate is tightly controlled around its 5% design target in every seed, as expected given the threshold definition (Eq. 2). Precision on flagged flows is consistently high (0.81–0.99). However, F1 and recall vary substantially across seeds: seed 256 detects only 2.8% of attacks (F1 = 0.054), while seed 789 detects 67.6% (F1 = 0.804), despite both seeds using an identical architecture, hyperparameters, and data split. Figure 2 and Figure 3 show this spread directly.

Table 3 Per-seed detection performance on the holdout test set.

Seed	F1	AUC-ROC	FPR	Recall	Precision
42	0.621	0.796	0.0503	0.453	0.986
123	0.713	0.869	0.0521	0.558	0.988
256	0.054	0.776	0.0497	0.028	0.812
789	0.804	0.917	0.0500	0.676	0.990
1024	0.614	0.867	0.0504	0.446	0.986

Table 4 Overall detection performance, mean ± 95% CI (t-distribution, $n = 5$ seeds).

Metric	Mean ± 95% CI	Min	Max
F1	0.561 ± 0.365	0.054	0.804
AUC-ROC	0.845 ± 0.072	0.776	0.917
FPR	0.0505 ± 0.0012	0.0497	0.0521
Recall	0.432 ± 0.304	0.028	0.676
Precision	0.953 ± 0.097	0.812	0.990

4.2 Score distributions and per-attack-class performance

Figure 5 shows that the benign and attack reconstruction-error distributions overlap substantially, with a non-trivial fraction of attack flows scoring below the threshold τ. Figure 6 shows the same scores relative to the threshold (error/τ) for each traffic class (seed 42): the benign median lies well below 1, the medians of most attack classes sit several-fold above it, and the UDP Flood median falls below the threshold, consistent

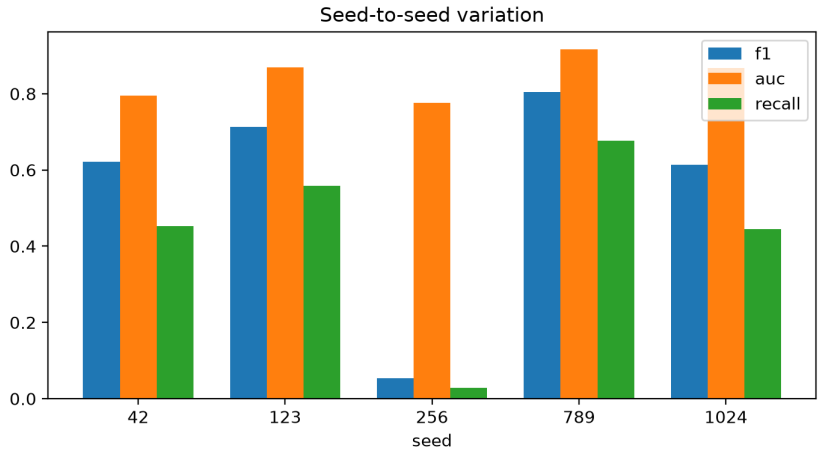

Fig. 2 F1, AUC-ROC, and recall for each of the five training seeds, showing seed 256 as a clear outlier despite an identical protocol.

with its low recall in Table 5. Table 5 and Figure 7 report recall by attack type: ICMP Flood is detected almost perfectly (0.9998), while UDP Flood – the largest attack class by flow count – is the weakest at 0.220, with wide confidence intervals for several other classes reflecting the same seed instability seen in the aggregate results.

Table 5 Per-attack-class recall, mean ± 95% CI across five seeds.

Attack class	n (holdout)	Recall (mean ± 95% CI)	Min / Max
ICMP Flood	1,155	0.9998 ± 0.0005	0.999 / 1.000
UDP Scan	15,906	0.906 ± 0.262	0.529 / 1.000
SYN Scan	20,043	0.887 ± 0.282	0.482 / 1.000
HTTP Flood	140,812	0.800 ± 0.549	0.009 / 1.000
Slowrate DoS	73,124	0.758 ± 0.532	0.000 / 1.000
SYN Flood	9,721	0.732 ± 0.538	0.000 / 1.000
TCP Connect Scan	20,052	0.492 ± 0.616	0.001 / 0.998
UDP Flood	457,340	0.220 ± 0.240	0.000 / 0.502

4.3 Why UDP Flood is poorly detected

Table 6 compares median feature values between benign UDP flows and UDP Flood flows sharing the same protocol. Most features – including `sMeanPktSz`, `sTtl`, and `sHops` – are identical or near-identical between the two classes; the only features showing a substantial difference are `SrcPkts`, `SrcBytes`, `SrcLoad`, `Dur`, and `Offset` (see Section 5.4). This suggests that, on the declared feature set, a UDP Flood flow and a benign UDP flow are only weakly separated; we interpret this as a likely feature-representation limitation, although we did not test alternative training or thresholding strategies.

