

Where Benchmark Intrusion Detection Meets Cloud Game Streaming: Two Generalisation Gaps in a Deep-Learning Detector

Bogdan Okreša Đurić, Lovro Đurđević

University of Zagreb Faculty of Organization and Informatics

Artificial Intelligence Laboratory

Pavlinska 2, 42000 Varaždin, Croatia

dokresa@foi.unizg.hr, ldurdevic22@student.foi.hr

Abstract. *Cloud game-streaming services such as GeForce NOW are high-value, latency-sensitive targets, making learned intrusion detection attractive. This paper asks whether a benchmark-trained detector can serve that purpose, using a single dense neural network as a fixed instrument. The model is evaluated at three removes from its NSL-KDD training distribution: a validation split, the held-out KDDTest+ partition, and game-streaming traffic. Validation accuracy above 0.99 falls to 0.776 on KDDTest+, concentrated in minority attack classes, and the benchmark feature schema does not map onto encrypted streaming traffic. Class imbalance and feature-representation mismatch, not model capacity, prevent transferring benchmark-trained detectors to streaming infrastructure.*

Keywords. cloud game streaming, network intrusion detection, deep learning, NSL-KDD, generalisation, feature extraction

1 Introduction

Cloud game-streaming services, in which a game runs on remote infrastructure and is delivered to the player as an interactive video stream, have grown into a significant class of online services. Platforms such as NVIDIA GeForce NOW, Xbox Cloud Gaming, and Amazon Luna run games on multi-tenant servers and stream the rendered output to thin clients over the Internet. The traffic they generate is continuous, high in bitrate, and acutely sensitive to latency and loss, since even a small disruption is immediately visible to the player as stutter or lag. These same properties make these services attractive targets: the gaming sector is consistently among the most heavily targeted by distributed denial-of-service campaigns (Yoachimik et al., 2026).

It is natural to ask whether the learned intrusion detectors that perform so well on standard benchmarks

could be applied to protect such infrastructure. Network intrusion detection has long been treated as a classification problem suitable for machine learning. Among the public benchmarks, the NSL-KDD dataset (Tavallaee et al., 2009) remains one of the most widely used, with a substantial body of work reporting classification accuracy exceeding 99 % on it. Taken at face value, such figures suggest that a capable detector is easily obtained and might simply be pointed at a new domain.

This paper tests that suggestion and finds it wanting, in two distinct ways. Rather than proposing a new architecture or pursuing an improved accuracy figure, it takes a single, deliberately modest deep neural network and treats it as a fixed instrument, observing its behaviour in three different scenarios based on the data on which it was trained. The first is an in-distribution validation split of the training partition. The second is the NSL-KDD test partition, which is constructed to differ from the training partition in attack difficulty and class composition (Tavallaee et al., 2009). The third is deployment against game-streaming traffic, captured both from a self-hosted streaming stack and from a commercial service. The three evaluation points yield two gaps in performance, which we characterise and then diagnose.

The four attack categories of NSL-KDD map directly onto threats to game-streaming infrastructure, which is what makes the benchmark a reasonable starting point for this research. Denial of service corresponds to the attacks of large volume that routinely disrupt latency-sensitive game services; reconnaissance (Probe) corresponds, e.g. to scanning of the streaming edge; remote-to-local (R2L) corresponds to account takeover, which on these platforms exposes payment details and game libraries; and user-to-root (U2R) corresponds, in a multi-tenant cloud setting, to tenant-isolation breakout, the escalation from a guest context to the shared host. The mapping is clean at the level of threat categories, which is precisely why the failure documented below is informative rather than expected: the challenge is not that the threats differ, but that the

traffic does.

The contribution of the paper is therefore methodological rather than architectural. It offers an account of how the apparent effectiveness of one benchmark-trained detector degrades as the evaluation moves towards a realistic target, together with a diagnosis of the two causes responsible. The first gap is attributable to severe class imbalance in the benchmark; the second to a mismatch between the connection-record features the model was trained on and the continuous media traffic that game streaming produces. The second gap is the more significant for the streaming setting, and the more general, since it is not concerned with any particular model.

The remainder of the paper is organised as follows. Section 2 reviews the relevant background and related work, including the traffic characteristics of cloud gaming. Section 3 describes the model, the data, the three evaluation points, and the deployment. Section 4 reports the results and the two gaps. Section 5 discusses what binds the detector’s performance and what a more informative evaluation protocol would require. Section 6 concludes and outlines future work.

