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.

Informer Encoded Gradient Boosting for Explainable Network Intrusion Detection with Ablation and SHAP Analysis

Mohammed Ziadul Amin
East Delta University
Imtiaz Akber Chowdhury
imtiaz.c@eastdelta.edu.bd

East Delta University

Research Article

Keywords:

Posted Date: November 26th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11663541/v1

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

Additional Declarations: No competing interests reported.

Informer Encoded Gradient Boosting for Explainable Network Intrusion Detection with Ablation and SHAP Analysis

Mohammed Ziadul Amin[1], Imtiaz Akber Chowdhury[2*]
[1]Department of Computer Science and Engineering, East Delta University, Chattogram, Bangladesh
[2]Department of Electronic and Telecommunication Engineering, East Delta University, Chattogram, Bangladesh
[*]Correspondence: Imtiaz.c@eastdelta.edu.bd

Abstract

Hybrid intrusion detectors that combine a neural encoder with a tree ensemble are often judged by accuracy and seldom explained at the pipeline level. This study examines one such pipeline: an Informer-style encoder converts each flow into 57 tokens, one per flow statistic, and XGBoost classifies the encoder's frozen 64-dimensional output. No intrusion-detection study identified in the literature places a frozen per-feature Informer encoder beneath a boosted-tree head and varies the head alone on balanced multiclass flows. Balanced samples are obtained by under sampling 9,167,581 cleaned CIC flow records, so no flow is synthetic. In the binary study, 97.24% of 600,000 test flows are classified correctly (Matthew's Correlation Coefficient, MCC = 0.9458), with 1,751 false alarms versus 7,256–11,371 for four ablation variants. In the eight-class study, the pipeline reaches 90.28% accuracy and a macro-F1 of 0.9012. With one shared encoder, the head alone raises accuracy from 50.41% (linear) to 77.61% (SVM) and 90.28% (XGBoost); a Random Forest on raw features reaches 85.83%, and Benign–Infiltration confusion accounts for 355 of the 526 errors. Permutation Shapley Additive Explanations (SHAP) on the whole pipeline, cross-checked against a random ranking and 29 groups of correlated features, assigns 36.6% of attribution to five features. The two TCP initial-window fields carry 14.8% overall and 28.2% in Webattack, and a leave-one-out lookup on these two fields alone labels 74.1% of the 324 explained flows correctly; testbed fingerprinting is proposed as an untested hypothesis. All scores derive from a single seed, a random split, and three epochs of encoder training.

1. Introduction

Flow-based network intrusion detection classifies summary statistics of each connection instead of its payload, and the public flow collections released by the Canadian Institute for Cybersecurity (CIC) have become a default testbed for learning-based detectors [1-4]. Accuracies reported on them are often very high, but the numbers depend on choices that are easy to get wrong. In a study, it is [5] found faults in how the flows of CIC-IDS2017 were built and labeled, and it is reported that correcting such faults makes a significant difference to measured detection performance [6]. The broader pattern is documented in another study, in which cataloged recurring pitfalls across the life cycle of learning-based security systems [7].

Flow statistics are tabular data, and on tabular data tree ensembles remain hard to beat with neural networks [8, 9], although giving every numerical feature its own token and applying attention yields a deep model that is competitive on many task [10]. Attention models have been adapted to flow records, for example in the FlowTransformer framework [11], but a recent leakage-aware re-evaluation on CIC-IDS2017 found that the padding convention changed a Transformer's score more than its architecture did, and that a class-balanced Random Forest was the most robust model of those tested [12]. A different route keeps a tree ensemble as the classifier and uses a neural or unsupervised model only to build the representation, as in the stacked clustering features in a study [13] and the convolutional embedding in another study [14].

Explaining such a hybrid is not straightforward. SHapley Additive exPlanations (SHAP) [15] and its tree-specific algorithm, TreeSHAP [16], attribute a prediction to the inputs of the function they are given, so applied to the classifier of a hybrid they attribute it to latent dimensions that have no name, and another study state that such latent-space attributions are indirect with respect to the traffic variables [14]. A model-agnostic Shapley estimator [17] can explain the whole pipeline in terms of the original flow features, at a far higher cost. Whether the resulting attributions deserve trust is a further question: it proposes explicit criteria for explanation methods in security, such

as descriptive accuracy and stability [18], and another survey [19] lists explanation metrics among the open problems of explainable intrusion detection.

This paper studies one such hybrid in depth. An Informer-style encoder turns each flow into a sequence of 57 tokens [20], one per flow statistic, and is trained briefly and then frozen; XGBoost (extreme gradient boosting) [21] classifies its 64-dimensional output. The pipeline is evaluated on a consolidated collection of CIC flow records, balanced by exact under sampling so that every training and test flow is a real captured flow, in a binary study of 2,000,000 flows and an eight-class study of 18,040 flows. Each study contains an ablation: the binary one compares the pipeline with four other models, and the multiclass one holds a single trained encoder fixed and varies only the classification head. The multiclass model is then explained twice, with TreeSHAP on its latent dimensions and with permutation SHAP on the whole pipeline, and the explanations are examined with quantitative checks and not only with plots.

The contribution is a combination and a body of evidence, not a new architecture. No intrusion-detection study found in the literature places a frozen per-feature Informer encoder under a boosted-tree head and varies the head alone on balanced multiclass flows. The ingredients have precedent: boosted trees are fitted on learned or clustering-based features [13, 14, 22], and pair several learned representations with several heads on imbalanced NetFlow data [23]. The explanation of the whole pipeline in terms of 57 named flow features, checked against a random ranking and against groups of correlated features, is the second element. It exposes a result that accuracy alone hides: the two Transmission Control Protocol (TCP) initial-window fields carry 14.8% of the attribution over all explained flows and 28.2% in the Webattack class, and a lookup keyed only on those two fields labels 74.1% of the explained flows correctly. The pattern is consistent with a model that has learned host fingerprints of the testbed, which is stated here as a hypothesis and was not tested.

The proposed pipeline reaches 97.24% accuracy in the binary study and 90.28% in the eight-class study, ahead of every ablation variant and of a Random Forest on the raw features (85.83% on eight classes). The scores carry limits that are reported beside them. Everything rests on one seed and a random split, and the encoder was trained for only three epochs; the multiclass benign class is a thin sample, and no boosted-tree control on the raw features was run.

The remainder of this paper is organized as follows. Section 2 reviews related work. Sections 3 and 4 describe the data and the pipeline, Section 5 the evaluation protocol, and Sections 6–8 the binary and multiclass ablations. Section 9 explains the multiclass model, Section 10 discusses limitations, and Section 11 concludes.

2. Related work

The CIC releases record labeled attacks against traffic from simulated users. CIC-IDS2017 spans five days in July 2017; a behavior model generates benign activity for 25 users and attack tools generate the labeled traffic, from which CICFlowMeter extracts more than 80 flow features per connection [1, 24]. CSE-CIC-IDS2018 repeats the design on a larger network, with 50 attacking machines against 420 victim machines and 30 servers [25], and CIC-DDoS2019 adds reflection and exploitation attacks over two capture days [2, 26]. Surveys catalog these and other intrusion-detection datasets [3, 4].

The collections have documented defects. It is re-examined CIC-IDS2017 and traced errors in traffic generation [5], flow construction, feature extraction and labeling; for instance, the flow tool ended a TCP connection at the first FIN packet and split the remaining packets into separate flows, and attack labels came mainly from IP addresses and time windows. Their corrected processing reconstructed or relabeled more than 20% of the original traffic traces. It is independently report mislabeled attacks and capture faults, namely packet misordering and duplication, and report significant differences in detection performance once these are corrected [6]. Another study identifies recurring pitfalls across the life cycle of learning-based security systems and find them widespread in a study of 30 papers from top-tier security conferences [7].

The split protocol is one such concern. It is rebuilt CIC-IDS2017 as ordered flow sequences and report that random splits combined with repeat-last padding can overstate a model’s robustness by up to 0.24 macro-F1 [12]. The

collection used in this paper consolidates several CIC releases and shares these concerns, which Section 10 discusses.

2.1. Attention models, tabular learning and hybrid detectors

Self-attention [27] has reached flow records through several routes. FlowTransformer lets a user swap the input encoding, the transformer and the classification head and evaluates the combinations on NSL-KDD, UNSW-NB15 and CSE-CIC-IDS2018; its authors report that the classification head had the largest effect on performance and that global average pooling, common in text classification, performed very poorly [11]. It represents each flow as a multivariate time series for early detection and propose a Multi-Domain Transformer that adds frequency-domain information, reaching an 84.1% macro-F1 on SCVIC-TS-2022, a benchmark derived from CIC-IDS2017 [28]. Informer was designed for long-sequence time-series forecasting: a ProbSparse attention lowers the cost of long inputs and a distilling step halves the sequence between layers [20]. The encoder used here keeps the layer layout and the distilling step but computes exact attention over its 57 tokens (Section 4.1).

Outside security, giving each numerical feature a learned token, as the FT-Transformer does, makes attention applicable to tables and yields a deep baseline that is competitive on many tasks, although its authors find no universally superior solution [10]. Benchmarks keep finding tree-based models competitive on typical tabular data [8, 9], which is the setting of flow statistics.

Hybrid detectors keep a tree ensemble as the classifier. Balance UNSW-NB15, CIC-IDS2017 and CSE-CIC-IDS2018 by random oversampling, add stacked K-Means and Gaussian-mixture features, reduce the result to ten principal components and train decision trees, Random Forests, Extra Trees and XGBoost with ten-fold cross-validation [13]. In another study, it feeds the 128-dimensional embedding of a one-dimensional CNN to XGBoost on the CIC IoT-DIAD 2024 benchmark and report an accuracy of 0.9324 but a macro-F1 of 0.591 under a leakage-aware protocol, because minority classes score poorly; they apply SHAP to the embedding dimensions [14]. Another study combines five separately trained models, among them XGBoost and an autoencoder, by weighted soft voting after SMOTE [29, 30] on CIC-IDS2017, and the autoencoder serves as an anomaly detector through its reconstruction error, not as a feature extractor for a tree model.

Autoencoder latent have also been passed to XGBoost: combine an autoencoder with a Gaussian mixture model on NSL-KDD and UNSW-NB15 [22], and try several autoencoder types on TII-SSRC-23 [31]. A separate representation learning from classification on three NetFlow collections and pair four learned representations with four supervised heads [23]; no combination wins everywhere, and their subsampled data keep the original class imbalance. The collection used here avoids oversampling altogether: the classes are cut to a common size, so that every flow is a real captured flow (Section 3).

2.2. Explaining intrusion detectors

A study presents what they describe as the first use of SHAP to explain intrusion detectors: their framework gives local and global explanations for detectors trained on NSL-KDD and compares one-vs-all with multiclass classifiers [32]. combine SHAP, LIME (local interpretable model-agnostic explanations) [33] and three further methods around a deep network on the same dataset, and explain misclassifications by searching for the smallest feature changes that would correct them [34]. Another survey argues for black-box detectors with post-hoc explainers and lists explanation metrics among the open problems [19]. In a recent preprint, feeds the 72 CIC-IDS2017 flow features, grouped into four semantic families, to a hierarchical Transformer that gives every feature its own token and ends in a neural classification head, and explains it with gradient-based SHAP over the input features and with attention rollout [35].

Shapley-based attributions come in variants, and the variant matters when features are correlated, as flow statistics are. The path-dependent estimate of TreeSHAP and the interventional estimate of permutation-based explainers answer different questions [36-38], and list problems with reading Shapley values as feature importance [39]. In security, compared six explanation methods, SHAP among them, on malware-detection and vulnerability-discovery systems with criteria that include descriptive accuracy, sparsity, stability and efficiency, and found considerable differences between the methods [18]. The checks of Section 9.6 draw on this line of work: deletion and insertion

curves [40] and the Gini coefficient of sparsity [41] are supplemented by a perturbation test of the kind that Alvarez-Melis and Jaakkola [42] apply to interpretability methods.

Two gaps follow. As far as the literature reviewed here shows, the hybrids above are not explained at the level of the original flow features, and [14] say so for their own latent-space explanation. [19] also name metrics for evaluating explanations as an open need. The present study explains a hybrid in terms of its 57 input flow features on a CIC collection and tests the explanations against a random ranking. It also reports their cost and uses them to expose how heavily the model relies on the TCP initial-window fields.

3. Data and preprocessing

Both studies draw on one consolidated collection of Canadian Institute for Cybersecurity (CIC) flow records, stored as a single Parquet file and described by 57 CICFlowMeter-style flow statistics [43], and both balance it by exact under sampling instead of synthetic oversampling such as the synthetic minority over-sampling technique (SMOTE) [30]. [Authors to add: public source, version and citation of the consolidated Parquet collection.] After removal of rows with missing values the collection holds 9,167,581 flows, 78.4% of them benign (Table 1). Its fine-grained labels follow the attack catalogs of the CIC releases, for example Hulk and GoldenEye DoS or NTP and TFTP reflection floods [1, 2].

The coarse label column was mapped onto eight classes (Benign, DDoS, DoS, Botnet, Bruteforce, Infiltration, Webattack and Portscan; DoS and DDoS denote denial of service and distributed denial of service) with case-insensitive substring rules; all 9,167,581 flows matched a class. The rarest class, Portscan, has 2,255 flows, about 3,187 times fewer than the benign class. Each class was therefore cut to a common size with a seeded random ordering in PySpark, and every flow in every experiment is a real captured flow.

Table 1. Flow counts per class in the cleaned collection and in the two experimental splits.

Class	Cleaned collection	Multiclass train	Multiclass test	Binary train	Binary test
Benign	7,186,189	1,579	676	700,000	300,000
DDoS	1,234,729	1,579	676	436,589	187,113
DoS	397,344	1,578	677	140,324	59,710
Botnet	145,968	1,578	677	51,698	22,195
Bruteforce	103,244	1,578	677	36,263	15,741
Infiltration	94,857	1,579	676	33,277	14,419
Webattack	2,995	1,578	677	1,027	451
Portscan	2,255	1,579	676	822	371
Total	9,167,581	12,628	5,412	1,400,000	600,000

In the binary study the seven attack rows form one Attack class (700,000 training and 300,000 test flows) and only the benign-versus-attack label is used.