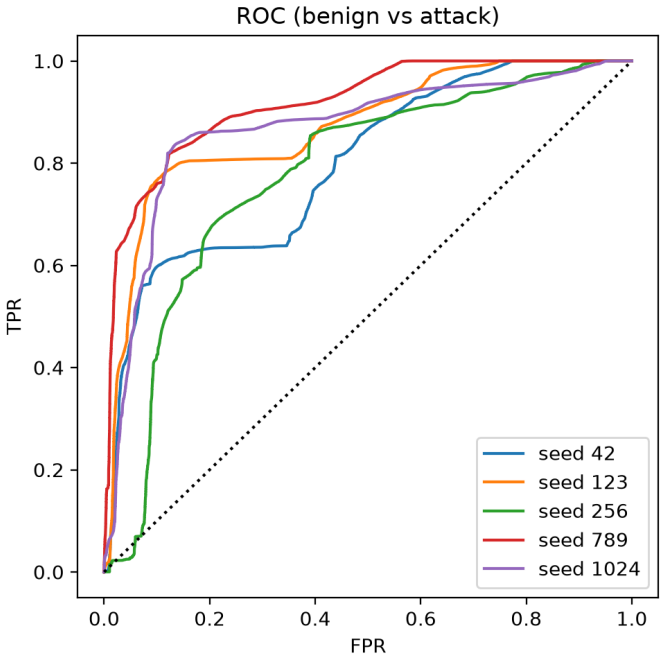

Fig. 3 ROC curves (benign vs. attack) for each of the five seeds.

Table 6 Median feature values: benign UDP flows vs. UDP Flood flows (same protocol).

Feature	Benign UDP (median)	UDP Flood (median)
Dur	0.00	2.28
SrcPkts	1.00	2.00
SrcBytes	74.00	84.00
SrcLoad	0.00	73.75
sMeanPktSz	42.00	42.00
sTtl	63.00	63.00
sHops	1.00	1.00
Offset	21,465,804	10,047,710

4.4 Latency

Table 7 reports single-flow inference latency. Detection alone completes in 0.095 ms on average (p99 0.312 ms); adding per-feature attribution roughly doubles this to 0.178 ms (p99 0.757 ms). Both configurations are consistent with sub-millisecond operation on commodity CPU hardware with no GPU.

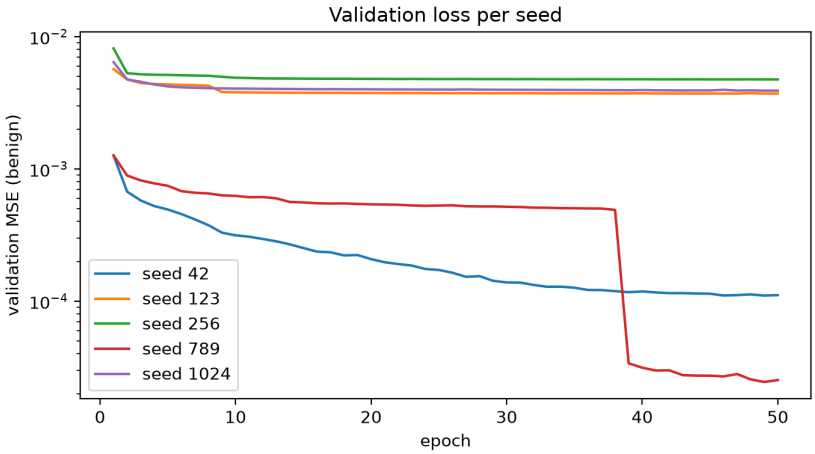

Fig. 4 Validation reconstruction-error loss per epoch, by seed (log scale). Seed 789 shows a sharp, late (epoch ≈39) drop in validation loss, coinciding with its eventual best-overall detection performance; this non-monotonic trajectory is discussed in Section 5.