2 Background and Related Work

2.1 The NSL-KDD Dataset

The NSL-KDD dataset was introduced as a refinement of the KDD Cup ’99 dataset, which had served for some time as a standard reference in intrusion-detection research (Tavallaee et al., 2009). The principal defect of KDD Cup ’99 was a large number of duplicate records in the training set, which biased models towards frequently occurring patterns and produced optimistic evaluation results. NSL-KDD removes the redundant records and selects samples across a range of classification difficulty, rendering a more balanced and more varying benchmark. Each record describes a single network connection through forty-one features that capture properties such as connection duration, protocol, the number of bytes transferred, and statistical indicators of network behaviour, and is labelled as normal traffic or as one of four attack categories: denial of service (DoS), reconnaissance (Probe), remote-to-local (R2L), and user-to-root (U2R).

For the purposes of this paper, the most important property of NSL-KDD is that its test partition, KDDTest+, is deliberately constructed to differ from the training partition, KDDTrain+, in the difficulty and representation of records (Tavallaee et al., 2009). A model evaluated on KDDTest+ is therefore not being asked simply to reproduce the observations applicable to the data it was trained on, and a decline in performance between an in-distribution split and KDDTest+ is a genuine indicator of limited generalisation rather than an artefact of the evaluation. It should also be noted here that NSL-KDD derives from network traffic

captured in the late 1990s and is, by common acknowledgement, no longer representative of modern network conditions. This limits the external validity of any result obtained on it, as is pointed out again in Section 5.

2.2 Saturation of Single-Dataset Results

Deep-learning classifiers reporting accuracy above 99 % on NSL-KDD are numerous, and the four-class taxonomy used here has been revisited many times. A recurring feature of the applicable literature is that this figure is obtained on a random split of the training partition rather than on the separate test partition, so that training and evaluation data share a distribution. Such a figure, therefore, measures the capacity of a model to fit a distribution and conveys little about its behaviour on data drawn from elsewhere. This paper treats the difference between the in-distribution figure and the separated figure as the object of study, rather than as a simple, obvious observation, or a result that should be minimised.

2.3 Cross-Dataset Generalisation

A smaller but established line of work has examined how intrusion-detection models transfer across datasets. The consistent finding is that models scoring near-perfectly within a single dataset degrade sharply, often towards chance, when evaluated on a different one. D’Hooge et al. (2020) train a range of supervised algorithms on one dataset and evaluate them on another, restricting attention to attack classes common to both, and report that the classifiers do not perform adequately on traffic from networks unseen at training time. Verkerken et al. (2022) report a comparable degradation for unsupervised methods. Related work on the statistical properties of synthetic intrusion-detection datasets (Layeghy et al., 2021) and on standardising feature sets across datasets (Sarhan et al., 2022) situates the problem in the lack of correspondence between datasets’ feature spaces and traffic characteristics. The work reported here is complementary to this line: rather than transferring a model from one labelled dataset to another, it confronts a benchmark-trained model with a class of real traffic, game streaming, that the benchmark schema was never designed to describe, which makes the feature-space mismatch concrete rather than statistical.

2.4 Cloud Game-Streaming Traffic

The properties of the traffic generated by cloud game-streaming services make it inappropriate for the benchmark schema. Di Domenico et al. (2021) analyse Stadia, GeForce NOW, and PS Now, and find that GeForce NOW and Stadia carry their video content over RTP, with bitrates reaching tens of megabits per second depending on resolution and network conditions. A subsequent measurement study of GeForce NOW (Lyu et

al., 2024) provides a detailed structural overview of its service and gameplay flows and extends the analysis to Xbox Cloud Gaming. Typically, it is described as a sustained, high-rate, latency-sensitive flow of encrypted media, dominated downstream by the video stream and accompanied upstream by a thin control channel. This is quite a different structure from the discrete, short-lived, service-typed connection that the NSL-KDD feature schema presupposes, and Section 4 shows that the difference is what defeats the trained model.