The binary study draws 1,000,000 flows from each side, giving 2,000,000 flows, and splits them 70/30 with stratification and seed 42 into 1,400,000 training and 600,000 test flows. The attack side inherits the family proportions of the collection: DDoS supplies 62.4% of the attack test flows and DoS 19.9%, whereas Portscan and Webattack supply 0.12% and 0.15%. Binary scores therefore describe volumetric attacks far more than rare ones. The multiclass study removes that dominance by keeping 2,255 flows per class, 18,040 in total, and splits them in the same way into 12,628 training and 5,412 test flows.

The classes stay internally heterogeneous. In the multiclass training split DDoS mixes 15 fine-grained labels, led by LOIC-HTTP with 677 of 1,579 flows (42.9%); DoS is 80.5% Hulk (1,271 of 1,578), Bruteforce is 94.4% SSH (1,489 of 1,578) and Webattack is 67.4% brute-force (1,063 of 1,578). Benign, Botnet, Infiltration and Portscan each carry a single fine-grained label.

Columns were cast to compact types in Spark (22 of the 57 needed re-casting), rows with missing values were dropped before sampling, and infinite values were converted to missing and dropped afterwards; the last step removed no flow in either study. The 57 features were then z-scored with scikit-learn's StandardScaler. The scaler

was fitted once on the complete balanced sample before the train-test split, so test-set means and standard deviations enter the scaling. This leaks only two moments per feature, but a scaler fitted on the training split alone is the cleaner protocol, and the point is listed among the limitations in Section 10.

4. Proposed pipeline and compared variants

The proposed detector trains an Informer-style encoder on the scaled flow features, freezes it, and classifies the encoder's 64-dimensional latent vector with XGBoost; every ablation variant swap one of those two stages. TreeSHAP (solid blue frame and arrow) explains the head on the latent dimensions, and permutation SHAP (dashed orange outline) explains the whole function from the 57 scaled inputs. Both explanations use the log-odds margin of the predicted class and were run on the multiclass model only. FFN, feed-forward network; GELU, Gaussian error linear unit; BN, batch normalization; ELU, exponential linear unit. Fig. 1 shows the pipeline and the two points at which the explanations of Section 9 attach to it.

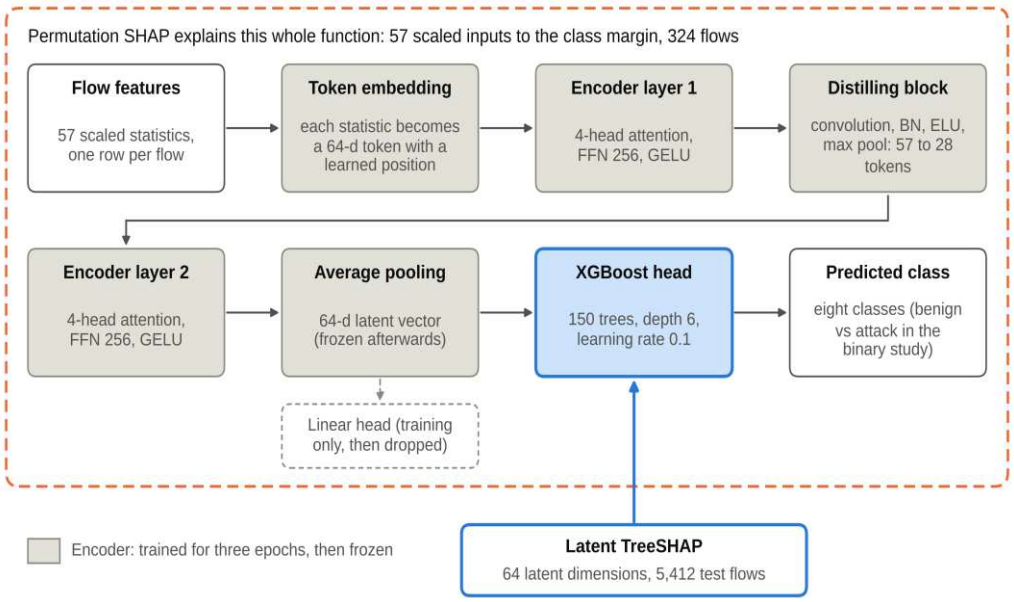


Fig. 1. Proposed Informer-XGBoost pipeline with the sizes of the multiclass study. Shaded boxes form the Informer-style encoder, which is trained for three epochs through a linear head and then frozen; the XGBoost head reads its 64-dimensional output.

4.1. Per-feature Informer encoder

Each scalar feature of a flow becomes its own token. A shared linear map lifts the value from one to 64 dimensions and a learned positional embedding is added, so a flow is a sequence of 57 tokens. Two encoder layers follow. Each applies four-head self-attention (head width 16), then a feed-forward block of width 256 with a Gaussian error linear unit (GELU) activation and dropout 0.1, with residual connections and post-layer normalization. Between the layers, a distilling block (one-dimensional convolution of width 3, batch normalization, an exponential linear unit (ELU) activation and max-pooling of stride 2) halves the sequence from 57 to 28 tokens, as in the original Informer [20]. Average pooling over the remaining tokens gives the 64-dimensional latent vector.

The encoder is Informer-style rather than a full Informer. The class named ***ProbSparseAttention*** in the code computes exact attention over all 57 tokens with PyTorch's fused scaled-dot-product routine, and its sparsity factor is never used. At this sequence length the score matrix has only 3,249 entries, so selecting a sparse subset of queries saves nothing; what the encoder keeps from the Informer is the layer layout and the self-attention distilling step.

For training, a linear head maps the latent vector to one logit (binary study, binary cross-entropy) or eight logits (multiclass study, cross-entropy with balanced class weights, which are close to 1 because the classes are balanced). Training uses Adam with a learning rate of 0.001, mini-batches of 1,024 flows and three epochs on the CPU, with seed 42. In the multiclass study this amounts to 13 mini-batches per epoch and 39 optimizer steps in total, and the training loss falls from 1.868 to 1.265 against a chance level of 2.079 (ln 8), with a running training accuracy of 57.5% in the last epoch [44]. The encoder is therefore only partly trained, a fact that conditions every comparison below.

4.2. XGBoost classifier

After training, the head is discarded and the encoder maps each flow to its latent vector in evaluation mode. XGBoost is fitted on the training latent with 150 trees, maximum depth 6, learning rate 0.1 and the histogram tree method; the multiclass model uses the softprob objective with multi-class log-loss and balanced sample weights [21]. No hyperparameter search or early stopping was used in any model, so the reported scores are those of one fixed configuration per variant.

4.3. Compared variants

Table 2 lists the five variants. The binary and multiclass studies share the names but not every implementation detail, and the differences matter for how the ablations can be read.

Table 2. Variants compared in the two studies. CNN-LSTM, convolutional neural network with long short-term memory; SVM, support vector machine; RBF, radial basis function.

Variant	Representation	Classifier	Binary study	Multiclass study
Random Forest	57 scaled features	150 trees, depth 6 [45]	Fitted on the raw features	Same, with balanced class weights
CNN-LSTM	57 features as a sequence	LSTM (64 units) with linear head [46]	Convolutions of 32 and 64 channels with batch normalization	Two 64-channel convolutions without batch normalization, 46,408 trainable parameters
Informer	Encoder trained end to end	Linear head of the encoder	Single-token transformer (sequence length 1)	Per-feature Informer-style encoder
Informer-SVM	Frozen encoder latents	RBF-kernel SVM, C = 1 [47]	Single-token transformer; SVM fitted on a 20,000-flow subsample; AUC from decision scores	Per-feature encoder; SVM fitted on all 12,628 training flows with sigmoid calibration and balanced weights
Informer-XGBoost (proposed)	Frozen encoder latents	XGBoost	Per-feature Informer-style encoder	Per-feature Informer-style encoder

The binary Informer and Informer-SVM rows use a vanilla Transformer encoder that treats the whole flow as one token, so it cannot attend across features. Only the proposed binary model uses the per-feature encoder, so the binary ablation changes encoder and classifier together. In the multiclass study the Informer, Informer-SVM and Informer-XGBoost variants share one trained per-feature encoder (their training logs are identical), which isolates the effect of the classifier head.

5. Experimental protocol and metrics

Each study uses one stratified 70/30 split (seed 42), scores every variant once on the untouched test split, and reports both threshold-based and ranking metrics. No variant was tuned: every hyperparameter is a fixed value chosen in advance, so the test split never informed a design decision.

Cross-validation. Five-fold stratified cross-validation (shuffled, seed 42) was run on the training split of each study. Random Forest, CNN-LSTM and the Informer-only model are retrained from scratch in every fold. Informer-SVM and Informer-XGBoost instead refit only the classifier on latent vectors from the encoder that was trained once on the whole training split. The second design measures the stability of the classifier stage for a fixed encoder, not of the whole pipeline, and its folds are not directly comparable with those of the first design. The two groups are kept visibly separate in the tables.

Binary metrics. Accuracy, precision, recall, F1 and the area under the receiver operating characteristic curve (ROC-AUC) were taken from the run logs. The Matthews correlation coefficient (MCC) [48], the false-positive rate (FPR) and the false-negative rate (FNR) were recomputed from the logged confusion matrices; the same recomputation reproduced every logged accuracy, precision, recall and F1 to within 5×10^{-6}. Attack is the positive class.

Multiclass metrics. The test split holds 676 or 677 flows per class, so accuracy and macro recall (which equals balanced accuracy) almost coincide. The reported metrics are accuracy, macro and weighted F1, macro precision and recall, the multiclass MCC, the macro one-vs-rest ROC-AUC, the macro precision–recall AUC (PR-AUC), a per-class table and the 8 × 8 confusion matrix. All of these were recomputed from the confusion matrices and matched the logged values to within 5×10^{-6}.

Uncertainty. All scores come from a single seed. Wilson 95% intervals [49] for test accuracy are about ±0.04 percentage points in the binary study (600,000 flows) and about ±0.8 points in the multiclass study (5,412 flows). These intervals assume independent flows; because the split is random, flows of the same attack campaign appear on both sides of it, so the true uncertainty is larger and the figures should be read as optimistic estimates of performance on unseen traffic.

Cost. Wall-clock times come from CPU-only runs (no GPU memory was allocated in any log). Per-flow latency is measured on the classifier stage for the hybrids and on the whole model for the Random Forest, the CNN-LSTM and the Informer-only network, so the hybrid latencies understate the cost of a full encoder-plus-classifier pass. The encoder forward pass is reported separately where a log allows it.

Data ingestion and sampling ran in PySpark, and the neural models in PyTorch [50]. Scaling, the Random Forest, the SVM and the probability calibration come from scikit-learn [51], and the explanations from the SHAP library [15, 16].

6. Binary classification and ablation

The Informer-XGBoost pipeline classifies 97.24% of 600,000 held-out flows correctly (MCC 0.9458) and raises 1,751 false alarms, against 7,256 to 11,371 for the four ablation variants (Table 3, Fig. 2).

6.1. Test-set performance

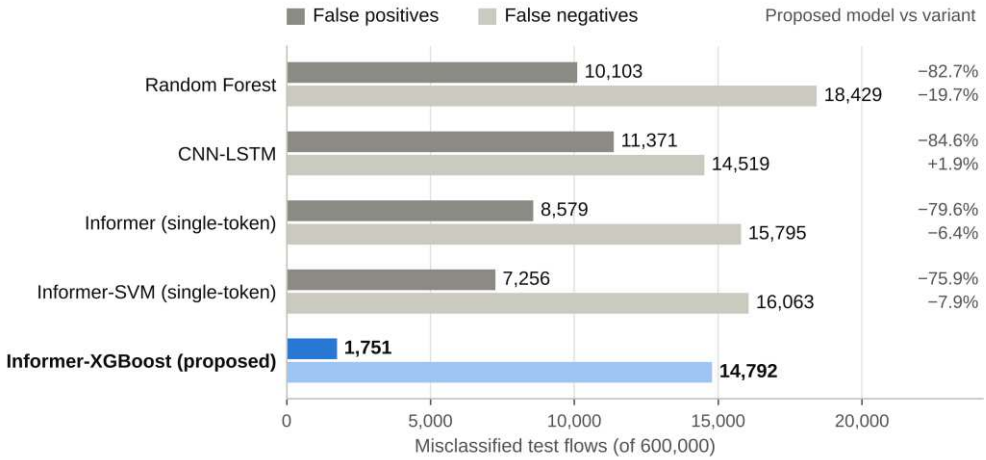
Table 3. Binary test-set performance (600,000 flows, 300,000 per class). Accuracy, precision, recall, F1 and AUC come from the run logs, and MCC was recomputed from the logged confusion matrices. The two single-token rows use a vanilla Transformer encoder (see Table 2).

Variant	Accuracy	Precision	Recall	F1	ROC-AUC	MCC
Random Forest	0.9524	0.9654	0.9386	0.9518	0.9805	0.9052
CNN-LSTM	0.9569	0.9617	0.9516	0.9566	0.9840	0.9138
Informer, single-token	0.9594	0.9707	0.9474	0.9589	0.9851	0.9190
Informer-SVM, single-token	0.9611	0.9751	0.9465	0.9606	0.9820	0.9227
Informer-XGBoost (proposed)	**0.9724**	**0.9939**	**0.9507**	**0.9718**	**0.9911**	**0.9458**

Against the Random Forest the proposed model gains 2.00 percentage points of accuracy and 0.0405 of MCC; against the next-best variant, Informer-SVM, the gains are 1.13 points and 0.0231. For 600,000 flows these gaps are many standard errors wide (an unpaired comparison of accuracies gives z = 34.5 to 57.6), although the independence caveat of Section 5 applies.

Fig. 2 shows where the gain comes from. The proposed model removes 75.9% to 84.6% of the false positives of the other four variants. Its 14,792 missed attacks, however, are 1.9% more than the CNN-LSTM's 14,519 and only 6.4% to 19.7% fewer than those of the remaining three. Recall stays between 93.9% and 95.2% for all five variants while precision rises from the 0.9617 to 0.9751 range to 0.9939, so the improvement is one of precision, with the decision threshold at its default.