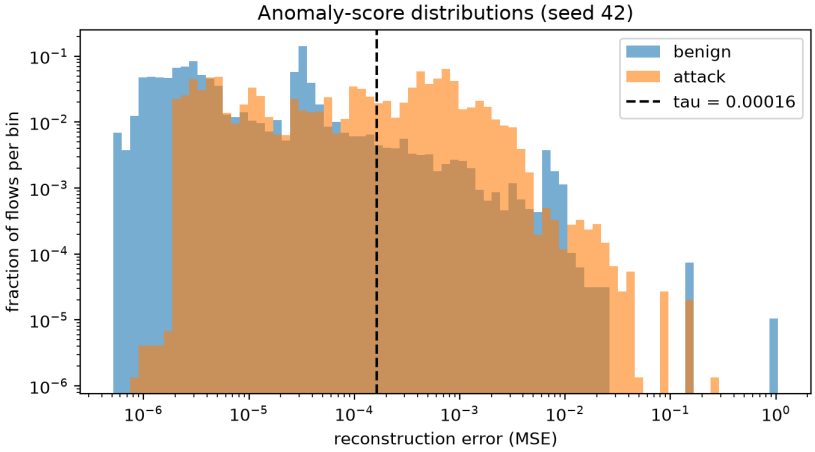

Fig. 5 Reconstruction-error distributions for benign and attack holdout flows (seed 42), with the detection threshold τ marked.

4.5 Reconstruction-error attribution

Figure 8 shows which features carry the largest share of reconstruction error for flagged flows of each class, and Figure 9 shows the top-5 attributed features for one representative (median-score) flagged flow per attack class (Table 8 lists the specific flow examined for each class). Attribution is concentrated differently across classes: `Proto` dominates for TCP Connect Scan (94.8% mean share), `sTtl` and `sHops` dominate for SYN Scan and UDP Scan respectively, `dTtl`/`dHops` dominate for HTTP Flood, SYN Flood, ICMP Flood and Slowrate DoS, and `sMeanPktSz` dominates for UDP Flood. No

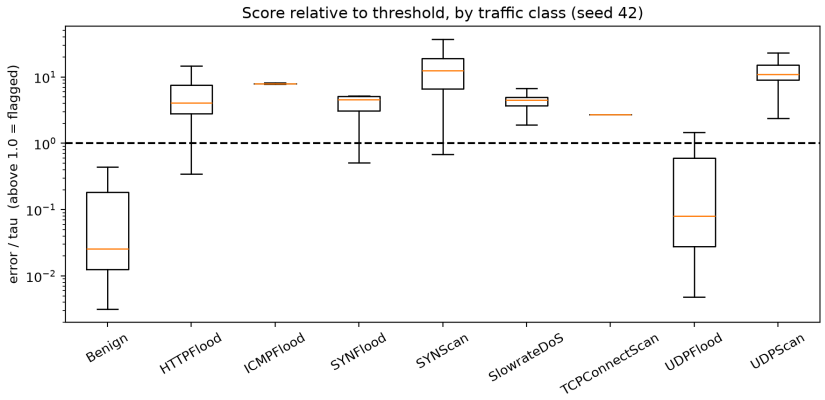

Fig. 6 Reconstruction error relative to threshold (error/τ), by traffic class (seed 42). Values above the dashed line at 1.0 are flagged.

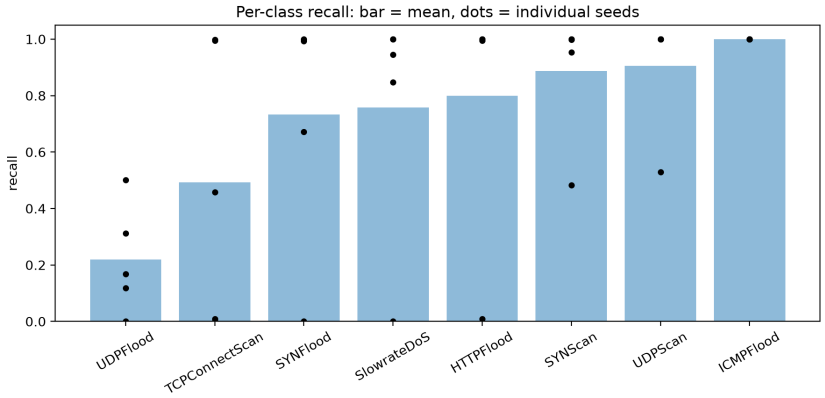

Fig. 7 Per-class recall: bars show the mean across seeds, dots show each individual seed.

single feature dominates across all classes, and `sVid` – despite being the single most-discussed feature in earlier iterations of this work – carries a negligible attribution share ($<0.1\%$) for every class on this feature set, consistent with it being zero-filled for the large majority of flows regardless of traffic type (Section 3).