2.5 Class Imbalance in Intrusion Detection

The R2L and U2R categories are severely under-represented in NSL-KDD, and the difficulty of detecting them is well documented. A range of solutions has been proposed, including oversampling of the minority classes (Chawla et al., 2002), cost-sensitive training, and related techniques. Per-class reporting of precision, recall, and the F1 measure is an expected practice where minority detection is at issue (Sokolova & Lapalme, 2009). The minority-class collapse observed in Section 4 is, in this sense, a known occurrence. Therefore, the contribution of this paper is not to rediscover it, but to place it within an account of how benchmark accuracy might not represent reality, i.e. how it may exaggerate real-life results.

3 Method

The study uses a single deep neural network, held fixed throughout, as an instrument for measuring generalisation. This section describes the model and its training, the data and its preparation, the three evaluation points, and the deployment in which the second gap is observed.

3.1 Model and Training

The model is a sequential, fully connected network with three hidden layers of 128, 64, and 32 units. The hidden layers use the rectified linear activation, and dropout with rates of 0.3 and 0.2 is applied after the first and second hidden layers respectively as regularisation (Srivastava et al., 2014). The output layer has five units with a softmax activation, producing a probability distribution over the five classes. The network is trained with the Adam optimiser (Kingma & Ba, 2015) and the categorical cross-entropy loss, with a batch size of 128. Training is permitted for up to fifty epochs but is governed by early stopping on the validation loss, with patience of five epochs and restoration of the best weights. In practice, though, training halts after twenty-two epochs. The network has 26 117 trainable parameters. The architecture is deliberately modest and is not varied: because the purpose of the study is to assess generalisation rather than to maximise ac-

curacy. Therefore, holding the model fixed is a feature of the design rather than a limitation of it.

3.2 Data and Preparation

Training uses the KDDTrain+ partition of NSL-KDD, comprising 125 973 records, and evaluation uses the KDDTest+ partition, comprising 22 544 records. The original fine-grained attack labels are mapped to the five-class scheme described in Section 2. Three columns are removed before training: the original label, the difficulty indicator, and num_outbound_cmds, whose values are uniformly zero and therefore carry no information. This leaves 40 input features, of which three (protocol_type, service, and flag) are categorical. The categorical features are one-hot encoded and the numeric features are scaled to the unit interval; one-hot encoding expands the input to 121 dimensions, which constitutes the final input to the network.

The distribution of classes in KDDTrain+ is given in Table 1. The imbalance is pronounced: the two minority classes together account for fewer than one per cent of the training records, against more than 90 % for the normal and DoS classes combined. This imbalance is the cause of the first generalisation gap reported below.

Table 1. Class distribution in the KDDTrain+ partition

Class	Records
Normal	67 343
DoS	45 927
Probe	11 656
R2L	995
U2R	52
Total	125 973

3.3 The Three Evaluation Points

The model is evaluated at three points, defined exactly so that the two gaps between them are clear.

In-distribution validation. During training, twenty per cent of KDDTrain+ is held out as a validation split. Because this split is drawn from the same distribution as the training data, performance on it measures how well the model fits that distribution. This is the figure that reaches above 0.99 and corresponds to what much of the literature reports.

Separate benchmark. The model is evaluated on KDDTest+, which differs from KDDTrain+ in attack difficulty and class composition (Tavallaee et al., 2009). This is the first test of generalisation.

Streaming-traffic deployment. The same trained model is confronted with cloud game-streaming traffic, captured both from a self-hosted streaming stack and from a commercial service. This is the second test of generalisation, and it exposes a different failure from the first.

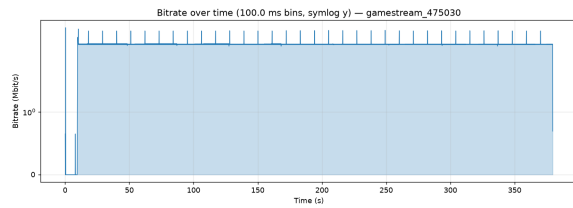


Figure 1. The constant bitrate with an occasional spike, of the self-hosted capture

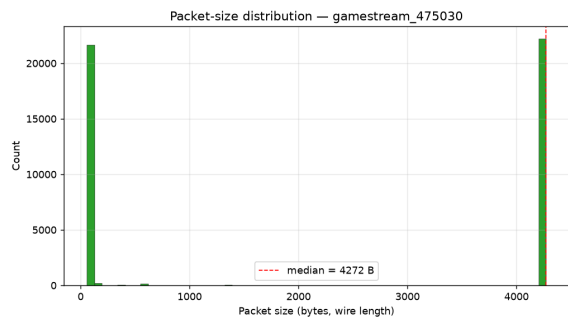


Figure 2. Distribution of packet size of the self-hosted capture

3.4 Deployment and Streaming Traffic

To confront the trained model with realistic streaming traffic, two captures are used:

