CYBER RANGE NETWORK CLASSIFICATION

Ashley Alt
RIT
Rochester, NY
aea7842@rit.edu

William Faber
RIT
Rochester, NY
wmf1426@rit.edu

Donovan Hwang
RIT
Rochester, NY
dwh4755@rit.edu

Roberto Reade
RIT
Rochester, NY
rar9027@rit.edu

ABSTRACT

Network intrusion detection increasingly relies on machine learning (ML) to identify malicious behaviors within high-volume traffic. However, most ML approaches require structured and labeled data, whereas real network captures exist primarily as raw packet-level PCAPs. This project develops a complete, reproducible processing pipeline that transforms more than 100 GB of raw UNSW-NB15 packet captures into flow-level records suitable for multiclass attack classification.

Using a custom Python streaming parser, we extracted packet metadata without loading entire PCAPs into memory, reconstructed bidirectional flows via canonicalized 5-tuples, and computed a comprehensive set of statistical, temporal, and protocol features inspired by the original UNSW-NB15 feature design. We then aligned these flows with the dataset's 2.54 million ground-truth labels using a composite key based on IP addresses, ports, protocol, and rounded timestamps, producing a fully labeled, ML-ready dataset.

We evaluated three major ML models, Random Forests, XGBoost, and a Deep Feedforward Neural Network, on the nine UNSW-NB15 attack categories. All models achieved high overall accuracy, with XGBoost providing the strongest performance on most attack types. Imbalanced classes such as Backdoor, Shellcode, and Worms remained challenging across all models, highlighting the impact of real-world traffic skew. Overall, this work provides an end-to-end flow generation and labeling pipeline for raw PCAP data and a comparative evaluation of modern ML techniques for network intrusion detection.

1 Introduction

Modern network environments produce massive volumes of packet-level data, making manual inspection impractical and driving the need for automated, data-driven intrusion detection. Machine learning has become central to network defense because of its ability to identify complex behavioral patterns, detect deviations from normal activity, and classify diverse cyber attacks [1]. However, ML models require structured, labeled, and feature-rich data, whereas real traffic is typically collected in raw packet-capture (PCAP) form, which is high-volume, noisy, and unsuitable for direct ML ingestion.

To bridge this gap, our project builds a complete processing pipeline that converts raw network traffic into flow-level records and evaluates multiple ML models for multiclass attack classification. We base our work on the UNSW-NB15 dataset, a widely used benchmark dataset for intrusion detection research. The raw UNSW-NB15 traffic was generated using the IXIA PerfectStorm tool in the Cyber Range Lab at UNSW Canberra and captures a hybrid of realistic benign activity and nine modern attack categories, including *Fuzzers, Analysis, Backdoor, DoS, Exploits, Generic, Reconnaissance, Shellcode*, and *Worms* [2]. The dataset consists of over 100 GB of PCAPs, associated Bro and Argus logs, and detailed ground-truth CSV files containing 2.54 million labeled flows.

Because UNSW-NB15 is distributed primarily as raw packet captures, significant preprocessing is required before it can be used for ML classification. Our project reconstructs bidirectional flows directly from the PCAP-derived packet data, computes statistical features such as inter-arrival times, jitter metrics, directional byte counts, packet counts, and protocol-level behaviors, and then performs a composite-key matching process to align each flow with its ground-truth attack label. This transformation converts unstructured raw traffic into a structured ML-ready dataset consistent with prior work on network intrusion detection [3].

After building the processed dataset, we evaluate three major ML models: Random Forests, XGBoost, and a Deep Feedforward Neural Network. These models were selected because of their strong historical performance in intrusion detection systems and their ability to handle large feature sets, nonlinear patterns, and multiclass classification tasks. Our goal is to compare their performance, identify model strengths and weaknesses, and understand how well modern ML techniques can distinguish between the nine UNSW-NB15 attack types, particularly in the presence of dataset imbalance.

Overall, this work contributes a reproducible pipeline for transforming raw UNSW-NB15 PCAP data into a high-quality flow-level dataset and provides an comparison of ML models for multiclass network attack classification.