4.6 Agreement between reconstruction-error attribution and KernelSHAP

Across 80 flagged flows (10 per attack class) explained by both methods, KernelSHAP agreed with the reconstruction-error method's top-1 feature in only 15.0% of cases (mean top-3 overlap 1.25 of 3 features), at a mean cost of 0.026 s per flow for KernelSHAP versus the sub-millisecond cost of the reconstruction-error method. Figure 10 shows this agreement broken down by class: only SYN Scan and UDP Scan show any meaningful top-1 agreement, with every other class at or near 0%.

Table 7 Single-flow CPU inference latency (1 thread, 5,000 sampled holdout flows, 200 warm-up calls discarded). Hardware: 14-core Intel x86-64 laptop CPU, PyTorch 2.14 (CPU build).

Configuration	Mean (ms)	p50 (ms)	p99 (ms)
Detection only	0.095	0.081	0.312
Detection + attribution	0.178	0.151	0.757

Table 8 Representative (median-score) flagged flow per attack class, seed 42, and its top attributed feature.

Attack class	Holdout row	Score ($\mathcal{L}_{\text{MSE}}$)	Top feature
HTTP Flood	336,374	0.000663	`dTtl`
ICMP Flood	96	0.001290	`dHops`
SYN Flood	400,323	0.000746	`dTtl`
SYN Scan	639,197	0.002023	`sTtl`
Slowrate DoS	458,368	0.000733	`dHops`
TCP Connect Scan	387,883	0.000441	`Proto`
UDP Flood	805,476	0.000233	`sMeanPktSz`
UDP Scan	139,198	0.001790	`sHops`

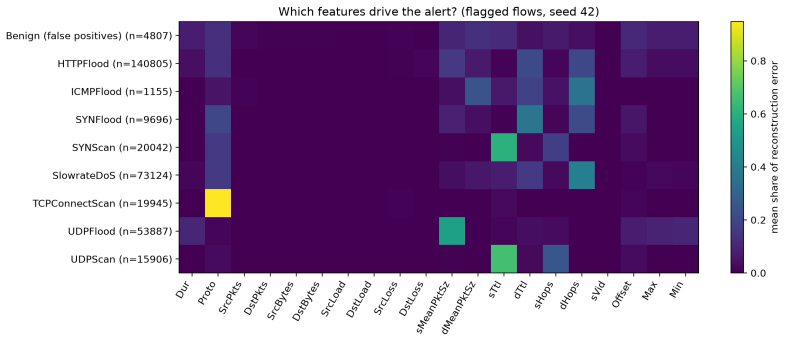

Fig. 8 Mean share of reconstruction error carried by each feature, for flagged flows of each class (including benign false positives), seed 42.

5 Discussion

5.1 Seed instability is a deployment-relevant finding, not noise

The more than ten-fold range in F1 (0.054 to 0.804) across otherwise identical training runs (Table 3) means that a single trained model, evaluated once, is not a reliable description of what this architecture achieves on this task. For a centralized benchmarking study this is a reproducibility caveat; for a distributed MEC deployment, where the practical proposal is to train once and deploy the same weights to many

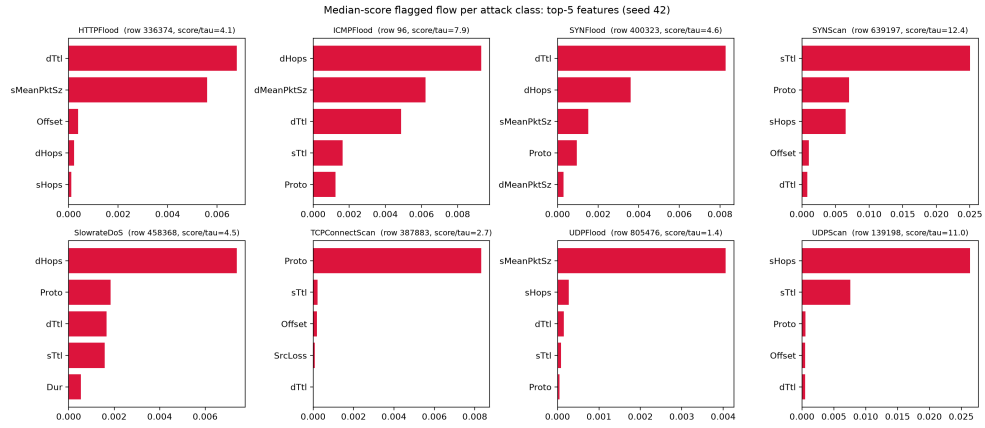

Fig. 9 Top-5 attributed features for the representative flagged flow of each attack class (Table 8).

