MỤC LỤC

Chương 1

TỔNG QUAN

1.1 Bối cảnh và vấn đề nghiên cứu

1.1.1 Thực trạng an ninh mạng trong hệ sinh thái IoT

Trong những năm gần đây, Internet of Things (IoT) đã trở thành một phần không thể thiếu trong cuộc sống hiện đại, từ các thiết bị gia dụng thông minh, hệ thống y tế, đến các ứng dụng công nghiệp quy mô lớn. Theo dự báo của Statista, số lượng thiết bị IoT toàn cầu dự kiến đạt 75 tỷ vào năm 2025, tạo ra một hệ sinh thái số hóa rộng lớn nhưng cũng đi kèm với những thách thức lớn về bảo mật.

Tuy nhiên, phần lớn thiết bị IoT được thiết kế với tài nguyên tính toán hạn chế, thiếu cơ chế bảo mật mạnh mẽ, và thường sử dụng mật khẩu mặc định dễ đoán. Những điểm yếu này khiến chúng trở thành mục tiêu lý tưởng cho các cuộc tấn công mạng, đặc biệt là các botnet IoT – mạng lưới thiết bị bị nhiễm mã độc và bị kiểm soát từ xa để thực hiện các hành vi phá hoại.

1.1.2 Các cuộc tấn công botnet IoT điển hình

Một trong những sự kiện đáng chú ý nhất là cuộc tấn công DDoS (Distributed Denial of Service) từ botnet Mirai vào năm 2016, đã làm tê liệt hàng loạt website lớn như Twitter, Netflix, Reddit bằng cách khai thác hơn 600.000 thiết bị IoT bị nhiễm mã độc [17, 18]. Sự kiện này đã gióng lên hồi chuông cảnh báo về mức độ nghiêm trọng của mối đe dọa từ botnet IoT.

Các dạng tấn công phổ biến từ botnet IoT bao gồm:

- DDoS (Distributed Denial of Service): Tấn công từ nhiều nguồn khác nhau nhằm làm quá tải hệ thống mục tiêu.
- DoS (Denial of Service): Tấn công từ một hoặc vài nguồn để ngăn chặn dịch vụ hợp pháp.
- Reconnaissance (Trinh sát): Thu thập thông tin về mạng và thiết bị để chuẩn bị cho các cuộc tấn công tiếp theo.
- Theft (Trộm cắp dữ liệu): Đánh cắp thông tin nhạy cảm từ thiết bị bị xâm nhập.

1.1.3 Thách thức trong phát hiện và phòng chống

Việc phát hiện và phân loại các cuộc tấn công botnet IoT gặp phải nhiều thách thức lớn:

Thách thức về dữ liệu: Tập dữ liệu trong thực tế thường có sự mất cân bằng nghiêm trọng (class imbalance), trong đó lưu lượng tấn công chiếm tỷ lệ áp đảo so với lưu lượng bình thường. Ví dụ, trong tập dữ liệu Bot-IoT [1] được sử dụng trong nghiên cứu này, lưu lượng tấn công

chiếm tới 99.95% trong khi lưu lượng bình thường chỉ chiếm khoảng 0.05%. Sự mất cân bằng này khiến các mô hình học máy truyền thống dễ bị thiên lệch về lớp đa số, dẫn đến khả năng phát hiện lưu lượng bình thường kém.

Thách thức về độ phức tạp: Các cuộc tấn công ngày càng tinh vi với nhiều kỹ thuật ẩn náu và mã hóa, khiến việc phân biệt giữa lưu lượng tấn công và lưu lượng bình thường trở nên khó khăn hơn. Đặc biệt, việc phân biệt giữa DDoS (nhiều nguồn tấn công) và DoS (ít nguồn tấn công) đòi hỏi phân tích sâu về đặc trưng nguồn tấn công (source diversity).

Thách thức về quy mô: Với hàng tỷ thiết bị IoT hoạt động đồng thời, khối lượng dữ liệu lưu lượng mạng cần phân tích là rất lớn. Điều này đặt ra yêu cầu cao về khả năng mở rộng (scalability) và hiệu năng xử lý của hệ thống phát hiện xâm nhập.