2 Literature Review

Early intrusion detection systems (IDS) were primarily signature-based, relying on manually engineered rules or misuse signatures to flag known attack patterns. While effective against previously observed threats, these systems struggle with zero-day attacks and evolving adversarial behavior. As a result, anomaly-based and machine-learning-driven IDS have become a major research focus, using statistical patterns in network traffic to distinguish benign and malicious activity. Sommer and Paxson caution that blindly applying machine learning to intrusion detection is insufficient without careful feature engineering and a clear understanding of operational constraints, but they also highlight the potential of data-driven methods when grounded in robust network semantics [1].

The UNSW-NB15 dataset was created to address limitations of older benchmarks such as KDD99, providing a more modern representation of network traffic and attack types. Moustafa and Slay describe how traffic was generated in the UNSW Canberra Cyber Range using the IXIA PerfectStorm tool, combining realistic background activity with nine contemporary attack categories and deriving 49 flow-based features using Bro, Argus, and custom aggregation algorithms [2]. A follow-up statistical evaluation demonstrates that UNSW-NB15 exhibits more realistic distributions, reduced redundancy, and better support for modern anomaly detection techniques compared to legacy datasets [3]. Because of these properties, UNSW-NB15 has become a standard benchmark for training and evaluating machine-learning-based IDS.

A large body of work has explored different machine learning models and feature-processing strategies on UNSW-NB15 and related datasets. Sarhan et al. investigate feature extraction and dimensionality reduction techniques such as PCA, autoencoders, and LDA across multiple intrusion detection datasets (including UNSW-NB15), showing that no single model or feature-extraction method dominates across all settings and that dataset characteristics heavily influence model performance [4]. Wu et al. propose Pelican, a deep residual network tailored for intrusion detection, and report high detection rates with reduced false alarms on UNSW-NB15 compared to traditional machine learning baselines [5]. More recently, Corea et al. apply explainable AI methods to compare a range of classifiers on UNSW-NB15 and find that tree-based models such as Random Forest achieve strong performance while often relying on a small subset of highly influential features, underscoring the central role of feature engineering in network intrusion detection [6].

Our work is positioned within this line of research but differs in a key way: rather than starting from the preprocessed UNSW-NB15 CSV files, we begin at the raw PCAP level and construct our own flow representation before training models. This aligns conceptually with prior flow-based approaches but more closely matches how an operational defender would process live network captures. By reconstructing flows from PCAP, re-deriving UNSW-style features, and then applying Random Forest, XGBoost, and a deep feedforward neural network, we can both validate the effectiveness of these models and evaluate the robustness of the UNSW feature design when reproduced from raw traffic.

3 Methodology

Data Analysis

The raw UNSW-NB15 capture consists of roughly 100 GB of packet-level traffic, which we converted into a flow-level CSV representation before analysis (see Section 3.1 for details).. After cleaning and aggregation, the resulting dataset contained on the order of millions of flows spanning benign activity and all nine attack categories.

We began by examining the class distribution of the labeled flows. As expected for a realistic enterprise network, the dataset is heavily imbalanced: benign traffic dominates the dataset, while several attack types (such as Backdoor, Shellcode, and Worms) appear relatively rarely. This imbalance has important implications for classifier design and motivated our later use of models and metrics that are robust to skewed label distributions.

Next, we generated descriptive statistics for key flow features, including total bytes, packet counts, flow duration, and basic timing measures. Benign flows tended to show longer durations and more balanced byte transfer between source and destination, whereas high-volume attacks such as DoS, Exploit, and Generic exhibited short-lived, bursty behavior with large byte counts and small inter-arrival times. Reconnaissance-style activity was characterized by small flows with low byte counts and minimal payload transfer, consistent with scanning and probing behavior.

These summary statistics confirmed that different attack categories leave distinct statistical signatures in the flow features. This, in turn, provided evidence that our flow-based representation is suitable for supervised learning and helped guide which features to emphasize in our subsequent machine learning experiments.

3.1 Data Preprocessing

The raw UNSW-NB15 data is distributed as large PCAP files containing more than 100 GB of packet-level network traffic. Because packet captures are not directly usable for machine-learning-based intrusion detection, we developed a custom Python streaming pipeline to extract packet information into a CSV file using Scapy, then converting the uncleaned CSV file into structured flow-level features. This preprocessing stage included four major components: packet parsing, flow reconstruction, feature aggregation, and ground-truth labeling.