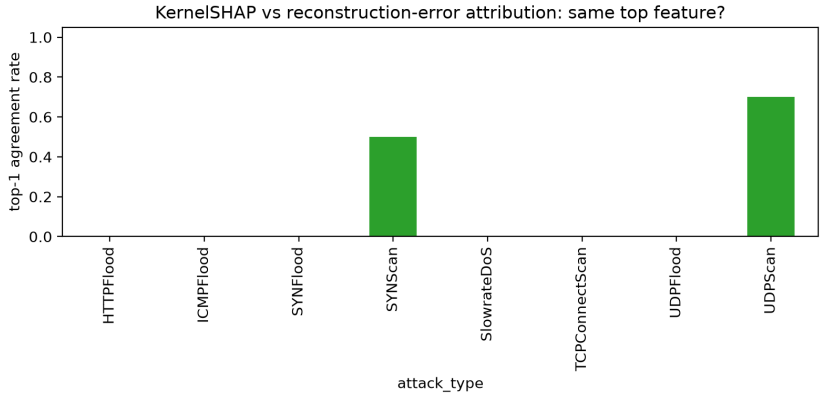

Fig. 10 Top-1 feature agreement rate between KernelSHAP and reconstruction-error attribution, by attack class.

nodes, it is a direct operational risk: a node could be running the equivalent of seed 256 (F1 = 0.054) without any local signal that it is doing so, since its false-positive rate looks identical to a well-performing node (Table 3). This is not unique to this architecture or task: Henderson et al. document closely analogous seed-driven variance across otherwise-identical deep reinforcement learning runs, and argue that reporting a single trained model's result without characterising this variance is a broader reproducibility failure mode in applied deep learning [15]. Our finding is, in that sense, a domain-specific instance of a more general phenomenon rather than an anomaly specific to this detector. We do not have a full mechanistic explanation for the instability observed here. We note descriptively that the best-performing seed (789) showed a sharp, late drop in validation loss partway through training (Figure 4, around epoch 39) that was not observed in the other four seeds, suggesting the optimisation landscape for

this architecture admits qualitatively different solutions depending on initialisation; we flag this as an open question rather than a resolved one. We also note that seeds 42 and 789 were still improving when the 50-epoch cap was reached (Figure 4), so part of the observed variance may reflect incomplete convergence rather than initialisation alone; a longer training budget was not tested. A practical implication is that any deployment of this approach should train multiple independent candidates rather than assume a single run is representative. Because our validation set is benign-only, validation loss cannot serve as the selection criterion: seed 42 reaches a final validation loss more than an order of magnitude lower than seed 123 (Figure 4) yet a lower F1 (Table 3). Selecting among candidates therefore requires a labelled or synthetically generated attack validation set, which we did not evaluate. More broadly, this instability sharpens a choice that distributed intrusion-detection deployments already face between centralized, on-device, and federated training regimes [27]: a single centrally trained model pushed unchanged to every edge node inherits whichever seed's instability it happened to train with, whereas on-device or federated retraining at each node offers a path to detect and correct a poorly performing instance locally, at the cost of the coordination and communication overhead such schemes introduce.

5.2 UDP Flood is a feature-representation limit, not a tuning failure

Section 4.3 shows that benign UDP flows and UDP Flood flows are nearly indistinguishable on most of the declared 20 features; the attack is a request-rate phenomenon that this per-flow, non-temporal feature set does not capture well. This is a plausible consequence of discarding temporal structure between consecutive flows, information a feedforward architecture cannot access by design; we did not evaluate a temporal (e.g. LSTM-based [16]) variant in this study, and flag that as the most direct next step for improving UDP Flood recall specifically, rather than as a general claim that this architecture is unsuitable.

5.3 Fast and principled attribution largely disagree

The 15% top-1 agreement rate between reconstruction-error attribution and KernelSHAP (Section 4.6) is low enough that the two cannot be treated as interchangeable explanations of the same flagged flow. This matters directly for the distributed-deployment framing of this paper: if an edge node reports a fast, real-time attribution to an operator, and a slower, offline KernelSHAP pass over the same flow would frequently have named a different feature, then the fast explanation should be presented with that caveat rather than as if it were a lightweight approximation of the principled one. We do not have grounds, from this study, to say which method is "more correct" in a causal sense; we report the disagreement itself as the finding. The two methods also answer different questions: reconstruction-error attribution decomposes the observed error across output features, whereas KernelSHAP measures each input feature's marginal effect on the error relative to a background distribution, so some disagreement is expected. For reference, random choice among 20 features would give

5% top-1 agreement and 0.45 top-3 overlap; the observed 15% (12/80; 95% Wilson interval approximately 9–24%) is above chance but low.