Fig. 2. False positives (dark bars) and false negatives (light bars) of the five binary variants on the 600,000-flow test split; blue marks the proposed model. The percentages give the change in the proposed model's count relative to the variant in that row, so a minus sign means fewer errors than the variant.



The missed-attack floor deserves a comment. The proposed model's 14,792 false negatives are close to the 14,419 infiltration flows in the test split. The binary logs contain no per-family recall, so they cannot show that these are the flows that are missed. The multiclass study nevertheless shows that benign and infiltration flows are the pair that the features separate worst (Section 7), which makes infiltration the leading hypothesis for the floor.

6.2. Cross-validation

Table 4 reports five-fold cross-validation on the training split, with the protocol of each variant in the second column.

Table 4. Five-fold cross-validation on the training split (mean ± sample standard deviation of the fold scores).

Variant	Protocol	Accuracy	F1	ROC-AUC
Random Forest	Retrained in each fold	0.9505 ± 0.0001	0.9497 ± 0.0001	0.9803 ± 0.0002
CNN-LSTM	Retrained in each fold	0.8523 ± 0.2081	0.7573 ± 0.4166	0.9162 ± 0.1378
Informer, single-token	Retrained in each fold	0.9603 ± 0.0005	0.9597 ± 0.0005	0.9857 ± 0.0004
Informer-SVM, single-token	Frozen encoder	0.9614 ± 0.0003	0.9608 ± 0.0003	0.9816 ± 0.0005
Informer-XGBoost (proposed)	**Frozen encoder**	**0.9724 ± 0.0005**	**0.9718 ± 0.0005**	**0.9911 ± 0.0003**

The retrained folds of the Random Forest and the Informer are tight, with standard deviations of 0.0001 to 0.0005. The CNN-LSTM is not: its third fold scored 0.4827 against 0.9018 to 0.9607 in the other four, which drags the mean to 0.8523 and indicates an unstable optimization in that fold, even though the final CNN-LSTM trained on the whole training split reached 0.9569 on the test split. The frozen-encoder folds track the test accuracies closely (0.9614 against 0.9611 for Informer-SVM, 0.9724 against 0.9724 for the proposed model), as expected when only the classifier is refitted; they do not show whether a retrained encoder would yield an equally good latent space.

6.3. Computational cost

The proposed pipeline is the cheapest to run at the classifier stage and by far the most expensive to train. Its encoder needed 5,181 s, 35.5 times the 145.9 s of the single-token transformer, because self-attention now runs over 57 tokens instead of one, and encoder training makes up 90.06% of the 5,753 s script time. The XGBoost fit takes 0.39%, XGBoost inference 0.02%, and the remaining 9.54% (549 s) covers ingestion, sampling, scaling, latent extraction, the five XGBoost folds (94 s in total) and exports. The Informer-SVM classifier was fitted on a 20,000-flow subsample (1.4% of the training flows), and its per-flow latency of 0.2983 ms is 201 times that of XGBoost. Because the 0.00148 ms of the proposed model covers the classifier only, an end-to-end binary latency cannot be derived from the log.

Table 5 compares the wall-clock cost of the five binary variants on the CPU.

Table 5. Wall-clock cost of each binary variant on the CPU. The first column covers the whole script, including ingestion, sampling, cross-validation and exports.

Variant	Whole script (s)	Final training (s)	Test inference (s)	Latency (ms per flow)	Latency covers
Random Forest	885.39	304.52	3.11	0.00518	Whole model
CNN-LSTM	1,867.64	335.63	21.88	0.03647	Whole model
Informer, single-token	829.08	145.93	7.70	0.01284	Whole model
Informer-SVM, single-token	950.15	149.00 encoder + 3.06 SVM	178.97	0.29828	SVM stage
Informer-XGBoost (proposed)	5,753.39	5,181.23 encoder + 22.27 XGBoost	0.89	0.00148	XGBoost stage

The rows support two conclusions. A gradient-boosted head on an encoder latent space is more accurate than a linear head (+1.31 points over the single-token Informer) or an RBF-SVM (+1.13 points), and it costs a small fraction of the SVM's inference time. The per-feature encoder also costs far more to train than the single-token one. One question stays open: the single-token variants differ from the proposed model in the encoder and the classifier at once, so the binary study cannot separate the contribution of per-feature tokenization from that of boosting. The multiclass ablation (Section 8) isolates the classifier with one shared encoder. Neither study contains an XGBoost model fitted on the raw features, which is the control that would show what the encoder adds to the classifier alone.

7. Multiclass classification

On the 5,412-flow, eight-class test split the proposed pipeline reaches 90.28% accuracy, a macro-F1 of 0.9012, an MCC of 0.8894 and a macro one-vs-rest ROC-AUC of 0.9899; six classes exceed an F1 of 0.95, while Benign (0.6427) and Infiltration (0.7366) carry most of the remaining error.

7.1. Test-set performance

The pipeline classifies 4,886 of 5,412 test flows correctly (Wilson 95% interval 89.5% to 91.0%) and reaches a macro precision-recall AUC of 0.9380. Bruteforce is nearly perfect (F1 0.9963, ROC-AUC 1.0000). The two weak classes are also the two with the lowest PR-AUC, 0.7731 for Benign and 0.7683 for Infiltration, against at least 0.9880 for every other class (Table 6).

Table 6. Per-class test results of the proposed Informer-XGBoost pipeline (eight classes, 5,412 test flows). Precision, recall and F1 were recomputed from the confusion matrix; the AUC columns are one-vs-rest values from the run metadata.

Class	Test flows	Precision	Recall	F1	ROC-AUC	PR-AUC
Benign	676	0.7185	0.5814	0.6427	0.9567	0.7731
DDoS	676	0.9430	0.9793	0.9608	0.9990	0.9923
DoS	677	0.9553	0.9468	0.9510	0.9975	0.9880
Botnet	677	0.9896	0.9882	0.9889	0.9994	0.9977
Bruteforce	677	0.9970	0.9956	0.9963	1.0000	0.9999
Infiltration	676	0.6867	0.7944	0.7366	0.9697	0.7683
Webattack	677	0.9548	0.9675	0.9611	0.9991	0.9924
Portscan	676	0.9747	0.9689	0.9718	0.9981	0.9920
Macro average	5,412	0.9025	0.9028	0.9012	0.9899	0.9380

7.2. Error analysis

The errors concentrate in one pair of classes (Fig. 3). Benign and Infiltration flows account for 355 of the 526 test errors (67.5%): 228 benign flows are labeled Infiltration and 127 infiltration flows are labeled Benign, which leaves Benign recall at 58.1% and Infiltration recall at 79.4%. The six remaining classes each lose 5.3% of their flows or fewer. Away from the Benign row and column, the largest confusion is DoS labeled as DDoS (17 flows), and no other attack-to-attack cell exceeds 9 flows.

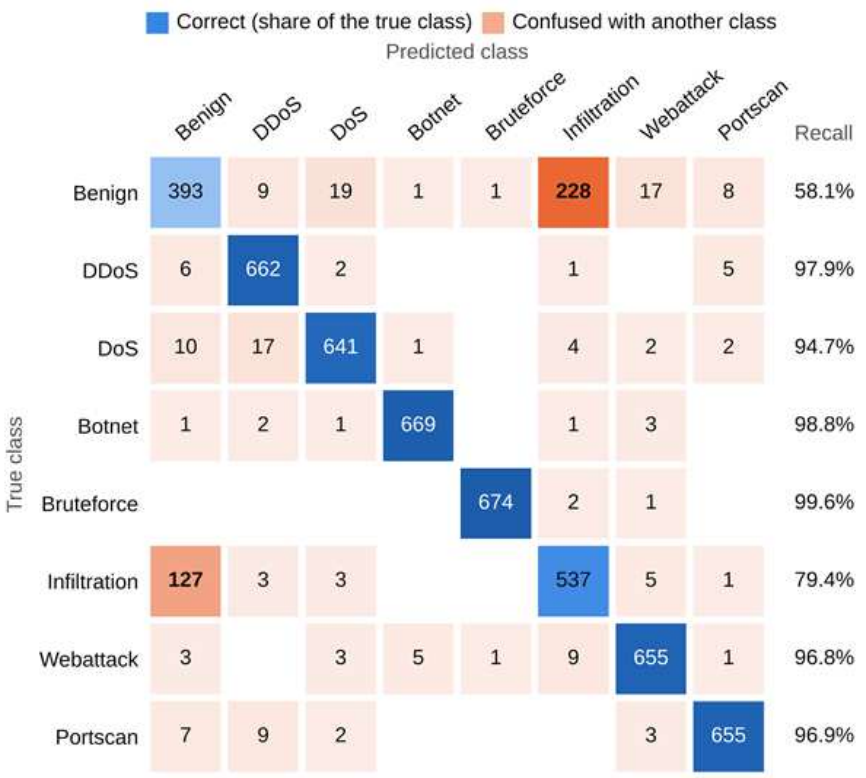


Fig. 3. Confusion matrix of the proposed pipeline on the eight-class test split (rows are true classes, columns predicted classes; empty cells are zero). Blue marks correct classifications and orange tints errors, both shaded by the share of the true class; recall is given at right. The two bold cells are the Benign–Infiltration confusions, which make up 355 of the 526 errors.

Collapsing the eight predicted labels into benign versus attack gives a detection rate of 96.75% (4,582 of 4,736 attack flows) but flags 283 of 676 benign flows (41.9%) as attacks. As a binary alarm the multiclass model would therefore be too noisy, and the number is not comparable with the binary study's false-positive rate of 0.58%, which rests on 1,000,000 benign training flows and a different attack mix. Here the benign class is represented by 2,255 of the 7,186,189 benign flows in the collection (0.03%), a thin sample of a very diverse traffic class, and the under-sampling floor is set by the rarest attack class. Section 10 returns to this limitation.

7.3. Cross-validation

Five-fold cross-validation, which refits XGBoost on frozen encoder latents, gives an accuracy of 0.8935 ± 0.0071, a macro-F1 of 0.8921 ± 0.0073, an MCC of 0.8786 ± 0.0081 and a macro-ROC-AUC of 0.9885 ± 0.0009; fold accuracies run from 0.8832 to 0.9002. The test accuracy of 0.9028 lies 0.9 points above the cross-validation mean and above every fold. The final classifier is fitted on all 12,628 training flows rather than 80% of them, and the fold spread (standard deviation 0.7 points) shows how far one random split can move a score, so a single test split cannot say which of the two explains the gap.

7.4. Computational cost

Encoder training took 43.0 s for three epochs, the XGBoost fit 4.8 s and XGBoost inference on the 5,412 test flows 0.058 s, or 0.0107 ms per flow for the classifier stage. The Informer-SVM run logs the encoder's feature-extraction time for the same flows, 0.9315 s or 0.172 ms per flow. Taking that as a proxy and adding the XGBoost stage gives an end-to-end estimate of about 0.18 ms per flow, 23 times the whole-model latency of the Random Forest (0.0079 ms). This estimate combines timings from two separate runs on the same CPU, so it is indicative only. It does show that the encoder, not the booster, sets the deployment cost. The SHAP suite added 1,647 s to a script that ran for 2,035 s in total.

8. Multiclass ablation study

Holding the encoder fixed, an XGBoost head lifts multiclass accuracy to 90.28% from 50.41% for the encoder's own linear head and from 77.61% for an RBF-SVM, and it also beats a Random Forest on the raw features by 4.45 points.

The Informer, Informer-SVM and Informer-XGBoost variants share a single trained encoder: their epoch logs agree to six decimals (training losses 1.867624, 1.501954 and 1.265129). They differ only in the head that reads the 64-dimensional latent vector, which makes this controlled head ablation. The Random Forest reads the 57 scaled features directly, and the CNN-LSTM is a separately trained end-to-end network, so these two rows compare the hybrid with other model families and do not isolate a component.

8.2. Overall results

On the shared encoder the head matters more than anything else in the study. Accuracy rises by 27.2 points from the linear head to the SVM and by a further 12.7 points to XGBoost, and the macro-ROC-AUC rises from 0.8738 to 0.9394 and then to 0.9899.

The encoder is far from converged (Section 4.1): its running training accuracy was 57.5% after 39 optimizer steps, which is what the 50.4% of its own linear head reflects. That the SVM and XGBoost read 77.6% and 90.3% from the same latent vectors shows that the latent space holds more class structure than a linear head extracts at this training budget. The ablation therefore measures the value of a nonlinear head on a partly trained encoder. A longer training run would probably narrow the gap between the heads, but these logs cannot say by how much.

Table 7 gives the test-set performance of the five multiclass variants.

Table 7. Multiclass test-set performance (5,412 flows). All values were recomputed from the logged confusion matrices, except the two AUC columns, which come from the run metadata.

Variant	Accuracy	Macro-F1	MCC	Macro ROC-AUC	Macro PR-AUC
Random Forest, raw features	0.8583	0.8514	0.8402	0.9829	0.9029
CNN-LSTM	0.4782	0.4025	0.4217	0.8262	0.6009
Informer, linear head	0.5041	0.4751	0.4452	0.8738	0.5887
Informer-SVM	0.7761	0.7698	0.7452	0.9394	0.7221
Informer-XGBoost (proposed)	**0.9028**	**0.9012**	**0.8894**	**0.9899**	**0.9380**

The Random Forest on raw features reaches 85.83%, so the proposed pipeline's margin over it is 4.45 points of accuracy, 0.0498 of macro-F1 and 0.0492 of MCC (an unpaired z of 7.2). The forest is shallow (150 trees of depth 6, untuned) and no XGBoost model was fitted on the raw features, so the margin cannot be credited to the encoder; it is the margin of the whole pipeline over one fixed baseline.

The CNN-LSTM reaches 47.82%. After three epochs its training loss was 1.841 against a chance level of 2.079 and its running training accuracy 41.8%. Like the Informer-only network, it documents what a three-epoch budget buys on 12,628 flows and does not measure the capacity of the architecture.