Packet Parsing

The PCAP files in UNSW-NB15 are extremely large, collectively exceeding 100 GB of raw packet data. Loading these captures into memory is infeasible for normal computational environments, so we implemented a fully streaming parsing pipeline using Python and Scapy. Instead of processing the PCAP as a whole, packets were read sequentially in a sliding-window fashion, and only a small batch of packets was held in memory at any time.

To accomplish this, we exported packet metadata into an intermediate "uncleaned" CSV file in incremental chunks. After several thousand packets were processed, the batch was immediately flushed to disk, the in-memory structures were cleared, and the parser advanced the window to the next segment of the PCAP. This strategy allowed us to process the entire dataset without exceeding system memory limits.

The fields extracted during this stage included:

- Source and destination IP addresses
- Source and destination ports
- Transport-layer protocol (TCP, UDP, ICMP)
- Packet size and raw payload (hex-encoded)
- TCP flags, TTL values, and window/sequence information
- Inferred application-level service (HTTP, DNS, FTP)
- Packet timestamp (UNIX epoch)

This streaming approach ensured that packet-level data could be analyzed, converted, and stored efficiently while maintaining compatibility with the follow-on flow reconstruction stage. By decoupling packet extraction from flow aggregation and enforcing a constant-memory processing model, our parser successfully handled the full UNSW-NB15 raw traffic without requiring specialized hardware.

Flow Reconstruction

To transform packets into flows, we employed a bidirectional 5-tuple identification scheme based on source IP, source port, destination IP, destination port, and protocol. Packets sharing the same canonicalized 5-tuple were aggregated into a single flow regardless of direction. This required normalizing each connection so that the "origin" of the flow was determined consistently (e.g., the endpoint with the numerically smaller IP address (and port, if needed) was assigned as the forward direction).

Because both the uncleaned CSV file was too large to fit into memory, we implemented a fully streaming flow-generation pipeline. Packets were processed in sequential chunks rather than loaded all at once to create flows. A sliding-window buffer maintained only the most recent portion of packet records in memory while previously completed flow entries were incrementally written to disk. As each new packet arrived, the corresponding flow statistics were updated, and flows were flushed to the output file after 60 seconds of inactivity (idle timeout). This chunked, window-based approach mirrors the design of real network collectors, enabling us to handle large-scale traffic efficiently without exceeding memory limits.

Feature Aggregation

Once packets were reconstructed into bidirectional flows, we extracted a comprehensive set of statistical, temporal, and protocol-level features inspired by the original UNSW-NB15 feature design. These features fall into five major categories: volume-based statistics, timing characteristics, protocol/header features, behavioral indicators, and duration/service information. All features were computed incrementally within the streaming pipeline to avoid storing raw packet histories in memory.

- **Volume-based statistics:** These features quantify the size and directionality of each flow. We compute total bytes (sbytes, dbytes), packet counts (Spkts, Dpkts), and mean packet sizes (smeansz, dmeansz) for each direction. High packet rates with small sizes often indicate DoS traffic, while asymmetric byte counts can signal exploit attempts or scanning behavior.
- **Timing characteristics:** To capture temporal patterns in network communication, we extract average inter-arrival times (Sintpkt, Dintpkt) and packet jitter (Sjit, Djit). These values are computed online using Welford's algorithm to maintain numerical stability. Burstiness, near-zero timing intervals, or highly irregular timing patterns are strong indicators of malicious activity.
- **Protocol and header-derived fields:** We include low-level indicators such as protocol type, connection state, time-to-live values (sttl, dttl), window sizes, and TCP handshake timing features including synack, ackdat, and round-trip estimates (tcprtt). These features help distinguish normal connections from spoofed packets, incomplete handshakes, or abnormal transport-layer behavior.
- **Behavioral and statistical indicators:** Several features capture higher-level flow behavior. Examples include counts of distinct connection states within flows (ct_state_ttl), repetitive use of the same IP/port pair (is_sm_ips_ports), HTTP method counts (ct_flw_http_mthd), FTP login attempts (is_ftp_login), and historical connection frequency to the same service or host (ct_srv_src, ct_srv_dst, ct_dst_ltm, ct_src_ltm). These indicators help reveal brute-force attempts, scans, botnet propagation, and other high-level malicious patterns.
- **Flow duration and service identification:** We store flow duration (dur) and start/end timestamps, which help differentiate short-lived exploit bursts from long-running encrypted channels. Service types (e.g., HTTP, DNS, SSH) are inferred using well-known ports and packet heuristics, enabling identification of attacks hidden within otherwise normal application protocols.