1. The first is from a self-hosted streaming stack, with Sunshine as the host and Moonlight as the client, communicating over the GameStream protocol. This stack is fully under our control, can be captured at the interface between host and client, and is reproducible by others.
2. The second is a capture of a commercial GeForce NOW session taken at the client.

The self-hosted capture serves as the primary, reproducible exhibit, while the GeForce NOW capture indicates that the result holds on a real service. Both captures record traffic shape only, i.e. packet headers, sizes, timing, direction, and protocol; payloads are encrypted and are neither inspected nor retained.

The traffic of the self-hosted capture (Sunshine as the host and Moonlight as the client) was being captured for a total of 379 seconds, wherein a total of 44 368 packages were captured, carrying a total of 96 674 250 bytes. Out of the captured packets, 21 473 were UDP packets (14 800 RTP packets), and 711 were TCP packets. During the captured period, after the connection was established, the bitrate was constant, with an occasional spike, as seen in Fig. 1. Although there are many small packages, the median size of a package is 4 272 bytes, as shown in Fig. 2.

The captured traffic of a GeForce NOW session shows a similar context, although with slightly different values. The session settings at the time of the cap-

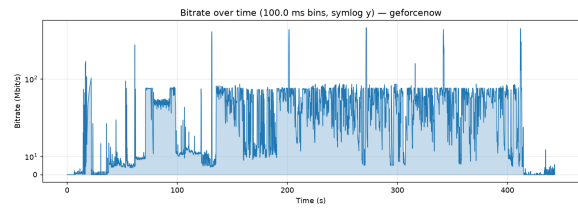


Figure 3. The varying bitrate is almost constant while the player or the scene of the GeForce NOW capture were moving

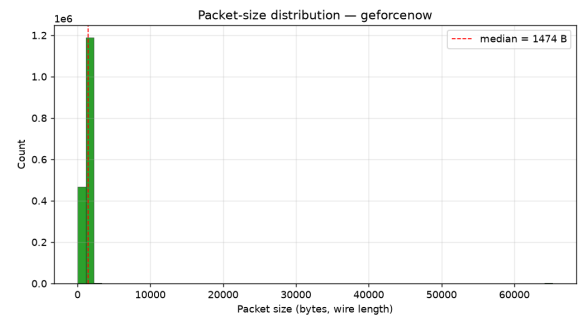


Figure 4. Distribution of packet size of the GeForce NOW capture

ture include the resolution of 1920×1080 , with automatic maximum bitrate, balanced streaming quality mode, over an Internet connection providing roughly 300 Mbps download and 150 Mbps upload speeds, running on a Lenovo ThinkBook 15 G2 ITL within the client version 2.0.86.129/ro running Escape Simulator 2 by Pine Studio.

The traffic of the GeForce NOW capture was being captured for a total of 442 seconds, wherein a total of 1 662 788 packages were captured, carrying a total of 2 128 208 510 bytes. Out of the captured packets, 816 071 were UDP packets (723 907 RTP packets), 15 028 were TCP packets, and 29 ICMP packets. During the captured period, after the connection was established, the bitrate was varying, but always high while the player or the scene were moving, as seen in Fig. 3. The packages during this capture were much smaller, as seen in Fig. 4, where the median packet size is 1 474 bytes, although their count is vastly greater than the median packets in the self-hosted capture.

The two packet-size distributions are not measured from the same position, and the difference in their medians (4 272 bytes for the self-hosted capture against 1 474 bytes for GeForce NOW) reflects the point of capture rather than a property of the traffic. The GeForce NOW session was captured at the wireless client, where packets appear at their on-wire size, so its median sits just below the Ethernet maximum transmission unit of roughly 1 500 bytes. The self-hosted session was captured at the interface between host and client, where segmentation offload coalesces several outgoing packets into larger units before they are

recorded, wherefore its distribution reflects host-side segments rather than on-wire frames. This caveat applies only to the packet-size comparison. The bitrate and total-volume figures aggregate bytes over time and are independent of how the traffic is segmented, so they remain directly comparable across the two captures. Where on-wire packet size is the quantity of interest, the GeForce NOW capture is the faithful measurement.