8.3. Per-class view

The XGBoost head scores higher than the SVM head in every class and higher than the Random Forest in seven of the eight, by 0.2 to 11.3 points of F1; Bruteforce is the exception, where the forest is 0.2 points better (0.9985 against 0.9963). The largest gains over the forest fall on Benign (+11.3), Portscan (+8.6), DDoS (+8.1) and DoS (+7.2). In all five variants, Benign and Infiltration are the two lowest-scoring classes (outlined in Fig. 4). A difficulty shared by five different model families points toward the data rather than one classifier: the benign class is thinly sampled, and infiltration flows may resemble ordinary traffic. Section 10 returns to this.

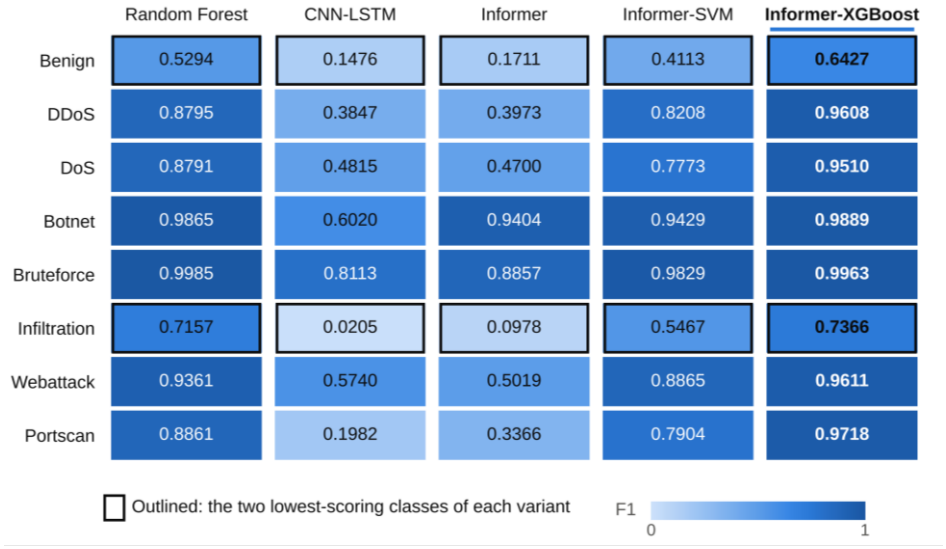


Fig. 4. Per-class F1 of the five multiclass variants on the 5,412-flow test split. Darker blue means a higher F1; the two lowest-scoring classes of each variant are outlined, and the proposed model is in the right-hand column.

8.4. Cross-validation

The two protocols tell different stories. Retraining the encoder in every fold gives an accuracy of 0.3130 ± 0.1147 with folds from 0.1810 to 0.4891, so three-epoch training of the per-feature encoder is unstable, and its test accuracy of 0.5041 exceeds every fold. The folds of the two hybrids reuse one trained encoder and so exclude this source of variance; their small standard deviations describe the head, not the pipeline. The Random Forest and the CNN-LSTM are retrained per fold, and their cross-validation accuracies (0.8663 and 0.4892) lie close to their test accuracies (0.8583 and 0.4782).

Table 8 lists five-fold cross-validation on the 12,628-flow training split, with the protocol of each variant in the second column.

Table 8. Five-fold cross-validation on the 12,628-flow training split (mean ± sample standard deviation). The folds of the Informer-only model are taken from the training-script output because its fold file was empty in the stored results.

Variant	Protocol	Accuracy	Macro-F1	Macro ROC-AUC
Random Forest	Retrained in each fold	0.8663 ± 0.0076	0.8609 ± 0.0084	0.9825 ± 0.0006
CNN-LSTM	Retrained in each fold	0.4892 ± 0.0142	0.4433 ± 0.0174	0.7414 ± 0.0085
Informer, linear head	Retrained in each fold	0.3130 ± 0.1147	0.2489 ± 0.1239	0.8144 ± 0.0162
Informer-SVM	Frozen encoder	0.7644 ± 0.0078	0.7568 ± 0.0084	0.9365 ± 0.0037
Informer-XGBoost (proposed)	**Frozen encoder**	**0.8935 ± 0.0071**	**0.8921 ± 0.0073**	**0.9885 ± 0.0009**

8.5. Computational cost

Table 9 gives the wall-clock cost of each multiclass variant on the CPU.

Table 9. Wall-clock cost of each multiclass variant on the CPU. The whole-script column includes data loading and, for the proposed model, the SHAP suite.

Variant	Whole script (s)	Training (s)	Test inference (s)	Latency (ms per flow)	Latency covers
Random Forest	314.19	0.48	0.043	0.00794	Whole model
CNN-LSTM	345.50	4.31	0.249	0.04596	Whole model
Informer, linear head	570.92	50.72	0.835	0.15430	Whole model
Informer-SVM	455.98	48.73 encoder + 15.25 SVM	0.932 encoder + 5.784 SVM	1.06868 (end to end 1.24079)	SVM stage
Informer-XGBoost (proposed)	**2,034.87 (of which SHAP 1,647.18)**	**42.95 encoder + 4.84 XGBoost**	**0.058**	**0.01071**	**XGBoost stage**

The XGBoost head adds 4.8 s of fitting and 0.058 s of inference for 5,412 flows, whereas the SVM head adds 15.2 s and 5.8 s, a classifier-stage latency about 100 times higher (1.069 ms against 0.0107 ms per flow). Including the

encoder, the SVM pipeline needs 1.241 ms per flow as measured and the XGBoost pipeline about 0.18 ms by the estimate of Section 7.4, so the end-to-end advantage is nearer sevenfold. Data ingestion and sampling alone took 309 to 329 s in each run, according to the resource checkpoints, so the whole-script times of the Random Forest and the CNN-LSTM barely exceed that floor.

9. SHAP explanations of the multiclass model

The multiclass model is explained at two levels: TreeSHAP on the 64 latent dimensions that XGBoost consumes, and permutation SHAP on the whole pipeline, which attributes each prediction to the 57 named flow features. The named-feature level carries the interpretation, and the latent level serves as a check on the classifier head and on the explanation metrics. Every result in this section comes from one run of the proposed pipeline; quantities that the run did not report itself were computed afterward from its stored SHAP arrays.

9.1. Explanation setup

Each latent dimension is a nonlinear function of all 57 inputs, because the encoder mixes tokens through attention, a distilling convolution and average pooling. A TreeSHAP value for a latent dimension therefore cannot be relabeled as a statement about a flow feature. To name the features that drive a prediction, the deployed pipeline (scaled flow features, Informer encoder, XGBoost) is treated as one function of the 57 scaled inputs and explained with the permutation explainer of the SHAP library [15], which estimates Shapley values by sampling permutations of the features [17]. The wrapper reproduces the stored class probabilities of all 5,412 test flows exactly.

Table 10 lists the configuration of the two analyses. Both explain each flow with respect to its own predicted class, in the raw log-odds margin of XGBoost, where Shapley additivity holds exactly and is not distorted by the SoftMax. They differ in how absent features are handled. The TreeExplainer was called without a background set and therefore uses the tree-path-dependent estimate, whereas the permutation explainer replaces absent features with values drawn from a background sample, which is the interventional definition. The two answer slightly different questions, and the difference matters when features are correlated, as they are here [36-38].

Table 10. Configuration of the two explanation analyses.

Item	Latent-space TreeSHAP	Pipeline-level permutation SHAP
Function explained	XGBoost head on the frozen 64-d latent	Scaled flow features, encoder and XGBoost as one function
Attributed to	64 latent dimensions (unnamed)	57 named flow features
Explainer	TreeExplainer without background set (tree-path-dependent, exact)	Permutation explainer with an independent masker (interventional)
Flows explained	All 5,412 test flows	324 test flows
Background	None	64 training flows, 8 per class
Sampling error	None	5 forward-and-reverse permutation pairs; budget of 575 masked evaluations per flow
Output and class	Log-odds margin of the predicted class	Log-odds margin of the predicted class
Wall-clock time	3.47 s	io1,602.66 s

The 324 explained flows are 40 test flows drawn at random from each true class, plus the highest-confidence true positive, true negative, false positive and false negative of the attack-versus-benign framing; these four raise the Benign count to 42 and the Botnet and Infiltration counts to 41. The sample is therefore balanced by true class and is not a random draw of the test set. The model classifies 295 of the 324 flows correctly (91.05%, against 90.28% on the full test set), and the 29 errors are kept because explanations of wrong predictions are informative in their own right. Attributions are grouped by predicted class, which leaves between 31 and 47 flows per class.

9.2. Global importance of the named features

Five flow features carry 36.6% of the total attribution, and they are the same five in every bootstrap resample of the explained flows. Fig. 5 ranks the 15 leading features by mean absolute SHAP value. Fwd Packet Length Max (0.701 log-odds) and Init Fwd Win Bytes (0.688) lead, and their bootstrap intervals overlap (0.634 to 0.769 and 0.619 to 0.764). Across 1,000 bootstrap resamples of the 324 flows the first ranks first in 60.6% and the second in 39.4%, so the sample does not settle their order. It does settle the set: the three features below them (ranks 3 to 5 in Fig. 5) complete the top five in every resample, and the global rankings of two random halves of the flows correlate at $\rho = 0.985$ on average (1,000 splits).

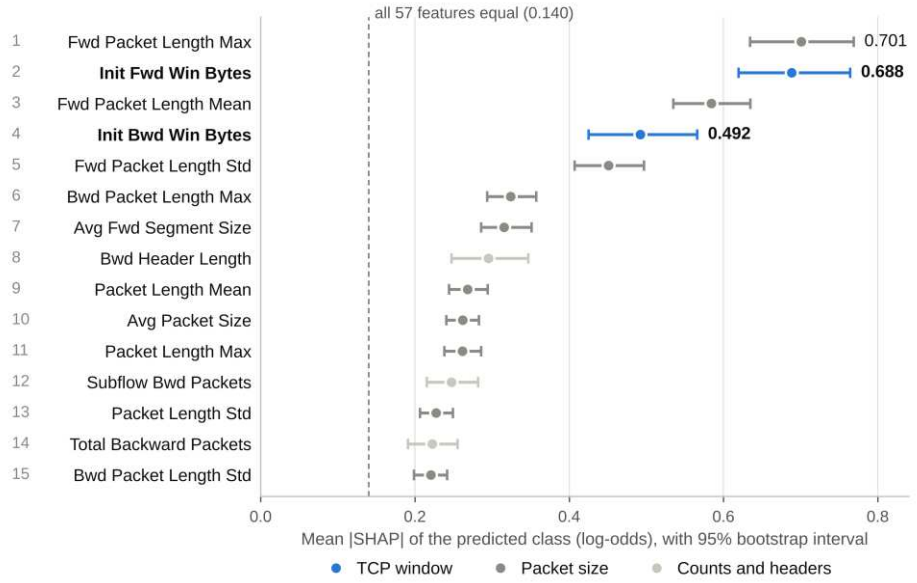


Fig. 5. Mean absolute SHAP value (log-odds) of the 15 leading flow features with 95% bootstrap intervals over the 324 explained flows. Blue marks the two TCP-window fields, medium gray the packet-size features and light gray the packet-count and header features; the dashed line is the level at which all 57 features would contribute equally (0.140).

Attribution is concentrated without being sparse. The leading 10 features carry 55.0% of the total and the leading 20 carry 79.1%. Within a single flow the top feature accounts for 18.7% of the absolute attribution on average, and the top 10 cover 71.2%.

Table 11. The ten leading groups of correlated features. Group attribution is the mean absolute value of the per-flow sum of SHAP values over the members; shares are taken over the 29 groups.

Rank	Leading feature	Members	Group attribution	Share (%)	Other members
1	Bwd Packet Length Max	8	1.234	16.06	Packet Length Std; Bwd Packet Length Std; Bwd Packet Length Mean; Avg Bwd Segment Size; Packet Length Variance; Subflow Bwd Bytes; Bwd Packets Length Total
2	Fwd Packet Length Mean	2	0.880	11.45	Avg Fwd Segment Size
3	Packet Length Mean	3	0.779	10.13	Avg Packet Size; Packet Length Max
4	Bwd Header Length	3	0.741	9.65	Subflow Bwd Packets; Total Backward Packets
5	Fwd Packet Length Max	3	0.723	9.40	Fwd Packets Length Total; Subflow Fwd Bytes
6	Init Fwd Win Bytes	1	0.688	8.96	None
7	Init Bwd Win Bytes	1	0.492	6.41	None
8	Fwd Packet Length Std	1	0.451	5.87	None
9	Flow Duration	5	0.362	4.70	Flow IAT Max; Flow IAT Mean; Fwd Packets/s; Flow Packets/s
10	Fwd IAT Total	3	0.313	4.07	Fwd IAT Max; Fwd IAT Mean

SHAP splits credit among correlated features, so ranks inside a block of near-duplicates carry little meaning. Clustering the 57 features on their absolute Spearman correlation in the training split (average linkage, $|\rho| \geq 0.90$) gives 29 groups, of which Table 11 lists the ten leading. The largest group is a block of eight packet-length statistics led by Bwd Packet Length Max, and it alone holds 16.1% of the group attribution. The two window fields are not correlated with any other feature at that level, so each forms its own group and keeps its rank (sixth and seventh). The window-field result therefore does not depend on how credit is divided among redundant features.

Grouping by feature family gives a coarser view. The 17 packet-size features carry 52.3% of the attribution, the 23 timing features carry 19.4%, and the two window fields carry 14.8%. Per feature, the window fields are the densest source of evidence, with an average of 7.4% of the total each against 3.1% for a packet-size feature and 0.8% for a timing feature.