Because payload bytes are not required for flow-based classification and significantly increase memory usage, we did not retain packet payloads during preprocessing. Only header-level information (IP, TCP/UDP, timing, and counters) was used for feature extraction.

Ground-Truth Alignment

The original UNSW-NB15 dataset includes four CSV files with ground-truth labels for all flows. After generating our own flow-level dataset, we aligned each reconstructed flow with the official labels. Because PCAP timestamps rarely match the ground-truth timestamps exactly, we rounded our flow start time (Stime) to the nearest second and constructed a composite key consisting of: srcip, sport, dstip, dsport, protocol, rounded timestamp. This key was matched against the concatenated ground-truth tables (approximately 2.54 million labeled flows). When a match was found, the flow was assigned the corresponding attack category (one of nine possible attacks) and a binary label indicating normal or malicious traffic. Flows without a match were labeled as benign.

Final Dataset

After preprocessing, we obtained a cleaned, feature-rich dataset that preserves the structure of UNSW-NB15 while capturing flows directly from raw packet data. The resulting data contains all 49 canonical flow features and trusted ground-truth labels, making it suitable for supervised classification with Random Forest, XGBoost, and deep neural networks.

3.2 ML Models Used

We evaluated three supervised learning models for multiclass intrusion detection on the processed UNSW-NB15 flows: a Random Forest classifier, an XGBoost gradient-boosted tree model, and a Deep Feedforward Neural Network (DFFNN). All models were configured to predict the nine attack categories plus a benign class.

Random Forest

Random Forest is an ensemble of decision trees trained on bootstrapped samples of the data, with feature subsampling at each split to reduce variance and overfitting. It is widely used in intrusion detection because it handles high-dimensional feature spaces, mixed feature scales, and nonlinear decision boundaries with relatively little hyperparameter tuning. In our experiments, we:

- Used `n_estimators` (number of trees) set to 300,
- Limited tree depth to unconstrained,
- Employed Gini impurity as the split criterion,
- Trained using class-stratified samples to preserve label distribution.

This model serves as a strong baseline for flow-based intrusion detection and provides interpretable feature importance scores that help understand which flow attributes contribute most to classification performance.

XGBoost

XGBoost is a gradient boosting framework that builds an ensemble of shallow decision trees sequentially, where each tree corrects the errors of the previous ones. It is known for strong performance on tabular data, especially in security analytics and intrusion detection tasks. For our XGBoost experiments, we:

- Used 800 boosting rounds with early stopping based on validation loss,
- Chose a learning rate of .03,
- Set maximum tree depth to 10,
- Applied uniform class weights.

XGBoost's ability to model complex interactions between flow features with fine-grained regularization makes it a strong candidate for distinguishing subtle differences between attack categories.

Deep Feedforward Neural Network

Our deep feedforward neural network consists of multiple fully connected layers with nonlinear activation functions. Unlike tree-based methods, which partition feature space using axis-aligned splits, the DFFNN learns distributed representations and can automatically capture interactions between features. The network architecture used in this project includes:

- An input layer matching the number of numeric flow features,
- Two hidden layers with 256 and 128 units respectively, each followed by a ReLU activation function,
- Dropout for regularization,
- A final fully connected layer with a softmax output over the ten classes (benign + nine attack types).

Before training, numeric features were standardized to zero mean and unit variance to stabilize gradient-based optimization. The network was trained using the Adam optimizer with a cross-entropy loss function and mini-batch gradient descent.