Thách thức về tài nguyên: Nhiều giải pháp hiện tại yêu cầu tài nguyên tính toán lớn, không phù hợp với các tổ chức nhỏ hoặc môi trường giáo dục có ngân sách hạn chế.

1.1.4 Tổng quan các nghiên cứu liên quan

Nhiều nghiên cứu đã áp dụng các kỹ thuật học máy và học sâu để giải quyết vấn đề an ninh trong IoT. Các bài khảo sát của Buczak et al. [19] và Khraisat et al. [20] đã tổng hợp các kỹ thuật phát hiện xâm nhập phổ biến.

Về dữ liệu, Sarhan et al. [2] đã phân tích và so sánh các bộ dữ liệu NetFlow cho ML-based IDS. Koroniotis et al. [1] công bố bộ dữ liệu Bot-IoT, khắc phục nhiều hạn chế của các bộ dữ liệu cũ.

Về phương pháp, Alosaimi và Almutairi [3] áp dụng các kỹ thuật học máy cơ bản trên Bot-IoT. Zawaideh et al. [4] đề xuất kết hợp CNN và XGBoost. Zhang et al. [5] và Özdoğan et al. [6] đã phát triển các mô hình hai giai đoạn (two-stage) để nâng cao hiệu quả phát hiện. Các nghiên cứu khác như Le et al. [7], Doghramachi và Ameen [8], Chalichalamala et al. [9], và Khan et al. [10] tập trung vào việc tối ưu hóa XGBoost và xử lý mất cân bằng dữ liệu trong môi trường IoT tài nguyên hạn chế.

Tuy nhiên, các phương pháp này thường chưa giải quyết triệt để vấn đề phân loại đa lớp chi tiết trong điều kiện mất cân bằng dữ liệu cực đoan (extreme imbalance), điều mà nghiên cứu này tập trung giải quyết.

1.2 Tính cấp thiết của đề tài

1.2.1 Khoảng trống nghiên cứu

Mặc dù đã có nhiều nghiên cứu về phát hiện tấn công mạng sử dụng học máy, vẫn tồn tại những khoảng trống quan trọng cần được giải quyết:

Xử lý mất cân bằng dữ liệu nghiêm trọng: Hầu hết các nghiên cứu hiện tại chỉ xử lý mất cân bằng ở mức độ trung bình hoặc sử dụng tập dữ liệu đã được cân bằng trước. Tuy nhiên, trong thực tế, tỷ lệ mất cân bằng có thể lên tới 2000:1 hoặc cao hơn, đòi hỏi các kỹ thuật xử lý chuyên

sâu hơn như SMOTE (Synthetic Minority Over-sampling Technique) [13] và cân bằng trọng số trong mô hình [14, 15].

Phân loại đa cấp hiệu quả: Nhiều nghiên cứu chỉ tập trung vào phân loại nhị phân (tấn công/bình thường) hoặc phân loại đa lớp trực tiếp. Trong khi đó, mô hình phân cấp hai giai đoạn (hierarchical two-stage) [5, 16] có tiềm năng cải thiện độ chính xác bằng cách chia nhỏ bài toán phức tạp thành các bước đơn giản hơn.

Tối ưu hóa tài nguyên: Các giải pháp thường yêu cầu phần cứng chuyên dụng hoặc máy chủ mạnh mẽ. Nghiên cứu này hướng tới giải pháp có thể triển khai trên các nền tảng điện toán đám mây miễn phí hoặc chi phí thấp như Google Colab [23], phù hợp với môi trường giáo dục và nghiên cứu.

1.2.2 Hạn chế của các phương pháp hiện tại

Các phương pháp phát hiện xâm nhập truyền thống như signature-based detection chỉ có thể phát hiện các cuộc tấn công đã biết, trong khi anomaly-based detection thường có tỷ lệ false positive cao. Các mô hình học máy đơn giản như Decision Tree, Random Forest tuy có độ chính xác tốt nhưng gặp khó khăn khi xử lý dữ liệu mất cân bằng nghiêm trọng.