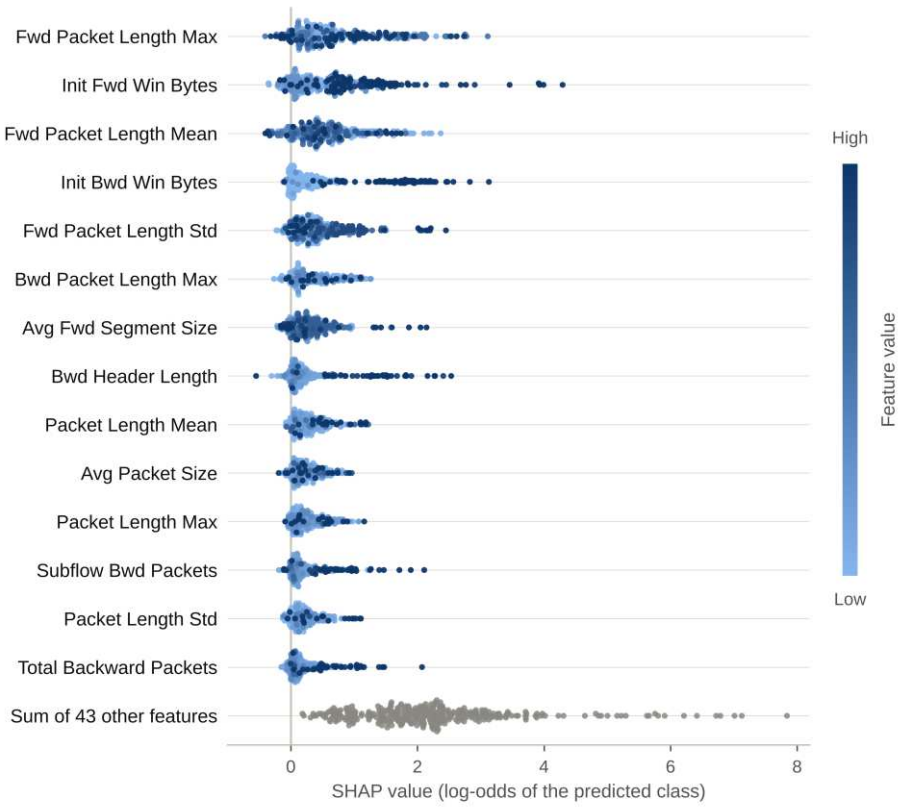


Fig. 6. SHAP summary plot of the 14 leading flow features and the sum of the remaining 43, for the 324 explained flows (permutation SHAP on the full pipeline; log-odds of the predicted class). Each point is one flow; its color gives the feature value, from low (light blue) to high (dark blue), and gray points show the summed contribution of the 43 features that are not listed.

The beeswarm in Fig. 6 adds direction. High values of Init Fwd Win Bytes and Init Bwd Win Bytes lie almost entirely on the positive side, with attributions above +3 log-odds for some flows, and each of the two fields has a positive attribution in 287 of the 324 flows (88.6%). For the packet-length features the sign depends on the class, as Section 9.3 shows, because each flow is explained for its own predicted class: the horizontal axis pools all eight classes, and a positive value means support for the class that was predicted, not for any fixed class.

9.3. Class-specific explanations

For four of the eight predicted classes the leading feature is a forward packet-length statistic (Benign, DDoS, DoS and Botnet), and for the other four it is a TCP-window field (Bruteforce, Infiltration, Webattack and Portscan). Table 12 lists the leading features of each class, and Fig. 7 shows how the 15 globally leading features are distributed over the classes. The split also appears in the share of each class's attribution that the two window fields carry (Fig. 8): 9.1% to 10.1% in the first group and 14.7% to 28.2% in the second.

Benign flows are explained by the spread of forward packet sizes. Fwd Packet Length Std leads (0.583), and high values of it and of Fwd Packet Length Mean raise the Benign margin ($\rho = +0.46$ and $+0.38$); the median standard deviation is 48.5 bytes among predicted-Benign flows and 10.4 bytes among the rest. Benign and Infiltration have the smallest mean margins (2.65 and 1.98 log-odds) and total attributions (4.36 and 3.97); the other six classes range from 3.77 to 7.31 log-odds and from 7.61 to 11.87.

Table 12. Leading features per predicted class among the 324 explained flows. Each cell gives the feature, its mean absolute SHAP value (log-odds) and the direction of its effect, read from ρ, the Spearman correlation between the feature value and its SHAP value within the class: high = high values raise the class score, low = low values raise it, mixed = $|\rho| < 0.20$, constant = the feature takes one value in the class.

Predicted class (flows)	Rank 1	Rank 2	Rank 3	Window share (%)	Rank 1 kept (%)
Benign (31)	Fwd Packet Length Std, 0.583 (high, +0.46)	Fwd Packet Length Mean, 0.316 (high, +0.38)	Fwd Packet Length Max, 0.287 (mixed, +0.07)	10.1	98
DDoS (46)	Fwd Packet Length Mean, 1.006 (mixed, +0.14)	Fwd Packet Length Max, 0.737 (mixed, −0.06)	Init Fwd Win Bytes, 0.581 (mixed, +0.03)	9.1	96
DoS (40)	Fwd Packet Length Mean, 0.566 (low, −0.72)	Init Fwd Win Bytes, 0.459 (mixed, +0.03)	Packet Length Mean, 0.365 (high, +0.51)	9.5	95
Botnet (40)	Fwd Packet Length Max, 1.698 (high, +0.27)	Fwd Packet Length Std, 0.869 (low, −0.27)	Init Fwd Win Bytes, 0.837 (constant)	9.9	100
Bruteforce (40)	Init Fwd Win Bytes, 1.421 (mixed, +0.13)	Bwd Header Length, 1.355 (mixed, +0.06)	Fwd Packet Length Max, 1.198 (high, +0.29)	15.4	67
Infiltration (47)	Init Bwd Win Bytes, 0.346 (high, +0.27)	Fwd Packet Length Mean, 0.300 (high, +0.38)	Fwd Packet Length Std, 0.273 (low, −0.50)	14.7	66
Webattack (41)	Init Bwd Win Bytes, 1.535 (high, +0.78)	Init Fwd Win Bytes, 0.802 (high, +0.49)	Fwd Packet Length Max, 0.503 (high, +0.30)	28.2	100
Portscan (39)	Init Fwd Win Bytes, 0.918 (high, +0.50)	Init Bwd Win Bytes, 0.746 (high, +0.88)	Fwd Packet Length Mean, 0.675 (high, +0.23)	20.7	81

DDoS and DoS flows carry little forward payload: the median forward packet-length mean is 6.7 bytes in predicted-DDoS flows and 6.0 bytes in predicted-DoS flows, against 44.8 and 42.3 bytes in the remaining flows. In DoS low values of this feature raise the class score ($\rho = -0.72$), and so do low values of Init Bwd Win Bytes ($\rho = -0.78$). In DDoS the effect of the leading feature is not monotone ($\rho = +0.14$), so a single direction cannot summarize it. Init Fwd Win Bytes ranks second in DoS (0.459; median 225 bytes against 8,192 in the other classes) and third in DDoS (0.581).

Botnet receives the largest single-feature attribution of any class, 1.698 log-odds for Fwd Packet Length Max (15.2% of the class attribution), and the explained sample shows why. Thirty-nine of the 40 predicted-Botnet flows have identical values in the leading packet-length statistics (forward maximum 326 bytes, forward standard deviation 188.2 bytes, backward maximum 112 bytes), and all 40 have a forward window of 8,192 bytes. The direction labels of this class in Table 12 therefore rest on one deviating flow and say nothing about a graded relation. The uniformity makes the class easy to separate and its explanation stable (the leading feature stays first in all 1,000 bootstrap resamples), but it also means that the explanation describes one traffic pattern rather than a range of botnet behavior.

Bruteforce is explained by a window field together with the volume of backward traffic. Init Fwd Win Bytes (1.421) and Bwd Header Length (1.355) are almost tied, which is why the bootstrap keeps the first at rank 1 in only 67% of resamples. Among the 40 flows the forward window is 26,883 bytes in 37, the forward maximum is 640 bytes in 38, and the backward header length (median 712 bytes against 40 in the other classes) and the backward packet count (median 22 against 2) lie far above those of the other classes. The counts-and-headers family therefore supplies 25.4% of the class attribution, against 4.3% to 8.2% in the other seven classes.

Infiltration is the least decisively explained class. Its mean margin (1.98) and total attribution (3.97) are the smallest of the eight, its leading feature Init Bwd Win Bytes holds only 8.7% of the class attribution, and the bootstrap keeps that feature first in 66% of resamples. The 47 predicted-Infiltration flows are 33 true Infiltration flows, 13 true Benign flows and one Botnet flow, so part of what this explanation describes is the Benign–Infiltration confusion that Section 7.2 identifies as the largest source of error.

Webattack and Portscan are the classes in which the window fields weigh most (28.2% and 20.7% of the class attribution), and large windows push toward both (Init Bwd Win Bytes and Init Fwd Win Bytes: $\rho = +0.78$ and +0.49 in Webattack, +0.88 and +0.50 in Portscan). For Webattack, Init Bwd Win Bytes alone holds 18.5% of the attribution (1.535 log-odds): the backward window is 28,960 bytes in 30 of the 41 predicted-Webattack flows, the forward window is 29,200 bytes in 33, and 35 flows carry no forward payload at all. Predicted-Portscan flows are similarly small (36 of 39 have a forward maximum of 6 bytes or less), and their forward window is 29,200 bytes in 21 flows and 1,024 bytes in 15.

Fig. 7 shows the same contrasts feature by feature. The Init Bwd Win Bytes row is pale in every class except Webattack (1.54) and Portscan (0.75), Bwd Header Length (1.36) and the two backward packet counts (0.85 and 0.79) matter almost only for Bruteforce, and the Benign and Infiltration columns are pale throughout (largest cells 0.58 and 0.35).

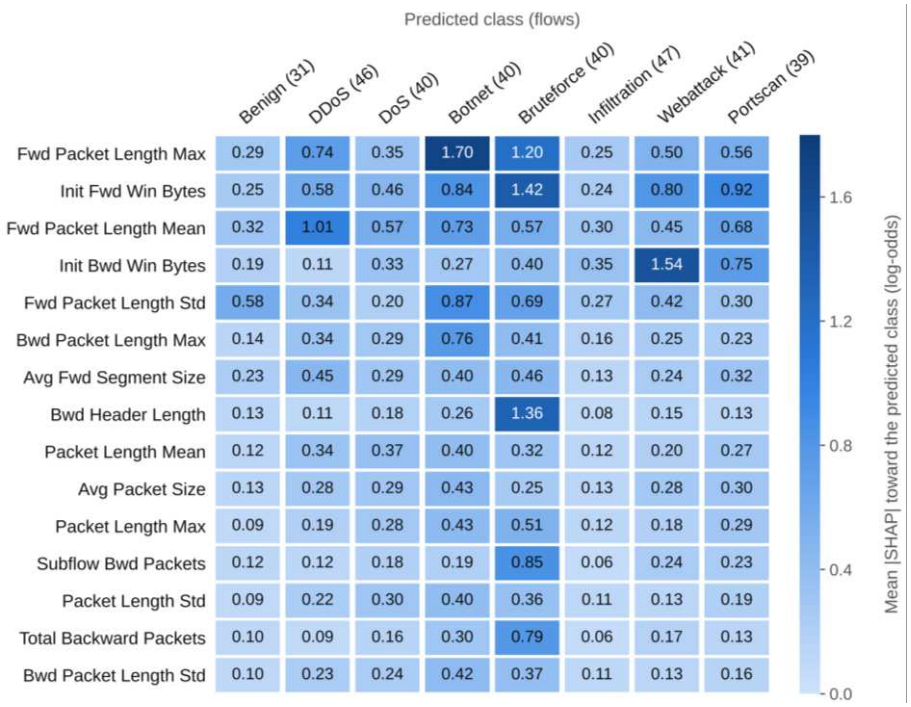


Fig. 7. Mean absolute SHAP value (log-odds toward the predicted class) of the 15 globally leading flow features within each predicted class, over the 324 explained flows (flows per class in brackets). Darker cells are larger.

Fig. 8 collapses the 57 features into families. Packet-size features supply the largest share in every class, from 39% in Webattack to 68% in Botnet, whereas the TCP-window share varies about threefold (9% in DDoS to 28% in Webattack) and the timing share from 9% in Bruteforce to 32% in DoS. The per-feature averages behind these family shares are given in Section 9.2.

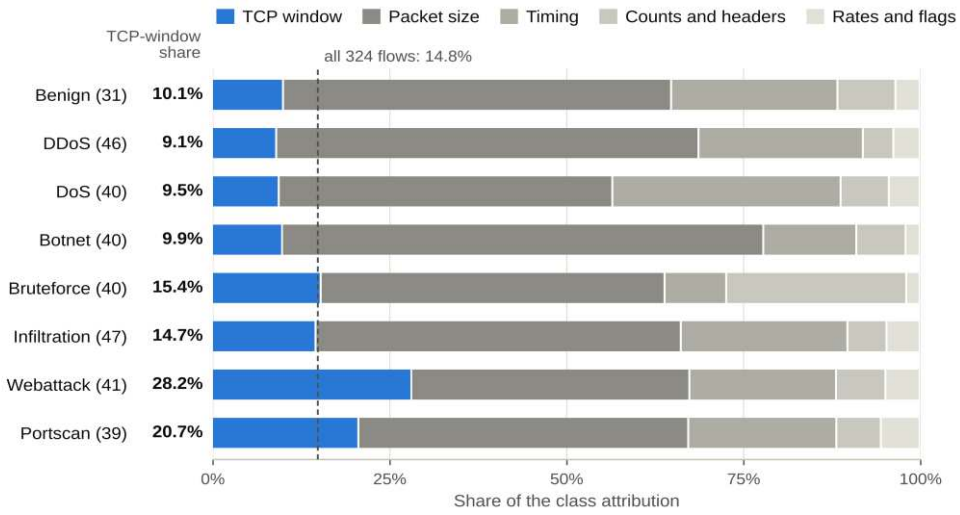


Fig. 8. Share of each predicted class's mean absolute SHAP attribution by feature family (324 explained flows; flows per class in brackets). For readability the packet-count and header families are merged into counts and headers and the rate and flag families into rates and flags. The column at left gives the TCP-window share of each class; the dashed line marks its value over all 324 flows (14.8%).

9.4. Local explanations

Fig. 9 shows the permutation-SHAP waterfall of one flow for each of the four outcomes of the attack-versus-benign reading of the eight-class output, in which any non-Benign class counts as an attack: an attack predicted as an attack (TP), a benign flow predicted as an attack class (FP), an attack predicted as Benign (FN) and a benign flow predicted as Benign (TN). For each outcome, the analysis script chose the test flow with the highest probability of its own

predicted class, so the four flows are the most confident examples of their kind and not typical ones; the FP and FN flows in particular are confident mistakes rather than borderline calls. Table 13 summarizes the four explanations.