3.3 Analysis Methods

To evaluate our models, we split the labeled flow dataset into training, validation, and test sets using a stratified sampling strategy to preserve the original class distribution in each split. Concretely, we used:

- A training set of 60% of total data for Deep Feedforward Neural Network,
- A training set of 80% of total data for Random Forest and XGBoost,
- A validation set of 20% for Deep Feedforward Neural Network only,
- A held-out test set of 20% for final performance reporting for all models.

Our primary evaluation focuses on multiclass classification performance across the ten labels (benign plus nine attack types). We report:

- Overall accuracy on the test set,
- Per-class precision, recall, and F1-score derived from the confusion matrix,
- Macro-averaged and weighted-averaged F1-scores to account for class imbalance.

Because the dataset is heavily imbalanced, with benign traffic dominating and several attack categories appearing relatively rarely, we place particular emphasis on macro-averaged metrics and per-class results rather than accuracy alone. For each model, we also inspect the confusion matrix to understand which attack types are frequently misclassified and whether errors are concentrated among specific classes (e.g., confusion between Exploits and Generic attacks).

Where appropriate, we compare models using the same train/validation/test splits to ensure a fair comparison and discuss trade-offs in terms of detection performance on rare attack types versus benign traffic.

4 Results

In this section, we present the performance of Random Forest, XGBoost, and the Deep Feedforward Neural Network on the processed UNSW-NB15 flow dataset. All results are reported on the held-out test set described in Section 3.

4.1 Random Forest Performance

The initial phase focused on balancing the dataset, which was heavily skewed toward the Benign class (684,009 samples). To improve sensitivity to attacks, the Benign class was downsampled to 2,803 instances, creating a more balanced pool of attack and non-attack traffic. SMOTE (Synthetic Minority Over-sampling Technique) was then applied to the minority classes, generating synthetic samples and expanding the dataset so that each class had 25,227 samples. This ensured the Random Forest classifier had sufficient and balanced data to learn the characteristics of each attack category.

```
Original dataset shape: (689616, 49)
Attack category counts after cleaning:
 attack_cat
Benign            684009
Fuzzers             2282
Exploits            2009
Reconnaissance       573
DoS                  328
Generic              274
Shellcode             80
Backdoors             49
Worms                 12
Name: count, dtype: int64
Counts after Benign downsampling:
attack_num
0    2803
1    2282
5    2009
7     573
4     328
6     274
8      80
3      49
9      12
Name: count, dtype: int64
Shape after SMOTE: (25227, 47) (25227,)
Accuracy: 0.9078478002378121
```

Figure 1: Data Shape: Original, Downsampling, SMOTE

The performance of the model was first examined using the confusion matrix. It showed strong precision for Benign (556), Fuzzers (557), Backdoors (560), Shellcode (559) and Worms (560). Challenges were observed in the DoS and Backdoors classes. 173 DoS samples were misclassified as Backdoors, and Backdoors, despite perfect recall, absorbed 57 samples from the Generic class. This resulted in a higher false positive rate. These misclassifications suggest overlapping features between these attack types.

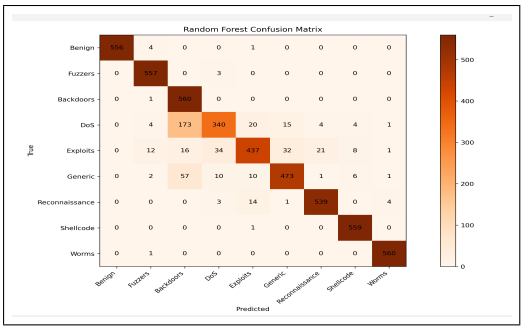

Figure 2: Confusion Matrix

The classification report confirmed overall strong performance, with a test set accuracy of 0.91. Macro and weighted averages for precision, recall, and f1-score were consistently around 0.91–0.92. Benign (f1-score: 1.00), Worms (f1-score: 0.99), and Shellcode (f1-score: 0.98) were the top-performing classes. DoS had the lowest f1-score (0.72) due to a recall of 0.61, while Backdoors had an f1-score of 0.82, limited by a precision of 0.69. Overall, the model performs well, but distinguishing DoS, Backdoors, and Generic traffic remains the primary area for improvement.