3.5 Mapping the Feature Schema onto Streaming Traffic

The model expects an input vector derived from the NSL-KDD connection-record schema, in which a single record describes one discrete, service-typed connection and several features are statistics aggregated over a window of recent connections, for instance counts of connections to the same host or service within a two-second interval. Game-streaming traffic does not present this structure. A streaming session is a small number of long-lived, high-rate media flows rather than a sequence of discrete connections. It is carried over UDP and so has no TCP connection-state flag, and it corresponds to no entry in the benchmark's enumeration of application services. The question for deployment is therefore not how accurately the model classifies streaming traffic, but whether the features it requires can be constructed from such traffic at all.

To answer this, each of the 41 NSL-KDD features is assessed against the captured streaming traffic and classified as derivable, degenerate, or inapplicable. A feature is derivable if it can be computed and carries discriminative information; degenerate if it can be computed but is reduced to a constant for streaming traffic, as the content features describing events within an interactive login session do; and inapplicable if the unit it depends on does not exist in the traffic, as is the case for the service-typed and TCP-flag features and for the windowed counts of repeated discrete connections.

Of the 41 features, four are derivable, fifteen degenerate (including `num_outbound_cmds`, which is constant in NSL-KDD and removed during preprocessing), and twenty-two inapplicable, as set out in Table 2. Close to nine in ten features therefore carry no information or cannot be honestly constructed for a media stream. The four features that do persist are dominated by the two byte-count features, i.e. by data volume, which is the signature that the model associates with the signature the model associates with denial of service.

The effect is more pronounced once the one-hot encoding is considered. The 121-dimensional input comprises the 37 numeric features and the expansion of the three categorical features; with `flag` contributing eleven columns and `protocol_type` three, the `service` feature accounts for the remaining seventy. Both `service` and `flag` are inapplicable to streaming traffic, so together they occupy 81 of the 121 input di-

mensions. Two-thirds of the vector the model actually receives is thus structurally meaningless for a media stream before the degenerate features are even considered.

4 Results

The three evaluation points are reported in turn, exposing the two gaps identified earlier.

4.1 In-Distribution Performance

During training, validation accuracy rises quickly within the first few epochs and stabilises above 0.99, where it remains for the rest of training; the validation loss declines and then oscillates mildly, while the training loss declines steadily. Figure 5 shows the accuracy and loss across the twenty-two epochs of training. Taken on its own, this figure would suggest a highly capable intrusion detection model, and it is consistent with the results commonly reported for NSL-KDD.

The closeness of the training and validation curves is sometimes read as evidence that a model has not overfitted. That interpretation is correct only in a narrow sense and is worth examining carefully. Because the validation split is drawn from the training distribution, the small gap shows that the model fits that distribution well and has not overfitted to the training sample in particular. On the other hand, it says nothing about how the model will behave on data constructed to differ from the training distribution. The above-described can be observed as a challenge in generalisation, rather than being a sign of an incapable model. Establishing this distinction is what licenses describing the subsequent fall as a generalisation gap rather than a failure of model capacity.

4.2 First Gap: Held-Out Benchmark Performance

On KDDTest+, overall accuracy is 0.776, a fall of more than 20 % from the in-distribution figure. The per-class results are given in Table 3. The fall is not distributed evenly across the classes: it is most visible in the two minority classes. DoS and Probe retain serviceable F1 measures of 0.89 and 0.77, and the normal class is recovered with a recall of 0.97. R2L and U2R, by contrast, attain F1 measures of only 0.12 and 0.09, driven by recall of 0.07 and 0.04 respectively. This can be interpreted as follows: the model misses roughly 19 of every 20 R2L attacks and 24 of every 25 U2R attacks.

The confusion matrix in Table 4 shows where the missed attacks are sent. The overwhelming majority of R2L and U2R records are classified as normal traffic: 2572 of 2754 R2L records and 174 of 200 U2R records are predicted as normal. This is the most damaging failure mode available to the trained model because an attack classified as normal produces no

Table 2. Mapping of the 41 NSL-KDD features onto steady-state game-streaming traffic. Group: B basic, C content, T time-based traffic, H host-based traffic. Class: Der. derivable, Deg. degenerate, Inap. inapplicable.