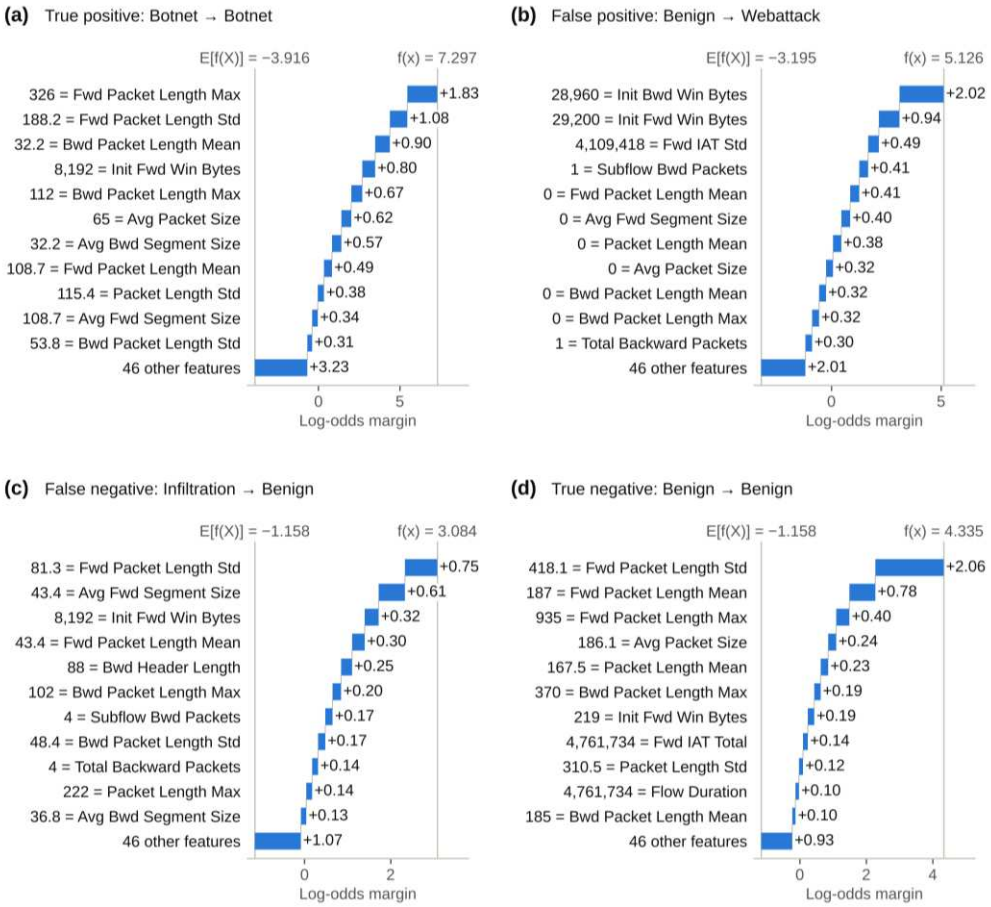


Fig. 9. Permutation-SHAP waterfall plots (log-odds margin of the predicted class) for the four example flows: true positive (a), false positive (b), false negative (c) and true negative (d). Each panel lists the 11 largest contributions with the flow's value of the feature, and the summed contribution of the other 46 features; the bars accumulate from the base value E[f(X)], the mean margin of the predicted class over the 64-flow background sample, to the model output f(x) for the flow.

Table 13. Explanation of the four example flows (permutation SHAP, log-odds of the predicted class). Lift = f(x) minus the base value, the mean margin of the predicted class over the 64 background flows.

Outcome (test row)	True → predicted (probability)	Base value → f(x) (lift)	Largest contribution	Window fields	46 remaining features
TP (520)	Botnet → Botnet (0.9999)	−3.916 → 7.297 (11.213)	Fwd Packet Length Max = 326: +1.831 (16.3%)	+1.031 (9.2%)	+3.231 (28.8%)
FP (4005)	Benign → Webattack (0.9988)	−3.195 → 5.126 (8.321)	Init Bwd Win Bytes = 28,960: +2.019 (24.3%)	+2.959 (35.6%)	+2.014 (24.2%)
FN (5161)	Infiltration → Benign (0.9579)	−1.158 → 3.084 (4.242)	Fwd Packet Length Std = 81.285: +0.753 (17.7%)	+0.351 (8.3%)	+1.074 (25.3%)
TN (2113)	Benign → Benign (0.9970)	−1.158 → 4.335 (5.493)	Fwd Packet Length Std = 418.145: +2.062 (37.5%)	+0.236 (4.3%)	+0.933 (17.0%)

The true positive is a Botnet flow (test row 520) predicted with probability 0.9999 and a margin of 7.297 log-odds, which is 11.213 above the Botnet base value of −3.916. Fwd Packet Length Max supplies 1.831 of the lift (16.3%), the ten features below it adds 0.31 to 1.08 each, and 54 of the 57 contributions are positive; the only negative one is −0.003. The flow carries the signature described in Section 9.3 (a forward maximum of 326 bytes, a forward standard deviation of 188.2, a backward maximum of 112 and a forward window of 8,192), which 39 of the 40 predicted-Botnet flows share.

The false positive is a Benign flow (test row 4005) predicted as Webattack with probability 0.9988. The two window fields, 28,960 bytes backward and 29,200 bytes forward, supply 2.959 log-odds (35.6% of the lift of 8.321), and nine features that describe an empty payload or a single backward packet add between 0.30 and 0.41 each. This is the Webattack profile of Section 9.3: 30 of the 41 predicted-Webattack flows share the backward window, 33 the forward window, and 35 carry no forward payload. A benign flow with that profile is therefore explained exactly

as a web attack would be, and whether the profile is a trait of the attack or of the way the collection was assembled is a question the explanation cannot answer (Section 9.5).

The false negative is an Infiltration flow (test row 5161) predicted as Benign with probability 0.958, and its explanation is the most diffuse of the four. The largest contribution, Fwd Packet Length Std of 81.3 bytes, supplies 17.7% of the lift of 4.242, the 46 features outside the 11 shown supply 25.3%, and 14 features are needed to account for 80% of the absolute attribution. The forward window of 8,192 bytes adds +0.321 toward Benign here and +0.797 toward Botnet in the true positive, so what a window value contributes depends on the class being explained and on the other features, which a global ranking cannot show. Because each flow is explained for its predicted class only, the waterfall shows what supported Benign and not what spoke for Infiltration.

The true negative (test row 2113) is explained by one feature more than by any other: Fwd Packet Length Std of 418.1 bytes supplies 2.062 of the 5.493 lift (37.5%), and Fwd Packet Length Mean (187 bytes) adds 0.777, in line with the Benign profile of Table 12, in which a wide spread of forward packet sizes raises the Benign score. The window fields supply only 4.3% of this lift, the smallest share of the four.

Across the four flows the largest single contribution carries between 16% and 38% of the lift and the window fields between 4.3% and 35.6%. The 11 displayed features hold 71% to 83% of the lift, and no flow needs fewer than 13 features to reach 80% of its absolute attribution (13 to 16 across the four), so the short explanations that a waterfall plot suggests owe something to its 11-feature display.

9.5. The TCP-window fields and the benchmark

The two TCP-window fields rank second and fourth overall and carry 28.2% of the Webattack attribution, so their meaning deserves a closer look. They are the initial-window features that CICFlowMeter extracts for each direction. The window that a host announces depends on its operating system and configuration, which is why the TCP window size is a long-standing attribute of operating-system fingerprinting [51]; it describes the sending machine more than the behavior of the traffic. Table 14 lists the window values that occur in at least five of the 324 flows and have a median SHAP of at least +0.25 log-odds, together with the most frequent value of each field.

Table 14. Window values among the 324 explained flows. Rows are the values that occur in at least five flows and have a median SHAP of at least +0.25 log-odds (rounded to two decimals), plus the most frequent value of each field. True classes give the number of flows of each true class that carry the value; the median SHAP and its interquartile range are those of the window field for the predicted class. The value −1 appears to mark a missing window.

Field	Window value (bytes)	Flows	True classes	Median SHAP (interquartile range)
Forward	8,192	93	Botnet 40, Infiltration 23, Benign 15, DDoS 13, Webattack 2	+0.32 (+0.06 to +0.75)
Forward	29,200	60	Webattack 32, Portscan 21, Bruteforce 3, DoS 2, Infiltration 1, Benign 1	+0.86 (+0.69 to +1.36)
Forward	26,883	40	Bruteforce 37, DoS 3	+1.22 (+1.07 to +1.62)
Forward	−1	35	Benign 15, Infiltration 11, DDoS 7, Portscan 1, Webattack 1	+0.32 (+0.19 to +0.46)
Forward	1,024	17	Portscan 15, Infiltration 2	+0.41 (+0.27 to +0.71)
Forward	225	16	DoS 16	+0.66 (+0.43 to +0.81)
Forward	65,535	6	DDoS 5, Portscan 1	+3.92 (+3.57 to +3.98)
Backward	−1	74	DoS 18, Benign 18, DDoS 18, Infiltration 14, Webattack 3, Portscan 2, Botnet 1	+0.09 (+0.02 to +0.28)
Backward	219	50	Botnet 39, DDoS 5, DoS 4, Webattack 1, Infiltration 1	+0.25 (+0.12 to +0.33)
Backward	28,960	39	Webattack 29, Portscan 9, Benign 1	+1.89 (+1.64 to +2.14)
Backward	230	37	Bruteforce 37	+0.38 (+0.32 to +0.43)
Backward	29,200	13	Portscan 13	+1.78 (+1.56 to +1.80)
Backward	62,852	5	Infiltration 4, Benign 1	+1.36 (+1.34 to +1.41)

The largest attributions belong to rare, class-specific values. The forward window 65,535 occurs in six flows, and in the five DDoS flows among them (all with a backward window of 219) it contributes +3.46 to +4.29 log-odds, the five largest single attributions among the 18,468 values of the sample; the sixth flow, predicted Portscan, receives −0.11. The backward windows 28,960 and 29,200 occur in 39 and 13 flows, 51 of which are Webattack or Portscan, and contribute +1.89 and +1.78 at the median. The values spread over the most classes (forward 8,192 and −1, backward −1, where the most frequent class holds 43%, 43% and 24% of the flows) receive only +0.32, +0.32 and +0.09. In all, 120 of the 324 flows (37.0%) have at least one window field contributing +1 log-odds or more, and 43 of the 100 largest single attributions are window fields.

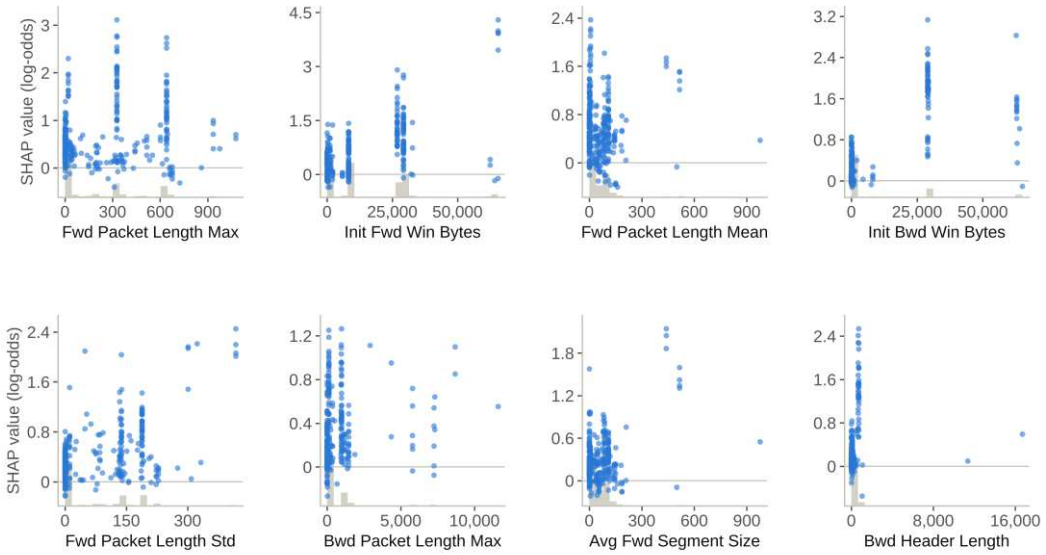


Fig. 10. SHAP dependence of the eight leading raw flow features: SHAP value (log-odds of the predicted class) against the feature value for the 324 explained flows. The gray bars at the foot of each panel show the distribution of the feature values.

Fig. 10 shows the same structure for the eight leading features. The points form vertical stripes at a few recurring raw values instead of a gradual trend, and the height of a stripe shows how differently the model treats flows that carry the same value. Fwd Packet Length Max has tall stripes at 326 and 640 bytes, the Botnet and Bruteforce signatures of Section 9.3, and Init Fwd Win Bytes has four separate stripes (up to 1,000, near 8,192, from 26,000 to 33,000 and above 62,000 bytes).

How much of the classification do the two fields hold on their own? A leave-one-out lookup labels each explained flow with the majority true class of the other flows that carry exactly the same pair of window values (eight flows face a tie, which goes to the lowest class index). It is right for 84.8% of the 283 flows that have such a partner (87.3% of the sample), against 93.3% for the model on the same flows, and for 74.1% of all 324 flows when the 41 flows without a partner count as errors, against 91.0% for the model. Counting every tie as an error lowers the 74.1% to 72.2%; chance is 12.5% and the largest class holds 13.0% of the flows. Keyed on one field alone, the lookup reaches 64.1% for the forward window (306 covered flows) and 61.7% for the backward window (303 covered flows). The lookup is a diagnostic on the explained sample and not a classifier trained and tested on separate data: its keys come from the same 324 flows, and flows from one capture session can share a key. It shows how much class information the two fields carry in this collection, not that the model relies on them for the right reasons.