Classification Report:

	precision	recall	f1-score	support
Benign	1.00	0.99	1.00	561
Fuzzers	0.96	0.99	0.98	560
Backdoors	0.69	1.00	0.82	561
DoS	0.87	0.61	0.72	561
Exploits	0.90	0.78	0.84	561
Generic	0.91	0.84	0.88	560
Reconnaissance	0.95	0.96	0.96	561
Shellcode	0.97	1.00	0.98	560
Worms	0.99	1.00	0.99	561
accuracy			0.91	5046
macro avg	0.92	0.91	0.91	5046
weighted avg	0.92	0.91	0.91	5046

Figure 3: Classification Report

4.2 XGBoost Performance

Initially, running XGBoost on the dataset resulted in what seemed to be very good numbers on the surface. Overall accuracy was extremely high, to the point where it rounded to 1.00 on overall average as well as weighted average. However, looking closer at the macro average and specific classes, we can see that while it did perform well on benign packets, it really struggled with some of the rare attack types that existed. Even with lots of hyperparameter tuning, the metrics varied by negligible amounts.

	precision	recall	f1-score	support
Fuzzers	0.95	0.99	0.97	571
Backdoors	0.00	0.00	0.00	12
Benign	1.00	1.00	1.00	171002
DoS	0.32	0.28	0.30	82
Exploits	0.77	0.88	0.83	502
Generic	0.67	0.20	0.31	69
Reconnaissance	0.92	0.80	0.85	143
Shellcode	0.60	0.45	0.51	20
Worms	0.00	0.00	0.00	3
accuracy			1.00	172404
macro avg	0.58	0.51	0.53	172404
weighted avg	1.00	1.00	1.00	172404

Figure 4: Classification Report Before SMOTE

Since the biggest issue was with data imbalance rather than the machine learning algorithms themselves, SMOTE, along with another technique to limit the amount of benign samples trained on, was implemented. This

resulted in a much more even distribution of packets, and, therefore, better results for the rarer attack categories as well as an overall accuracy of .9168. As shown by Figure 5, all metrics improved greatly, especially for rarer attack categories. However, the results for the extremely rare classes, such as worms, should be taken with a grain of salt. Since the original dataset was so skewed, with only 12 pcap packets for worms, SMOTE is replicating a lot of the same data, which leads to that 1.00 accuracy, which might not be the case for more data.

Accuracy: 0.9167657550535078

Classification Report:

	precision	recall	f1-score	support
Fuzzers	0.98	0.98	0.98	561
Backdoors	0.69	1.00	0.82	560
Benign	1.00	1.00	1.00	561
DoS	0.87	0.62	0.72	561
Exploits	0.92	0.83	0.87	561
Generic	0.93	0.86	0.89	560
Reconnaissance	0.97	0.98	0.97	561
Shellcode	0.98	0.99	0.99	560
Worms	1.00	1.00	1.00	561
accuracy			0.92	5046
macro avg	0.93	0.92	0.92	5046
weighted avg	0.93	0.92	0.92	5046

Figure 5: Classification Report After SMOTE

The confusion matrix shows a strong diagonal across each class. Since the metrics were good overall for the classification matrix after SMOTE, it is reflected here in the new confusion matrix. Additionally, the data balancing also makes it so that benign data isn't contributing to 99% of our overall data, resulting in the balanced diagonal. Overall, there weren't many false positives/negatives, reflected by the very low numbers outside the diagonal. One section of interest is DoS attacks; 173 packets were labeled as backdoors, resulting in lower precision for backdoor and DoS attacks.