#	Feature	Group	Class	#	Feature	Group	Class
1	duration	B	Der.	22	is_guest_login	C	Deg.
2	protocol_type	B	Der.	23	count	T	Inap.
3	service	B	Inap.	24	srv_count	T	Inap.
4	flag	B	Inap.	25	serror_rate	T	Inap.
5	src_bytes	B	Der.	26	srv_serror_rate	T	Inap.
6	dst_bytes	B	Der.	27	rerror_rate	T	Inap.
7	land	B	Deg.	28	srv_rerror_rate	T	Inap.
8	wrong_fragment	B	Deg.	29	same_srv_rate	T	Inap.
9	urgent	B	Inap.	30	diff_srv_rate	T	Inap.
10	hot	C	Deg.	31	srv_diff_host_rate	T	Inap.
11	num_failed_logins	C	Deg.	32	dst_host_count	H	Inap.
12	logged_in	C	Deg.	33	dst_host_srv_count	H	Inap.
13	num_compromised	C	Deg.	34	dst_host_same_srv_rate	H	Inap.
14	root_shell	C	Deg.	35	dst_host_diff_srv_rate	H	Inap.
15	su_attempted	C	Deg.	36	dst_host_same_src_port_rate	H	Inap.
16	num_root	C	Deg.	37	dst_host_srv_diff_host_rate	H	Inap.
17	num_file_creations	C	Deg.	38	dst_host_serror_rate	H	Inap.
18	num_shells	C	Deg.	39	dst_host_srv_serror_rate	H	Inap.
19	num_access_files	C	Deg.	40	dst_host_rerror_rate	H	Inap.
20	num_outbound_cmds	C	Deg.	41	dst_host_srv_rerror_rate	H	Inap.
21	is_host_login	C	Deg.				

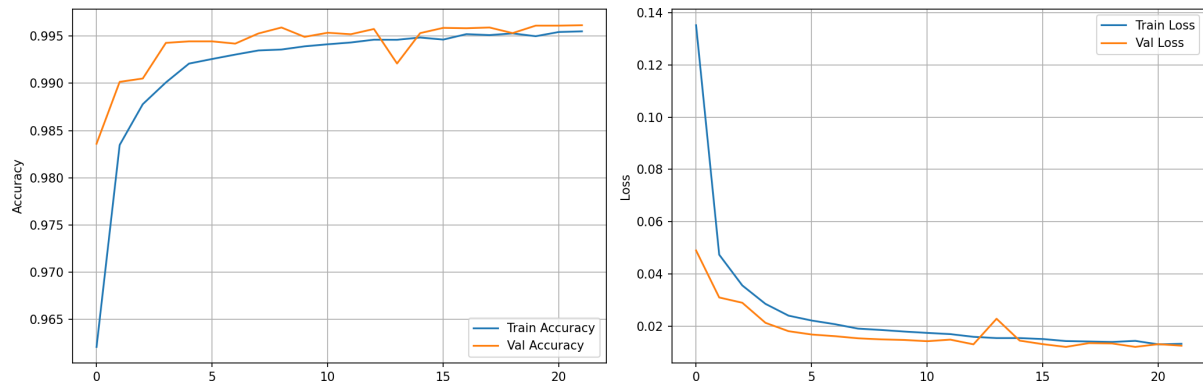


Figure 5. Accuracy and loss over the training epochs

Table 3. Per-class results on KDDTest+

Class	Prec.	Rec.	F1	Samples
DoS	0.96	0.83	0.89	7 458
Probe	0.87	0.70	0.77	2 421
R2L	0.96	0.07	0.12	2 754
U2R	0.90	0.04	0.09	200
Normal	0.68	0.97	0.80	9 711

Table 4. Confusion matrix on KDDTest+; rows are true classes, columns are predicted classes

	DoS	Pr.	R2L	U2R	Norm.
DoS	6 169	22	2	0	1 265
Probe	210	1 695	1	0	515
R2L	0	2	180	0	2 572
U2R	0	14	3	9	174
Normal	54	222	1	1	9 433

alert and passes silently. The pattern of high precision alongside very low recall for these classes is itself informative. When the model does assign a record to R2L or U2R it is usually correct, but it does so very rarely. The above-stated is the signature of a model that has seen too few examples of a class to form a usable decision rule, and it follows directly from the training distribution in Table 1, in which R2L is represented by 995 records and U2R by 52.