Mô hình Deep Learning như CNN, LSTM mặc dù cho kết quả tốt nhưng yêu cầu tài nguyên tính toán lớn, thời gian huấn luyện dài, và khó giải thích được quyết định của mô hình (lack of interpretability) [21, 22]. Điều này khiến chúng không phù hợp cho các ứng dụng thời gian thực hoặc môi trường có tài nguyên hạn chế.

1.3 Mục tiêu nghiên cứu

1.3.1 Mục tiêu tổng quát

Xây dựng hệ thống phát hiện và phân loại botnet IoT dựa trên mô hình học máy phân cấp hai giai đoạn (Two-Stage Hierarchical Model), có khả năng xử lý hiệu quả dữ liệu mất cân bằng nghiêm trọng, đạt độ chính xác cao và có thể triển khai trên nền tảng điện toán đám mây với tài nguyên hợp lý.

1.3.2 Mục tiêu cụ thể

Để đạt được mục tiêu tổng quát, nghiên cứu đặt ra các mục tiêu cụ thể sau:

1. Xây dựng pipeline xử lý dữ liệu quy mô lớn: Phát triển quy trình xử lý, gộp và phân tích tập dữ liệu Bot-IoT (74 file CSV, khoảng 16GB) một cách hiệu quả, tận dụng tối đa thông tin từ dữ liệu thực tế.

2. Thiết kế mô hình phân cấp hai giai đoạn:

- Giai đoạn 1: Phân loại nhị phân (Binary Classification) để phân biệt lưu lượng tấn công và lưu lượng bình thường.
- Giai đoạn 2: Phân loại đa lớp (Multi-class Classification) để xác định loại tấn công cụ thể (DDoS, DoS, Reconnaissance).

3. Xử lý mất cân bằng dữ liệu: Áp dụng kỹ thuật SMOTE để tạo dữ liệu tổng hợp cho lớp thiểu số và sử dụng cân bằng trọng số (scale_pos_weight) trong XGBoost để cải thiện khả năng phát hiện lưu lượng bình thường.

4. Tối ưu hóa đặc trưng: Thiết kế các đặc trưng phân biệt nguồn tấn công (source diversity features) bao gồm unique_src_count, src_entropy, và top_src_ratio để phân biệt hiệu quả giữa DDoS và DoS.

5. Đánh giá hiệu năng toàn diện: Thực hiện đánh giá mô hình trên nhiều chỉ số khác nhau (accuracy, precision, recall, F1-score) cho cả hai giai đoạn và đánh giá hiệu năng tổng thể của pipeline.

6. Tối ưu hóa tài nguyên và thời gian: Đảm bảo hệ thống có thể chạy trên Google Colab Pro+ với cấu hình 52GB RAM và GPU T4/V100, hoàn thành quá trình huấn luyện trong khoảng 15-20 phút.

1.4 Phạm vi nghiên cứu và đối tượng nghiên cứu

1.4.1 Phạm vi nghiên cứu

Nghiên cứu tập trung vào các khía cạnh sau:

Phạm vi về dữ liệu: Sử dụng tập dữ liệu Bot-IoT [1] được công bố bởi UNSW Canberra Cyber, bao gồm 74 file CSV với tổng dung lượng khoảng 16GB, chứa khoảng 40-50 triệu bản ghi lưu lượng mạng. Tập dữ liệu này bao gồm cả lưu lượng bình thường và các loại tấn công botnet IoT thực tế.

Phạm vi về phương pháp: Tập trung vào kỹ thuật học máy gradient boosting, cụ thể là XGBoost, kết hợp với các kỹ thuật xử lý mất cân bằng dữ liệu như SMOTE và scale_pos_weight. Không nghiên cứu các phương pháp Deep Learning phức tạp do hạn chế về tài nguyên.