A plausible reading is that the window fields identify the machines and tools that generated each class rather than the behavior of an attack. If each attack class was launched from a few machines with fixed software, as testbed captures usually are, the window values would recur within a class and differ between classes, and the Botnet and Bruteforce flows of Section 9.3, which share a forward window of 8,192 bytes in all 40 cases and of 26,883 bytes in 37 of 40, fit that picture. The model would then have learned part of the testbed's layout, and the explanations would describe that faithfully instead of revealing a property of the attacks. Documented flaws in related CIC benchmarks [5,6] (both on CIC-IDS2017) and the general risk of spurious correlations in security data [7] make the reading credible, but it remains a hypothesis: the file used here records neither capture sessions nor the hosts behind each class.

A direct test would retrain the whole pipeline, encoder included, without the two fields and compare the loss in accuracy with the loss from removing two other fields, so that the drop is not simply the effect of removing any two features. Splitting by capture session or by time, or evaluating on a second collection with different hosts, would show whether the fields support generalization or only this benchmark split. None of these experiments was run, so the study cannot say whether the window fields are shortcuts. What the explanations establish is the model's heavy use of them; whether that use transfers to other networks are untested.

9.6. Quality of the explanations

Both explanation levels were scored with a set of quantitative checks (Table 15). The checks borrow from the evaluation of explanation methods elsewhere. The deletion and insertion curves follow Petsiuk et al. [40] and the Gini coefficient as a sparsity measure follows Hurley and Rickard [41]; the perturbation tests of stability and robustness are of the kind discussed by Warnecke et al. [18] and Alvarez-Melis and Jaakkola [42]. The latent suite ran on the 5,412 test flows (200 flows for stability and robustness, 100 for interactions) and the pipeline suite on the 324 explained flows; the two columns describe different objects, samples and baselines and should not be compared line by line.

Table 15. Quantitative quality checks of the two explanation levels. The columns differ in object (latent dimensions of the XGBoost head against named flow features of the pipeline), sample and baseline (training latent mean against training mean of the scaled features). Stability perturbs the latent features of 200 flows with Gaussian noise (three repeats; two repeats per level in the robustness sweep) and compares the original and perturbed attributions of the originally predicted class. SD, standard deviation.

Check	Latent TreeSHAP (64 dimensions, 5,412 flows)	Pipeline permutation SHAP (57 features, 324 flows)
Completeness error in margin space (mean; maximum)	7.1×10^{-7}; 4.8×10^{-6}	maximum 2.6×10^{-3}
Deletion AUC (lower = more faithful)	0.194	0.140 (random ranking 0.223)
Insertion AUC (higher = more faithful)	0.691	0.355 (random ranking 0.219)
Probability drops when the top feature is masked (mean ± SD; median)	0.105 ± 0.159; 0.057	not computed
Stability at 0.05 SD noise: Spearman ρ; top-10 Jaccard (mean ± SD)	0.779 ± 0.182; 0.663 ± 0.207	not computed
Robustness from 0.01 to 0.3 SD noise: Spearman ρ	0.902 to 0.495 (mean 0.684)	not computed
Robustness from 0.01 to 0.3 SD noise: top-10 Jaccard	0.828 to 0.418	not computed
Gini coefficient of the absolute attributions (mean ± SD)	0.736 ± 0.064	0.715 ± 0.046
Features needed for 80%, 90% and 95% of the attribution (mean)	14.5, 24.6 and 33.7	14.1, 21.1 and 28.0

Completeness holds at both levels. In margin space the attributions plus the base value reproduce the model output to within 4.8 × 10−6 for TreeSHAP (mean 7.1 × 10−7) and to within 2.6 × 10−3 for the permutation explainer, whose estimate rests on five antithetic permutation pairs per flow. The wrapper that passes the pipeline to SHAP returns the same predicted class as the pipeline for every flow (agreement 100%, maximum probability difference 0.0).

The faithfulness checks mask or reveal features in order of decreasing absolute SHAP value and follow the probability of the predicted class. Deleting the pipeline-level features in SHAP order removes that probability faster than deleting them in random order (deletion AUC 0.140 against 0.223 for a random ranking, 37.0% lower), and inserting them restores it faster (insertion AUC 0.355 against 0.219, 1.62 times higher). The random control is needed because an AUC has no natural scale, and the two random AUCs are close to each other, as expected from a ranking that carries no information. The absolute insertion AUC is modest because the baseline is the training mean of the scaled features, which is not an observed flow, and because the probability recovers fully only when most of the discriminating fields are back. In the latent space the same procedure gives a deletion AUC of 0.194 and an insertion AUC of 0.691 (Fig. 11), and masking the single most important latent dimension of each flow lowers the predicted-class probability by 0.105 on average (median 0.057). The mean probability falls from 0.91 to about 0.5 once the three most important dimensions are masked (read from the deletion curve in Fig. 11).

Sparsity is moderate at both levels. The Gini coefficient of the absolute attributions of a flow is 0.715 ± 0.046 at the pipeline level (0.736 ± 0.064 for the latent dimensions), and 14.1 of the 57 features on average account for 80% of a flow's attribution, 21.1 for 90% and 28.0 for 95% (medians 14, 22 and 28); the latent figures are 14.5, 24.6 and 33.7 of 64. The first five features of a flow carry 51.3% of its attribution on average and the first ten 71.2%, so a typical explanation needs about 14 named features to be 80% complete, more than the 11 that a standard waterfall plot displays.

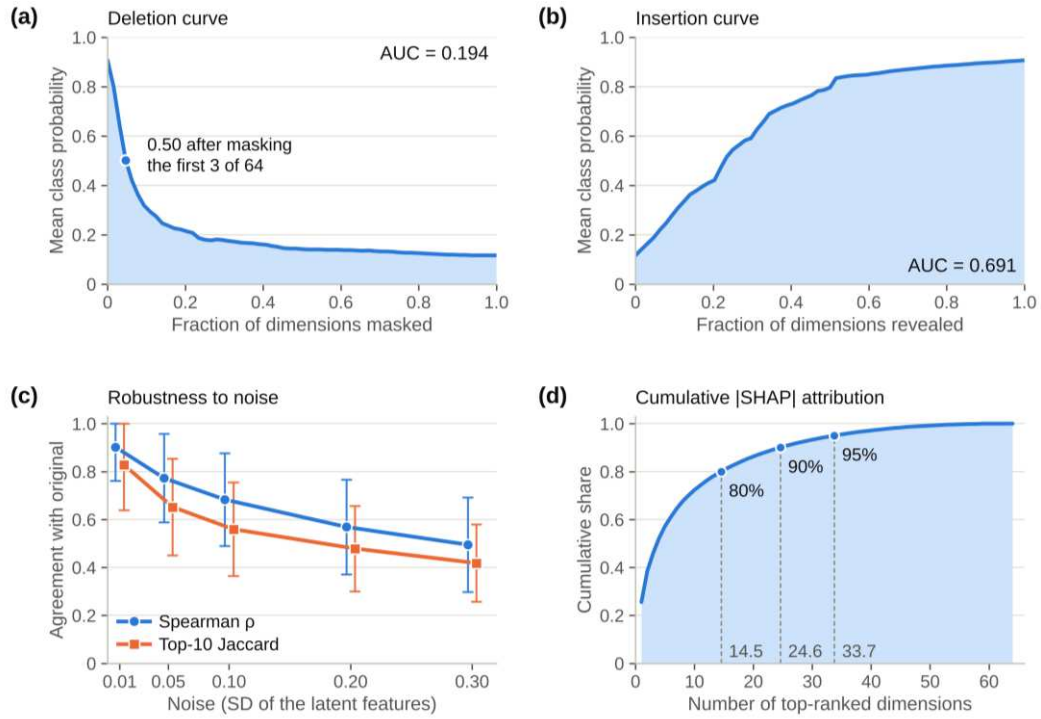


Fig. 11. Diagnostics of latent-space TreeSHAP explanations. (a) Deletion curve showing predicted-class probability as dimensions are masked by absolute SHAP importance. (b) Insertion curve showing probability as dimensions are revealed from the training mean. (c) Attribution stability under noise, measured by Spearman correlation and top-10 Jaccard overlap across 200 flows (±1 SD). (d) Cumulative absolute SHAP attribution, showing dimensions needed to capture 80%, 90%, and 95% of total attribution.

Stability and robustness were measured only in the latent space. Adding Gaussian noise of 0.05 standard deviations to the latent features of 200 flows and recomputing TreeSHAP for the originally predicted class leaves a Spearman correlation of 0.779 ± 0.182 and a top-10 overlap (Jaccard) of 0.663 ± 0.207 with the original attributions (three repeats). Agreement falls steadily with the noise level, from 0.902 and 0.828 at 0.01 standard deviations to 0.495 and 0.418 at 0.3 (Fig. 11; the robustness score of 0.684 is the mean Spearman correlation over the five levels). The latent attributions are therefore only moderately stable: a perturbation of one twentieth of a standard deviation replaces about two of the ten leading dimensions. The test perturbs the encoder output and not the flow features, and a Spearman correlation over 64 mostly small attributions is partly driven by the ordering of near-zero values, so the Jaccard figure is the more demanding one. No stability or robustness figure exists for the pipeline-level explanations, whose features are named but correlated (29 correlated groups at $|\rho| \geq 0.9$), and the Monte-Carlo error of the five-permutation estimate was not measured: the explained flows contain no exact duplicates that could serve as replicates, and the explainer was not rerun with other seeds.

Interaction values for 100 flows over the 20 leading latent dimensions are dominated by the diagonal: main effects carry the attribution and off-diagonal interactions are close to zero (a visual reading of the interaction heatmap, which is not reproduced here), so an additive attribution is a fair summary of the tree ensemble on these dimensions. Individually the latent dimensions are diffuse: the leading one, latent_dim_43, carries about 8% of the total attribution, and the 45 dimensions outside the 19 leading ones carry about 40% together (read from the bar plot of mean absolute attributions). This diffuseness, and the fact that a latent dimension has no name, are the reasons for the pipeline-level explanation of Sections 9.2 to 9.5.

9.7. Cost of explanation

Latent TreeSHAP explains all 5,412 test flows in 3.47 s, or 0.64 ms per flow, and the whole latent suite of Table 15 (interaction values, both faithfulness curves, stability, robustness and the plots) takes 19.96 s. The pipeline-level permutation explanations of the 324 flows take 1,602.66 s, or 4.95 s per flow, about 7,700 times the TreeSHAP cost per flow; each flow needs 575 evaluations of the full pipeline (five permutation pairs over 57 features). The

explanation steps account for 1,647 s of the 2,035 s of the whole run (80.9%), against 388 s for data loading, encoder training, XGBoost training and evaluation. At this rate explaining every test flow at the pipeline level would take about 7.4 hours on the same CPU, and a single explanation cost roughly 27,000 end-to-end predictions at the 0.18 ms per flow estimated in Section 7.4. Named-feature explanations of this kind therefore suit audits and offline analysis; for online use the cheap latent explanation or an approximation of the permutation estimate would be needed, and the approximation would have to be validated against it.

10. Discussion and threats to validity

The evidence supports a narrow claim. On this collection and split, a boosted-tree head on a frozen Informer-style encoder outperforms every tested alternative, and its explanations rest heavily on a pair of TCP window fields that may be testbed fingerprints. The study does not show that the encoder is needed or that the model transfers to another network, and the stability of the pipeline-level explanations is unmeasured.

With one shared encoder, the head alone moves multiclass accuracy by 39.9 points, from 50.41% with the linear head through 77.61% with the SVM to 90.28% with XGBoost. That ordering agrees with the FlowTransformer finding that the classification head had the largest effect on performance [11] and with benchmark evidence that tree ensembles are strong on tabular data [8]. The encoder is only partly trained, though: its running training accuracy was 57.5% after 39 optimizer steps, so the ablation measures how much a nonlinear head can extract from a partly trained latent space, and longer training would probably narrow the gaps by an amount that these logs cannot give.

What the ablation cannot show is that the encoder is needed. No boosted-tree model was fitted on the 57 scaled features, and none on the output of an untrained encoder. If the first matched the pipeline, the encoder would account for about 0.17 ms of the estimated 0.18 ms per flow and would add a layer of opacity, for no gain in accuracy; if the second matched it, the learned weights would be irrelevant and the encoder would act as a random feature map. Both controls are cheap, since the XGBoost fit on the 12,628 training flows took 4.8 s (Section 7.4).

The binary and multiclass numbers answer different questions. The attack class of the binary study inherits the family proportions of the collection, so DDoS and DoS supply 82.3% of its test attacks and the 97.24% accuracy describes volumetric attacks far more than rare ones; its 14,792 missed attacks are close to the 14,419 infiltration flows in the test split, which makes Infiltration the leading hypothesis for the recall ceiling (Section 6.1). The multiclass study removes that dominance and exposes the weak pair: Benign and Infiltration account for 355 of the 526 errors and are the two lowest-scoring classes in all five variants, so the difficulty lies at least partly in the data. Only 2,255 of the 7,186,189 benign flows (0.03%) are represented, and infiltration flows may resemble ordinary traffic. The explanations agree with this reading: Benign and Infiltration have the smallest mean margins (2.65 and 1.98 log-odds) and total attributions, and 13 of the 47 predicted-Infiltration flows in the explained sample are benign.

At class level the explanations tell a coherent story. Packet-size features supply the largest share of the attribution in every class, from 39% in Webattack to 68% in Botnet, while the TCP-window share rises from about 9% in DDoS to 28.2% in Webattack and 20.7% in Portscan. At flow level the waterfalls show what a global ranking hides: the same forward window of 8,192 bytes adds +0.321 toward Benign in the false-negative flow and +0.797 toward Botnet in the true-positive flow, so a feature value has no fixed meaning outside the class that is being explained. They also show a confident mistake that a reader can follow, a benign flow predicted as Webattack with probability 0.9988 because its window fields and its empty payload match the Webattack profile.

The TCP initial-window result carries the most practical weight. A window value describes the sending host more than the behavior of the traffic, so if each class was launched from a few hosts, a model can score well by recognizing the hosts. A lookup on the two fields alone labels 74.1% of the explained flows correctly against 91.0% for the model, and five DDoS flows with a forward window of 65,535 bytes receive the five largest single attributions of the sample (+3.46 to +4.29 log-odds). Whether the model survives the removal of the fields or a change of hosts is unknown. If it does not, the accuracy figures of this paper say less about attack behavior than they appear to, and the explanation will have done its job by pointing to a concrete audit.