5.4 The `Offset` feature is a caveat, not a discovery

`Offset` carries a non-trivial share of attribution for several classes (Figure 8; e.g. 10.5% for benign false positives, 7.4% for UDP Flood). In the 5G-NIDD flow-record schema, `Offset` reflects a flow's position within the capture file rather than an intrinsic traffic-behaviour property. Its presence among the attributed features for some classes may therefore partly reflect capture-time ordering rather than genuine behavioural signal, and we flag this explicitly as a limitation of the feature set rather than presenting it as a discovered causal mechanism. A replication excluding `Offset` would be a useful, low-cost follow-up check, which we did not run as part of the frozen protocol reported here. Because each attack type was captured in a separate session [32], attack classes are likely to occupy different regions of the capture file, so `Offset` is likely to correlate with class label (Table 6), which could affect the detection results themselves and not only the attribution.

5.5 Limitations and threats to validity

We organise this section around construct, internal, and external validity, since each points to a different kind of follow-up work.

Construct validity concerns whether our measurements capture what we intend them to. The `Offset` caveat (Section 5.4) is the clearest threat here: part of the measured attribution for some classes may reflect flow position within the capture file rather than traffic behaviour, which we flag rather than resolve. Our zero-fill treatment of missing `sVid`, `dTtl`, and `dHops` values is a second: zero is a defensible default but is not validated against alternative imputation strategies, and a model trained under a different missing-value convention could behave differently.

Internal validity concerns whether our experimental design supports the causal claims we draw from it. Five seeds is enough to reveal that instability exists (Section 4.1) but is a small sample for precisely characterising its distribution; the confidence intervals we report are correspondingly wide, and a larger seed count is the most direct way to tighten them. We also did not run baseline comparisons against established methods such as Isolation Forest or Kitsune [24]; our claims are about this detector's own behaviour (stability, latency, attribution agreement) rather than its relative standing against alternatives, and we consider a baseline comparison necessary before any claim of relative superiority.

External validity concerns how far these findings generalise beyond this study. All results are offline, benchmark evaluations on a single dataset; no live network deployment was performed. Latency was measured on two specific machines (Section 3.9) rather than a representative sample of MEC-class hardware; the two machines agreed closely, which is encouraging but not a claim of generality across all commodity edge hardware, virtualisation layers, or CPU architectures that a real distributed deployment would encounter. We also note that our evaluation is a single snapshot in time: network traffic composition drifts as protocols, devices, and usage

patterns change, and concept drift of this kind is a well-documented long-run threat to any static ML-based detector [13]; nothing in this study measures how quickly the reported performance would degrade under such drift, since 5G-NIDD, like the benchmark datasets in Section 2.2, is itself a fixed snapshot. The holdout set is 88.5% attack traffic by construction, which inflates precision relative to a deployment in which benign traffic dominates; at the observed recall and false-positive rate, precision would fall to a few per cent at 0.5% attack prevalence, so the precision values reported here should not be read as operational estimates.

6 Conclusion

We evaluated a Dense Autoencoder for benign-only anomaly detection on 5G-NIDD under a fixed, five-seed protocol, with an explicit framing around the constraints of distributed 5G edge deployment: per-node latency budget, commodity hardware, and interpretable output. The detector holds its false-positive rate near its design target and achieves high precision on flagged flows, but its recall and F1 vary substantially across otherwise identical training runs, and its weakest attack class (UDP Flood) is weak, plausibly because the per-flow, non-temporal features do not capture request-rate behaviour. A fast, deployment-feasible attribution method and a slower, principled KernelSHAP explainer agree on the dominant contributing feature in only 15% of examined cases. We report these as the central findings of this study: a detector intended for distributed deployment across many independent edge nodes should be evaluated, and re-evaluated, for this kind of seed-level and explanation-level variability, not only for its headline accuracy on a single trained instance. The complete implementation and full per-seed results are released for independent verification.

Statements and Declarations

Competing Interests

The authors declare no competing interests.