4.3 Second Gap: Streaming-Traffic Deployment

The second gap appears when the same model is confronted with game-streaming traffic, and it is of a different kind from the first. The first gap concerns attacks the model was never given enough examples of to learn; the second concerns traffic the model cannot be given at all because the feature schema it requires has

no coherent mapping onto a continuous media stream.

The mapping of Section 3.5 is itself the result. Of the 41 NSL-KDD features, only four can be faithfully derived from a streaming flow, fifteen collapse to constants, and twenty-two depend on a structure: a named service, a TCP connection state, or a population of repeated discrete connections, that a single sustained media flow does not possess. The model cannot be supplied with the inputs it was trained on because those inputs do not exist in this traffic.

The practical consequence follows from which features survive. When only the volume and rate features carry real information, a sustained high-bitrate stream presents the same signature the model associates with flooding, and benign streaming traffic is liable to be classified as a denial-of-service attack. This is the worst available outcome for the streaming setting, where a false positive on legitimate play is itself a denial of service. The failure is not a deficiency of the model, but of the attempt to express streaming traffic in a representation built for late-1990s connection records.

5 Discussion

The two gaps point to a single conclusion: a benchmark-trained detector is bound by class imbalance and by feature representation, not by model capacity, and neither obstacle would be removed by a larger or deeper neural network. At the first gap, the minority classes are represented by too few examples for any model to learn a reliable boundary from them, and additional statistical power applied to the same data would not supply the missing examples. At the second gap, the inputs the streaming setting can offer do not match the representation the model requires; a more expressive model would only fit the benchmark representation more closely and would not reconcile the two. For the streaming target specifically, the second gap is essential: it is not that the model classifies streaming traffic poorly, but that the traffic cannot be faithfully expressed in its required input schema in the first place.

This has a direct effect on the appeal of reusing benchmark-trained detectors to protect game-streaming infrastructure. The threat categories map cleanly onto that setting, which is what makes the idea tempting, but the mapping of threats is not a mapping of traffic. A detector trained on connection-record features cannot be pointed at a continuous encrypted media stream and expected to function, and the volume-and-rate features that do survive the transition are precisely the ones that cause benign streaming traffic to resemble a flood. Protecting streaming infrastructure with learned detection would require features defined natively over media flows, such as bitrate stability, packet-size and inter-arrival distributions, and flow-level rather than connection-level statistics, and train-

ing data drawn from such traffic rather than from a connection-oriented benchmark.

The wider methodological point is that results reporting only in-distribution accuracy, as found in much of the NSL-KDD literature, systematically overstate operational capability because they reflect the one quantity least likely to fail. More informative evaluation reports featuring separated performance with per-class metrics would render a minority-class collapse unable to hide inside a favourable average. What is more, where deployment against a particular kind of traffic is considered, an explicit account should be given of whether the feature schema can represent the applicable traffic at all. The feature-extraction path, which is rarely documented, turns out to be decisive and deserves to be reported as a first-class part of the model training process.

Several limitations bound the above proposals. The study uses a single model and a single benchmark; NSL-KDD is an ageing dataset whose traffic is not representative of present-day networks, so the benchmark figures should be read as illustrative. The GeForce NOW capture is a single-user convenience sample that illustrates the representation mismatch rather than characterising the service, and the self-hosted stack stands in for commercial platforms whose proprietary protocols and infrastructure differ. These limitations constrain the generality of the figures but not the logic of the result, which concerns the structure of the evaluation and the accurate representation of the traffic rather than the score of any particular model. The two captures are also taken from different vantage points, so their packet-size distributions are not strictly comparable, as noted in Section 3; their volume and bitrate characteristics are unaffected.

6 Conclusion and Future Work

This paper used a single, fixed deep neural network as an instrument to ask whether a benchmark-trained intrusion detector can be applied to cloud game-streaming traffic. An in-distribution validation accuracy above 0.99 falls to 0.776 on the held-out KDDTest+ partition, with the fall concentrated in the minority R2L and U2R classes, and the NSL-KDD feature schema is shown to have no coherent mapping onto continuous encrypted streaming traffic, with only the volume-and-rate features surviving the move and those causing benign streaming traffic to resemble a denial-of-service attack. The first gap is attributable to class imbalance and the second to the mismatch between the connection-record format the model requires and the media-flow traffic streaming produces. The binding constraints are the data and its representation, not the capacity of the model, and the clean mapping of threat categories onto the streaming context does not carry a mapping of traffic with it.