10.1. Threats to validity

Internal validity. Several choices could inflate or distort the scores. The scaler was fitted on the whole balanced sample before the split (Section 3), which leaks two moments per feature; the effect is probably small but was not measured. The three-epoch budget means that the Informer-only and CNN-LSTM rows document what a short run buys and not what the architectures can reach (Section 8.2). Every result comes from one seed and one fixed configuration without tuning, so small differences between the weaker variants, and even the 4.45-point margin over the Random Forest (an unpaired z of 7.2 that assumes independent flows), should be confirmed over several seeds. The binary ablation changes encoder and classifier together (Section 4.3), and the folds of the multiclass Informer-only model were transcribed from the training-script output because its fold file was empty (Table 8).

Construct validity. The latency of the hybrids covers the classifier stage, and the 0.18 ms end-to-end figure combines timings from two runs (Section 7.4). The explanations describe the model and not the traffic [38]. The pipeline-level estimate rests on a background of 64 flows and five permutation pairs per flow, applied to 324 flows balanced by true class; its Monte-Carlo error and its stability were not measured (Section 9.6). Attributions toward the predicted class cannot show what spoke for the true class of a misclassified flow, and correlated features share credit (Section 9.2).

External validity. The collection consolidates CIC releases with documented defects (Section 2.1), and its classes are internally heterogeneous: DDoS mixes 15 fine-grained labels, and DoS is 80.5% Hulk, Bruteforce 94.4% SSH and Webattack 67.4% brute force (Section 3). A random split places flows from the same campaign on both sides, so the scores are optimistic estimates for unseen traffic (Section 5), and Moczkodan and Ragab [12] show in a related setting that random splits with repeat-last padding can overstate robustness by up to 0.24 macro-F1. Balancing to 2,255 flows per class keeps 0.03% of the benign flows, so the 41.9% false-alarm rate of the multiclass output collapsed to benign versus attack (Section 7.2) must not be read as a deployment figure.

10.2. Experiments that would settle the open questions

Table 16. Open questions, the evidence available in this study and the experiment that would answer each.

Open question	Evidence in this study	Experiment that would settle it
Does the encoder add anything beyond the booster?	A shallow Random Forest on the raw features reaches 85.83%; no boosted-tree control was run	XGBoost on the 57 scaled features, and XGBoost on the output of an untrained encoder
Is the head effect a product of the short training run?	Encoder loss fell from 1.868 to 1.265 in 39 optimizer steps	Train the encoder to convergence with early stopping and repeat the head ablation over several seeds
Do per-feature tokens or boosting explain the binary gain?	The binary Informer and Informer-SVM use a single-token encoder	Repeat the binary head ablation with the per-feature encoder
Does performance survive a change of hosts or time?	One random 70/30 split of a consolidated collection	Split by capture session or day, and evaluate on a second collection
Are the window fields shortcuts?	They carry 14.8% of the attribution, and a look-up on them reaches 74.1%	Retrain the whole pipeline without them and without two matched control fields, and compare the losses
How noisy and stable are the pipeline-level attributions?	Five permutation pairs per flow; no replicate runs and no perturbation test	Rerun the explainer with other seeds and perturb the 57 inputs as the latent space was perturbed
Can named-feature explanations run at deployment speed?	4.95 s per flow, about 7,700 times the TreeSHAP cost	Validate a cheaper approximation against the permutation estimate

Table 16 pairs each open question with the evidence in this study and the experiment that would settle it. The first four rows decide whether the performance claims survive, and the last three how far the explanations can be trusted and used.

11. Conclusion

A frozen Informer-style encoder with an XGBoost head classified 97.24% of 600,000 binary test flows (MCC 0.9458) and 90.28% of 5,412 eight-class test flows (macro-F1 0.9012) correctly, ahead of every ablation variant and of a Random Forest on the raw features. With one shared encoder the head moved multiclass accuracy by 39.9 points. The encoder was trained for only three epochs and no boosted-tree control on the raw features was run, so the study shows the value of the head and not the necessity of the encoder.

Permutation SHAP on the whole pipeline named the flow features behind each prediction. Packet-size statistics carry the largest share of the attribution in every class, and the two TCP initial-window fields carry 14.8% overall and 28.2% in Webattack; a lookup on those two fields alone labels 74.1% of the explained flows correctly, which is consistent with, but does not prove, a fingerprint of the capture hosts. The explanations satisfied completeness and beat a random ranking in the faithfulness checks, but they cost 4.95 s per flow, and their stability was measured only in the latent space.

The next steps are the controls and splits listed in Table 16, in particular a retraining without the window fields and a session- or time-grouped evaluation on a second collection. Until they are run, the scores describe this benchmark and not deployed traffic.

Declarations

Funding
This research did not receive any specific funding.

Author contributions
M.Z.A. did the data analysis and wrote the main manuscript, and I.A.C. wrote the draft and supervised the complete work. All authors reviewed the manuscript.

Competing interests
Not applicable.

Ethics approval
Not applicable.

Consent to participate
Not applicable.

Consent to participate
Not applicable.

Consent for publication
Not applicable.

Data availability
The data used in this research is publicly available and were obtained from open-access sources. Online available at: https://www.kaggle.com/datasets/dhoogla/cicidscollection. Any additional datasets generated or analyzed during the study can be made available upon reasonable request to the corresponding author.

Use of artificial intelligence tools

Artificial intelligence tools were used solely for language editing and grammatical refinement of the manuscript. No AI tools were used for scientific reasoning, data analysis, or result generation.

Competing interests

The authors declare no competing interests.

References

1. Sharafaldin, I., A.H. Lashkari, and A.A. Ghorbani, *Toward generating a new intrusion detection dataset and intrusion traffic characterization.* ICISSp, 2018. **1**(2018): p. 108-116.
2. Sharafaldin, I., et al. *Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy*. in *2019 international carnahan conference on security technology (ICCST)*. 2019. IEEE.
3. Khraisat, A., et al., *Survey of intrusion detection systems: techniques, datasets and challenges.* Cybersecurity, 2019. **2**(1): p. 20.
4. Ring, M., et al., *A survey of network-based intrusion detection data sets.* Computers & security, 2019. **86**: p. 147-167.
5. Engelen, G., V. Rimmer, and W. Joosen. *Troubleshooting an intrusion detection dataset: the CICIDS2017 case study*. in *2021 IEEE Security and Privacy Workshops (SPW)*. 2021. IEEE.
6. Lanvin, M., et al. *Errors in the CICIDS2017 dataset and the significant differences in detection performances it makes*. in *International Conference on Risks and Security of Internet and Systems*. 2022. Springer.
7. Arp, D., et al. *Dos and don'ts of machine learning in computer security*. in *31st USENIX Security Symposium (USENIX Security 22)*. 2022.
8. Grinsztajn, L., E. Oyallon, and G. Varoquaux, *Why do tree-based models still outperform deep learning on typical tabular data?* Advances in neural information processing systems, 2022. **35**: p. 507-520.
9. Shwartz-Ziv, R. and A. Armon, *Tabular data: Deep learning is not all you need.* Information fusion, 2022. **81**: p. 84-90.
10. Gorishniy, Y., et al., *Revisiting deep learning models for tabular data.* Advances in neural information processing systems, 2021. **34**: p. 18932-18943.
11. Manocchio, L.D., et al., *Flowtransformer: A transformer framework for flow-based network intrusion detection systems.* Expert Systems with Applications, 2024. **241**: p. 122564.
12. Moczkodan, Z. and H. Ragab, *Do Transformers Actually Help Intrusion Detection? A Temporal Sequence Evaluation on CIC-IDS2017.* arXiv preprint arXiv:2606.11098, 2026.
13. Talukder, M.A., et al., *Machine learning-based network intrusion detection for big and imbalanced data using oversampling, stacking feature embedding and feature extraction.* Journal of big data, 2024. **11**(1): p. 33.
14. AlFuraih, D., L. Mhamdi, and A.S. Karar, *Explainable Hybrid CNN–XGBoost Framework for Multi-Class IoT Intrusion Detection with Leakage-Aware Feature Selection.* Applied System Innovation, 2026. **9**(3): p. 49.
15. Lundberg, S.M. and S.-I. Lee, *A unified approach to interpreting model predictions.* Advances in neural information processing systems, 2017. **30**.
16. Lundberg, S.M., et al., *From local explanations to global understanding with explainable AI for trees.* Nature machine intelligence, 2020. **2**(1): p. 56-67.
17. Strumbelj, E. and I. Kononenko, *An efficient explanation of individual classifications using game theory.* The Journal of Machine Learning Research, 2010. **11**: p. 1-18.
18. Warnecke, A., et al. *Evaluating explanation methods for deep learning in security*. in *2020 IEEE european symposium on security and privacy (EuroS&P)*. 2020. IEEE.
19. Neupane, S., et al., *Explainable intrusion detection systems (x-ids): A survey of current methods, challenges, and opportunities.* IEEE Access, 2022. **10**: p. 112392-112415.
20. Zhou, H., et al. *Informer: Beyond efficient transformer for long sequence time-series forecasting*. in *Proceedings of the AAAI conference on artificial intelligence*. 2021.
21. Chen, T. and C. Guestrin. *Xgboost: A scalable tree boosting system*. in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 2016.
22. Hussein, S.A. and S.R. Répás, *A hybrid intrusion detection framework using deep autoencoder and machine learning models.* AI, 2026. **7**(2): p. 39.

23. Butt, M.U., A. Hotho, and D. Schlör, *Evaluating Tabular Representation Learning for Network Intrusion Detection.* arXiv preprint arXiv:2605.02519, 2026.
24. Sharafaldin, I., A. Lashkari, and A. Ghorbani, *CICIDS2017: Canadian Institute for Cybersecurity Intrusion Detection System Dataset*. 2018.
25. Alrikabi, R., *Performance analysis of flow-based attacks detection on CSE-CIC-IDS2018 dataset using deep learning.* Indonesian Journal of Electrical Engineering and Computer Science, 2020.
26. Saheb, M.C.P., et al. *A review of DDoS evaluation dataset: CICDDoS2019 dataset*. in *International Conference on Energy Systems, Drives and Automations*. 2021. Springer.
27. Vaswani, A., et al., *Attention is all you need.* Advances in neural information processing systems, 2017. **30**.
28. Liu, J., et al. *Multidomain transformer-based deep learning for early detection of network intrusion*. in *GLOBECOM 2023-2023 IEEE Global Communications Conference*. 2023. IEEE.
29. Almuhanna, R. and S. Dardouri, *A deep learning/machine learning approach for anomaly based network intrusion detection.* Frontiers in Artificial Intelligence, 2025. **8**: p. 1625891.
30. Chawla, N.V., et al., *SMOTE: synthetic minority over-sampling technique.* Journal of artificial intelligence research, 2002. **16**: p. 321-357.
31. Janati, M. and F. Messaoudi, *Network Behavior–Driven Intrusion Detection: A Hybrid Deep Learning and XGBoost Approach*, in *AI-Driven Security for Next-Generation IoT Systems*. 2026, Springer. p. 107-118.
32. Wang, M., et al., *An explainable machine learning framework for intrusion detection systems.* IEEE access, 2020. **8**: p. 73127-73141.
33. Mane, S. and D. Rao, *Explaining network intrusion detection system using explainable AI framework.* arXiv preprint arXiv:2103.07110, 2021.
34. Marino, D.L., C.S. Wickramasinghe, and M. Manic. *An adversarial approach for explainable ai in intrusion detection systems*. in *IECON 2018-44th Annual Conference of the IEEE Industrial Electronics Society*. 2018. IEEE.
35. Şahin, F., *HiT-IDS: A Hierarchical Transformer with Intrinsic Explainability for Network Intrusion Detection.* 2026.
36. Janzing, D., L. Minorics, and P. Blöbaum. *Feature relevance quantification in explainable AI: A causal problem*. in *International Conference on artificial intelligence and statistics*. 2020. PMLR.
37. Sundararajan, M. and A. Najmi. *The many Shapley values for model explanation*. in *International conference on machine learning*. 2020. PMLR.
38. Chen, H., et al., *True to the model or true to the data?* arXiv preprint arXiv:2006.16234, 2020.
39. Kumar, I.E., et al. *Problems with Shapley-value-based explanations as feature importance measures*. in *International conference on machine learning*. 2020. PMLR.
40. Petsiuk, V., A. Das, and K. Saenko, *Rise: Randomized input sampling for explanation of black-box models.* arXiv preprint arXiv:1806.07421, 2018.
41. Hurley, N. and S. Rickard, *Comparing measures of sparsity.* IEEE Transactions on Information Theory, 2009. **55**(10): p. 4723-4741.
42. Alvarez-Melis, D. and T.S. Jaakkola, *On the robustness of interpretability methods.* arXiv preprint arXiv:1806.08049, 2018.
43. D'Hooge, L., *CIC-IDS-Collection*, in *Kaggle*. 2022, Kaggle.
44. Kingma, D.P. and J. Ba, *Adam: A method for stochastic optimization.* arXiv preprint arXiv:1412.6980, 2014.
45. Breiman, L., *Random forests.* Machine learning, 2001. **45**(1): p. 5-32.
46. Hochreiter, S. and J. Schmidhuber, *Long short-term memory.* Neural computation, 1997. **9**(8): p. 1735-1780.
47. Cortes, C. and V. Vapnik, *Support-vector networks.* Machine learning, 1995. **20**(3): p. 273-297.
48. Chicco, D. and G. Jurman, *The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation.* BMC genomics, 2020. **21**(1): p. 6.
49. Brown, L.D., T.T. Cai, and A. DasGupta, *Interval estimation for a binomial proportion.* Statistical science, 2001: p. 101-117.
50. Paszke, A., et al., *Pytorch: An imperative style, high-performance deep learning library.* Advances in neural information processing systems, 2019. **32**.
51. Pedregosa, F., et al., *Scikit-learn: Machine learning in Python.* the Journal of machine Learning research, 2011. **12**: p. 2825-2830.