Funding

Open access / publication funding for this work is provided by BMS College of Engineering (publication funding only; the underlying research received no dedicated project funding).

Ethics Declaration

Ethics approval and consent to participate: Not applicable. This study did not involve human participants, human data or tissue, or animals. It is a computational secondary analysis of the publicly available 5G-NIDD dataset [32], which was generated by its creators on a 5G test network using their own attacker nodes and test devices; the authors of this study did not collect any data, and did not recruit or interact with any participants. Because no human or animal subjects were involved, no ethics committee approval was sought or required, and no ethical guidelines for

research involving human or animal subjects applied. The authors declare that this research was conducted in accordance with the principles of research integrity, including honest reporting of results and full disclosure of AI tool use (see "Use of Artificial Intelligence Tools").

Consent for publication: Not applicable.

Author Contributions

R.A.K. conceived the research direction and supervised the project. P.V. designed and implemented the experimental pipeline, executed all experiments, performed the analysis, and wrote the original draft. Both authors reviewed and approved the final submitted version.

Data Availability

The 5G-NIDD dataset used in this study is publicly available from its creators via IEEE DataPort (https://ieee-dataport.org/node/10203) and the Fairdata service (https://etsin.fairdata.fi/dataset/9d13ef28-2ca7-44b0-9950-225359afac65); the `Combined.csv` file was used here, obtained through a mirror at https://www.kaggle.com/datasets/humera11/5g-nidd-dataset. No new datasets were generated.

Code Availability

The complete implementation and reference results are available at https://github.com/PrajwalPrasana/IEI-5G-DenseAE/releases/tag/v1.0-ijndc-submission.

Use of Artificial Intelligence Tools

AI-based tools were used solely to assist with language editing, including grammar correction, proofreading, and improving the clarity and readability of the manuscript. AI-based tools were also used to check the manuscript for potential plagiarism or textual similarity. No AI tools were used to conduct the research, develop the methodology, perform experiments, analyze data, generate results, or draw scientific conclusions. The authors remain fully responsible for the content, accuracy, originality, and integrity of the manuscript.

References

[1] 3GPP (2024) Architecture enhancements for 5g system (5gs) to support network data analytics services. Tech. Rep. TS 23.288, 3rd Generation Partnership Project (3GPP)

[2] Afolabi I, Taleb T, Samdanis K, et al (2018) Network slicing and softwarization: A survey on principles, enabling technologies, and solutions. IEEE Communications Surveys & Tutorials 20(3):2429–2453. https://doi.org/10.1109/COMST.2018.2815638

[3] Ahmad I, Kumar T, Liyanage M, et al (2018) Overview of 5g security challenges and solutions. IEEE Communications Standards Magazine 2(1):36–43. https://doi.org/10.1109/MCOMSTD.2018.1700063

[4] Ahmad I, Shahabuddin S, Kumar T, et al (2019) Security for 5g and beyond. IEEE Communications Surveys & Tutorials 21(4):3682–3722

[5] Antwarg L, Miller RM, Shapira B, et al (2021) Explaining anomalies detected by autoencoders using shapley additive explanations. Expert Systems with Applications 186:115736. https://doi.org/10.1016/j.eswa.2021.115736

[6] Apruzzese G, Pajola L, Conti M (2022) The cross-evaluation of machine learning-based network intrusion detection systems. IEEE Transactions on Network and Service Management https://doi.org/10.1109/TNSM.2022.3157344

[7] Barredo Arrieta A, Díaz-Rodríguez N, Del Ser J, et al (2020) Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. Information Fusion 58:82–115. https://doi.org/10.1016/j.inffus.2019.12.012

[8] Bennis M, Debbah M, Poor HV (2018) Ultra-reliable and low-latency wireless communication: Tail, risk, and scale. Proceedings of the IEEE 106(10):1834–1853

[9] Borghesi A, Bartolini A, Lombardi M, et al (2019) Anomaly detection using autoencoders in high performance computing systems. In: Proceedings of the AAAI Conference on Artificial Intelligence, pp 9428–9433, https://doi.org/10.1609/aaai.v33i01.33019428

[10] Breunig MM, Kriegel HP, Ng RT, et al (2000) Lof: Identifying density-based local outliers. In: Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, pp 93–104, https://doi.org/10.1145/335191.335388

[11] Doshi-Velez F, Kim B (2017) Towards a rigorous science of interpretable machine learning. ArXiv:1702.08608

[12] Foukas X, Patounas G, Elmokashfi A, et al (2017) Network slicing in 5g: Survey and challenges. IEEE Communications Magazine 55(5):94–100. https://doi.org/10.1109/MCOM.2017.1600951