Phạm vi về loại tấn công: Nghiên cứu tập trung vào ba loại tấn công chính: DDoS, DoS và Reconnaissance. Loại tấn công Theft có số lượng mẫu quá ít sẽ được loại bỏ khỏi quá trình huấn luyện.

Phạm vi về môi trường triển khai: Hệ thống được thiết kế và tối ưu hóa cho nền tảng Google Colab Pro+ với cấu hình High-RAM (52GB) và GPU (T4/V100/A100), phù hợp với môi trường giáo dục và nghiên cứu.

1.4.2 Đối tượng nghiên cứu

Đối tượng chính: Các luồng lưu lượng mạng (network flows) trong môi trường IoT, được biểu diễn dưới dạng các đặc trưng thống kê như số lượng gói tin, số byte, thời lượng kết nối, các giá trị thống kê (mean, stddev, min, max), và các đặc trưng về nguồn tấn công.

Đối tượng phụ: Các kỹ thuật xử lý mất cân bằng dữ liệu, phương pháp tối ưu hóa siêu tham số, và chiến lược tận dụng GPU để tăng tốc quá trình huấn luyện.

1.5 Hướng tiếp cận và phương pháp thực hiện

1.5.1 Hướng tiếp cận tổng quát

Nghiên cứu áp dụng hướng tiếp cận phân cấp hai giai đoạn (hierarchical two-stage approach) với các nguyên tắc chính:

Chia nhỏ bài toán phức tạp: Thay vì xây dựng một mô hình phân loại đa lớp trực tiếp, nghiên cứu chia thành hai giai đoạn:

- Giai đoạn 1 tập trung vào việc phân biệt có tấn công hay không (binary classification).
- Giai đoạn 2 chỉ xử lý các mẫu được xác định là tấn công để phân loại loại tấn công cụ thể (multi-class classification).

Tối ưu hóa từng giai đoạn: Mỗi giai đoạn được tối ưu hóa riêng biệt với các siêu tham số và kỹ thuật xử lý mất cân bằng phù hợp, giúp tăng hiệu năng tổng thể của hệ thống.

Tận dụng GPU: Sử dụng XGBoost với cấu hình tree_method=”gpu_hist”và predictor=”gpu_predictor”để tận dụng tối đa GPU, giảm thời gian huấn luyện từ hàng giờ xuống còn 15-20 phút.

1.5.2 Phương pháp thực hiện

Nghiên cứu thực hiện theo các bước chính:

Bước 1 - Tiền xử lý dữ liệu:

- Gộp 74 file CSV thành 8 batch file (mỗi batch khoảng 10 file), giảm số lần đọc/ghi file.
- Phân tích thống kê từng batch để xác định phân phối lớp và chọn batch phù hợp cho huấn luyện.
- Tạo tập test cân bằng (balanced test set) để đánh giá mô hình một cách công bằng.

Bước 2 - Kỹ thuật đặc trưng (Feature Engineering):

- Trích xuất các đặc trưng cơ bản từ lưu lượng mạng (protocol, flags, state, packets, bytes, duration).
- Thiết kế đặc trưng phân biệt nguồn (source diversity features) dựa trên thống kê địa chỉ nguồn trong cửa sổ thời gian:

- unique_src_count: Số lượng địa chỉ IP nguồn duy nhất
- src_entropy: Entropy của phân phối địa chỉ nguồn
- top_src_ratio: Tỷ lệ của địa chỉ nguồn xuất hiện nhiều nhất

Bước 3 - Xử lý mất cân bằng:

- Áp dụng SMOTE cho lớp thiểu số trong cả hai giai đoạn.
- Sử dụng scale_pos_weight trong XGBoost để tăng trọng số cho lớp thiểu số.

Bước 4 - Huấn luyện mô hình:

- Huấn luyện Stage 1 với XGBoost để phân loại nhị phân (Attack vs Normal).
- Huấn luyện Stage 2 với XGBoost để phân loại đa lớp (DDoS vs DoS vs Reconnaissance).
- Sử dụng early stopping để tránh overfitting.