Several directions are left for future work, named

here so that their absence is understood as a matter of scope. The first is the construction of a model over features defined natively for media flows, trained on game-streaming traffic rather than on a connection-oriented benchmark. The captures and feature analysis reported here are a step towards specifying such features. The second is the mitigation of the minority-class collapse through oversampling, class weighting, or cost-sensitive training (Chawla et al., 2002), evaluated honestly on the selected partition rather than on a resampled test set. The third is the use of modern, flow-based datasets such as CICIDS2017 (Sharafaldin et al., 2018) or UNSW-NB15 (Moustafa & Slay, 2015), whose flow-level features are closer to what a streaming-aware detector would require and whose reconciliation with older schemas remains an open problem (Sarhan et al., 2022).

References

- Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16, 321–357. <https://doi.org/10.1613/jair.953>
- D’Hooge, L., Wauters, T., Volckaert, B., & De Turck, F. (2020). Inter-dataset generalization strength of supervised machine learning methods for intrusion detection. *Journal of Information Security and Applications*, 54, 102564. <https://doi.org/10.1016/j.jisa.2020.102564>
- Di Domenico, A., Perna, G., Trevisan, M., Vassio, L., & Giordano, D. (2021). A Network Analysis on Cloud Gaming: Stadia, GeForce Now and PSNow. *Network*, 1(3), 247–260. <https://doi.org/10.3390/network1030015>
- Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization [arXiv:1412.6980]. *3rd International Conference on Learning Representations (ICLR)*.
- Layeghy, S., Gallagher, M., & Portmann, M. (2021). Benchmarking the benchmark – analysis of synthetic NIDS datasets [arXiv preprint arXiv:2104.09029].
- Lyu, M., Madanapalli, S. C., Vishwanath, A., & Sivaraman, V. (2024). Network Anatomy and Real-Time Measurement of Nvidia GeForce NOW Cloud Gaming. In P. Richter, V. Bajpai, & E. Carisimo (Eds.), *Passive and Active Measurement: 25th International Conference, PAM 2024, Virtual Event, March 11–13, 2024, Proceedings* (Vol. 1). Springer Nature Switzerland. <https://doi.org/10.1007/978-3-031-56249-5>
- Moustafa, N., & Slay, J. (2015). UNSW-NB15: A comprehensive data set for network intrusion detection systems. *2015 Military Communications and Information Systems Conference (MilCIS)*, 1–6. <https://doi.org/10.1109/MilCIS.2015.7348942>
- Sarhan, M., Layeghy, S., Moustafa, N., & Portmann, M. (2022). Towards a standard feature set for network intrusion detection system datasets. *Mobile Networks and Applications*, 27(1), 357–370. <https://doi.org/10.1007/s11036-021-01843-0>
- Sharafaldin, I., Lashkari, A. H., & Ghorbani, A. A. (2018). Toward generating a new intrusion detection dataset and intrusion traffic characterization. *Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP)*, 108–116. <https://doi.org/10.5220/0006639801080116>
- Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4), 427–437. <https://doi.org/10.1016/j.ipm.2009.03.002>
- Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1), 1929–1958.
- Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009). A detailed analysis of the KDD CUP 99 data set. *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA)*, 1–6. <https://doi.org/10.1109/CISDA.2009.5356528>
- Verkerken, M., D’hooge, L., Wauters, T., Volckaert, B., & De Turck, F. (2022). Towards model generalization for intrusion detection: Unsupervised machine learning techniques. *Journal of Network and Systems Management*, 30(1), 12. <https://doi.org/10.1007/s10922-021-09615-7>
- Yoachimik, O., Pacheco, J., & Cloudforce One. (2026, February). *2025 Q4 DDoS Threat Report: A Record-Setting 31.4 Tbps Attack Caps a Year of Massive DDoS Assaults*. Retrieved September 9, 2026, from <https://blog.cloudflare.com/ddos-threat-report-2025-q4/>