[13] Gama J, Žliobaitė I, Bifet A, et al (2014) A survey on concept drift adaptation. ACM Computing Surveys 46(4):44. https://doi.org/10.1145/2523813

[14] Goodfellow I, Bengio Y, Courville A (2016) Deep Learning. MIT Press

[15] Henderson P, Islam R, Bachman P, et al (2018) Deep reinforcement learning that matters. In: Proceedings of the AAAI Conference on Artificial Intelligence, pp 3207–3214

[16] Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Computation 9(8):1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

[17] Kingma DP, Ba J (2015) Adam: A method for stochastic optimization. In: 3rd International Conference on Learning Representations (ICLR)

[18] Kingma DP, Welling M (2014) Auto-encoding variational bayes. In: 2nd International Conference on Learning Representations (ICLR)

[19] Kolias C, Kambourakis G, Stavrou A, et al (2017) Ddos in the iot: Mirai and other botnets. Computer 50(7):80–84. https://doi.org/10.1109/MC.2017.201

[20] Krishna S, Han T, Gu A, et al (2024) The disagreement problem in explainable machine learning: A practitioner's perspective. Transactions on Machine Learning Research ArXiv:2202.01602

[21] Liu FT, Ting KM, Zhou ZH (2008) Isolation forest. In: 2008 Eighth IEEE International Conference on Data Mining (ICDM), pp 413–422, https://doi.org/10.1109/ICDM.2008.17

[22] Lundberg SM, Lee SI (2017) A unified approach to interpreting model predictions. In: Advances in Neural Information Processing Systems, pp 4765–4774

[23] Mach P, Becvar Z (2017) Mobile edge computing: A survey on architecture and computation offloading. IEEE Communications Surveys & Tutorials 19(3):1628–1656. https://doi.org/10.1109/COMST.2017.2682318

[24] Mirsky Y, Doitshman T, Elovici Y, et al (2018) Kitsune: An ensemble of autoencoders for online network intrusion detection. In: 25th Annual Network and Distributed System Security Symposium (NDSS), https://doi.org/10.14722/ndss.2018.23204

[25] 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), pp 1–6, https://doi.org/10.1109/MilCIS.2015.7348942

[26] Nguyen QP, Lim KW, Divakaran DM, et al (2019) Gee: A gradient-based explainable variational autoencoder for network anomaly detection. In: 2019 IEEE Conference on Communications and Network Security (CNS), pp 91–99, https://doi.org/10.1109/CNS.2019.8802833

[27] Rahman SA, Tout H, Talhi C, et al (2020) Internet of things intrusion detection: Centralized, on-device, or federated learning? IEEE Network 34(6):310–317. https://doi.org/10.1109/MNET.011.2000286

[28] Ribeiro MT, Singh S, Guestrin C (2016) "why should i trust you?": Explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp 1135–1144, https://doi.org/10.1145/2939672.2939778

[29] Rudin C (2019) Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. Nature Machine Intelligence 1(5):206–215. https://doi.org/10.1038/s42256-019-0048-x

[30] Sharafaldin I, Lashkari AH, Ghorbani AA (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 (ICISSP), pp 108–116, https://doi.org/10.5220/0006639801080116

[31] Shi W, Cao J, Zhang Q, et al (2016) Edge computing: Vision and challenges. IEEE Internet of Things Journal 3(5):637–646. https://doi.org/10.1109/JIOT.2016.2579198

[32] Siriwardhana Y, Samarakoon S, Porambage P, et al (2025) Descriptor: 5g wireless network intrusion detection dataset (5g-nidd). IEEE Data Descriptions https://doi.org/10.1109/IEEEDATA.2025.3592888

[33] Sommer R, Paxson V (2010) Outside the closed world: On using machine learning for network intrusion detection. In: 2010 IEEE Symposium on Security and Privacy, pp 305–316, https://doi.org/10.1109/SP.2010.25

[34] Taleb T, Samdanis K, Mada B, et al (2017) On multi-access edge computing: A survey of the emerging 5g network edge cloud architecture and orchestration. IEEE Communications Surveys & Tutorials 19(3):1657–1681. https://doi.org/10.1109/COMST.2017.2705720

[35] Tavallaee M, Bagheri E, Lu W, et al (2009) A detailed analysis of the kdd cup 99 data set. In: 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA), pp 1–6, https://doi.org/10.1109/CISDA.2009.5356528