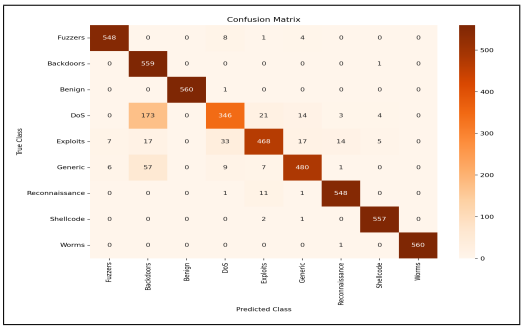

Figure 6: Confusion Matrix after SMOTE

4.3 Deep Feedforward Neural Network Performance

The initial Deep Feedforward Neural Network (DFFN) was trained on the original dataset without resampling. Although the model achieved a high overall accuracy of 99.56%, this metric was misleading due to the extreme class imbalance. The classification report [Figure 7] showed that the benign traffic dominated the dataset, and the model classified it with extremely high precision and recall of 0.9978 and 0.9993. The data also included several minority attack categories, such as Backdoors, Shellcode, Worms, and Generic, which showed near-zero precision, recall, and F1-Score. One of the attack categories, Analysis, had no instances in the training/validation set at all, making the classification impossible.

This demonstrates that the model learned to heavily favor the benign majority class and ignore the rarer attacks. Overall, the pre-SMOTE performance appeared strong but failed to generalize across all attack types due to severe data imbalance.

	precision	recall	f1-score	support
Benign	0.9978	0.9993	0.9985	136802
Fuzzers	0.6623	0.4386	0.5277	456
Analysis	0.0000	0.0000	0.0000	0
Backdoors	0.0000	0.0000	0.0000	10
DoS	0.4118	0.3182	0.3590	66
Exploits	0.7171	0.8259	0.7676	402
Generic	0.2500	0.0727	0.1127	55
Recon	0.6667	0.4348	0.5263	115
Shellcode	0.4000	0.3750	0.3871	16
Worms	0.0000	0.0000	0.0000	2
accuracy			0.9956	137924
macro avg	0.4106	0.3464	0.3679	137924
weighted avg	0.9949	0.9956	0.9951	137924

Figure 7: DFFN Confusion Report before SMOTE

To mitigate class imbalance, SMOTE was applied to oversample minority attack categories in the training set. We also utilized decreasing the overall amount of benign samples within the dataset to further reduce the imbalance. As you can see in Figure 8, after making these changes, the model reached a test accuracy of 82.78%, which is lower than the misleading 99% value from the imbalanced model; however, this accuracy is far more meaningful because the model is now correctly evaluated across all classes. The post SMOTE improves showed strong precision and recall across nearly all attack categories. The minority classes, such as Backdoors, which achieved an F1-score of 0.809, Shell code, with an F1-score of 0.987, and Worms, which achieved an F1-score of 0.989. While SMOTE did help with the data imbalance, there are still certain categories that remained challenging, like DOS, with a recall of 0.382, and exploits, with a recall of 0.545, likely due to overlapping features and the difficulty of synthesizing some types of attack behavior.

Classification Report:

	precision	recall	f1-score	support
Benign	1.0000	0.9893	0.9946	561
Fuzzers	0.8591	0.7839	0.8198	560
Backdoors	0.6843	0.9893	0.8090	561
DoS	0.7839	0.3815	0.5132	561
Exploits	0.8095	0.5455	0.6518	561
Generic	0.7017	0.8232	0.7576	560
Recon	0.7486	0.9394	0.8332	561
Shellcode	0.9588	0.9982	0.9781	560
Worms	0.9774	1.0000	0.9885	561
accuracy			0.8278	5046
macro avg	0.8359	0.8278	0.8162	5046
weighted avg	0.8359	0.8278	0.8162	5046

Macro F1: 0.8162087784611558

Figure 8: DFFN Classification Report after SMOTE

The confusion matrix in Figure 9 shows that the deep feedforward neural network was able to accurately distinguish most attack categories after using SMOTE. There is a strong diagonal that can be observed across nearly all classes, indicating a consistent classification. This is particularly for Benign, Analysis, Generic, Recon, and Shellcode, which show near-perfect performance. Overall, the confusion matrix reflects substantial improvement compared to the pre-SMOTE model, where minority attack categories were nearly ignored; it demonstrates a more balanced recognition of all attack types, consistent with the observed 83% post-SMOTE accuracy seen in Figure 8.

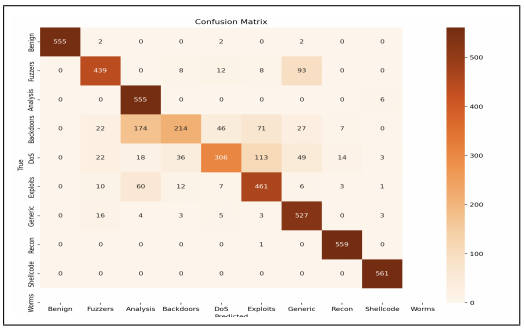

Figure 9: DFFN Confusion Matrix after SMOTE

4.4 Comparative Discussion

The dataset chosen was initially extremely imbalanced. The minority attack classes were heavily underrepresented, so we applied SMOTE to generate synthetic samples and bring the classes closer to balance. This significantly improved the models' ability to learn meaningful patterns.

After performing hyperparameter tuning and evaluating all three models, we relied on classification reports and confusion matrices to understand the strengths and weaknesses of each model. Once all testing of the models was complete, we selected the model that performed best for our goal of accurate multi-class intrusion detection across all 10 categories. Based on preliminary research and prior academic findings, we had predicted that XGBoost would likely emerge as the strongest performer. After testing all three models using the same evaluations, the results confirmed this expectation. XGBoost consistently performed the best across the minority attack classes and the overall 10-category classification task. Based on these results and its strong performance, we selected XGBoost as the best model for our project.

5 Conclusion

This project implemented an end-to-end pipeline that transforms over 100 GB of raw UNSW-NB15 PCAP data into a labeled, flow-level dataset suitable for multiclass intrusion detection. Starting from packet captures, we used a streaming parser to extract header-level metadata, reconstructed bidirectional flows using a canonical 5-tuple, and computed a rich set of statistical, temporal, and protocol features inspired by the original UNSW-NB15 design. We then aligned our reconstructed flows with the official ground-truth labels via a composite key, producing a fully labeled dataset that closely mirrors how an operational defender would process live traffic.

On top of this processed dataset, we evaluated three supervised learning models: Random Forest, XGBoost, and a Deep Feedforward Neural Network. All models achieved high overall accuracy, with XGBoost being the general top performer and our pick for the best model to use for this data. However, rare attack types remained challenging across all models, highlighting the persistent impact of class imbalance and the limited coverage of certain behaviors in training data.

Our findings support several key takeaways. First, careful preprocessing and flow construction from raw PCAPs are essential to unlock the full potential of machine learning in network intrusion detection. Second, tree-based ensemble methods remain highly effective for tabular flow data and often match or exceed deep neural networks when feature engineering is strong. Finally, future work should focus on improving the detection of low-frequency attacks through imbalance-aware training strategies, additional data, or hybrid detection schemes that combine supervised classification with anomaly detection.

By releasing a reproducible processing approach and documenting our experimental results, we hope to provide a useful reference for future research that seeks to bridge the gap between raw network captures and deployable ML-based intrusion detection systems.

References

[1] Robin Sommer and Vern Paxson. Outside the closed world: On using machine learning for network intrusion detection. In *2010 IEEE Symposium on Security and Privacy*, pages 305–316. IEEE, 2010.

[2] Nour Moustafa and Jill Slay. Unsw-nb15: A comprehensive data set for network intrusion detection systems. In *2015 Military Communications and Information Systems Conference (MilCIS)*, pages 1–6. IEEE, 2015.

[3] Nour Moustafa and Jill Slay. The evaluation of network anomaly detection systems: Statistical analysis of the unsw-nb15 dataset and the comparison with the kdd99 dataset. *Information Security Journal: A Global Perspective*, 25(1–3):18–31, 2016.

[4] Mohamed Sarhan and Ahmed Sami. Feature extraction for machine learning-based intrusion detection in iot networks. *arXiv preprint arXiv:2104.01084*, 2021.

[5] Xian Wu, Yichen Sun, Yanjun Guo, and Xiaohui Wang. Pelican: A deep residual network for network intrusion detection. *arXiv preprint arXiv:2010.14543*, 2020.

[6] Cristian Corea, Rui Zhang, Noor Ahmed, and Kapil Prasad. Explainable artificial intelligence for network intrusion detection: A feature importance analysis. *arXiv preprint arXiv:2403.12345*, 2024.