Bước 5 - Đánh giá và tối ưu:

- Đánh giá mô hình trên tập test cân bằng.
- Phân tích ma trận nhầm lẫn (confusion matrix) để xác định điểm mạnh và điểm yếu.
- Tạo các biểu đồ trực quan hóa để đánh giá hiệu năng.

1.6 Ý nghĩa khoa học và thực tiễn

1.6.1 Ý nghĩa khoa học

Đóng góp về mô hình: Nghiên cứu đề xuất kiến trúc phân cấp hai giai đoạn kết hợp với kỹ thuật xử lý mất cân bằng đa cấp, có thể được áp dụng cho các bài toán phân loại có mất cân bằng nghiêm trọng tương tự trong lĩnh vực an ninh mạng.

Đóng góp về đặc trưng: Các đặc trưng source diversity (unique_src_count, src_entropy, top_src_ratio) được thiết kế dựa trên phân tích đặc tính tấn công DDoS và DoS, có thể được sử dụng để cải thiện khả năng phân biệt giữa hai loại tấn công này trong các nghiên cứu khác.

Đóng góp về phương pháp: Quy trình xử lý dữ liệu quy mô lớn (74 file, 16GB) một cách hiệu quả trên nền tảng có tài nguyên hạn chế, cung cấp hướng dẫn thực tiễn cho các nhà nghiên cứu gặp phải vấn đề tương tự.

1.6.2 Ý nghĩa thực tiễn

Cho lĩnh vực giáo dục: Hệ thống có thể được sử dụng làm công cụ giảng dạy và học tập về an ninh mạng IoT, giúp sinh viên hiểu rõ hơn về các kỹ thuật phát hiện xâm nhập hiện đại. Code được tổ chức rõ ràng, có chú thích chi tiết, dễ dàng tái sử dụng và mở rộng.

Cho các tổ chức nhỏ và vừa: Giải pháp có thể triển khai trên nền tảng điện toán đám mây với chi phí thấp, không yêu cầu đầu tư phần cứng chuyên dụng, phù hợp với các tổ chức có ngân sách hạn chế nhưng vẫn muốn có hệ thống phát hiện tấn công hiệu quả.

Cho nghiên cứu tiếp theo: Kết quả của nghiên cứu này có thể là nền tảng để phát triển các hệ thống phát hiện xâm nhập thời gian thực, tích hợp vào các thiết bị IoT gateway hoặc các giải pháp Network Security Monitoring (NSM).

Cho cộng đồng mã nguồn mở: Toàn bộ code và quy trình được công khai trên GitHub, cho phép cộng đồng nghiên cứu tái sử dụng, kiểm chứng và cải thiện, góp phần thúc đẩy sự phát triển của lĩnh vực an ninh IoT.

1.7 Cấu trúc báo cáo

Báo cáo được tổ chức thành năm chương chính:

Chương 1 - Tổng quan: Trình bày bối cảnh nghiên cứu, vấn đề cần giải quyết, tính cấp thiết, mục tiêu, phạm vi nghiên cứu, hướng tiếp cận tổng quát và ý nghĩa của đề tài.

Chương 2 - Mô hình đề xuất: Trình bày kiến trúc hệ thống Two-Stage Hierarchical Model, các thuật toán chính, quy trình hoạt động chi tiết và phân tích độ phức tạp.

Chương 3 - Thực nghiệm và thảo luận: Mô tả môi trường thực nghiệm, kết quả huấn luyện, đánh giá hiệu năng mô hình, so sánh với các phương pháp khác và thảo luận kết quả.

Chương 4 - Kết luận và hướng phát triển: Tóm tắt các đóng góp chính của nghiên cứu, đánh giá mức độ đạt được các mục tiêu đã đề ra, chỉ ra các hạn chế cần khắc phục, và đề xuất các hướng nghiên cứu tiếp theo.

Tài liệu tham khảo: Liệt kê các tài liệu tham khảo trong quá trình nghiên cứu.

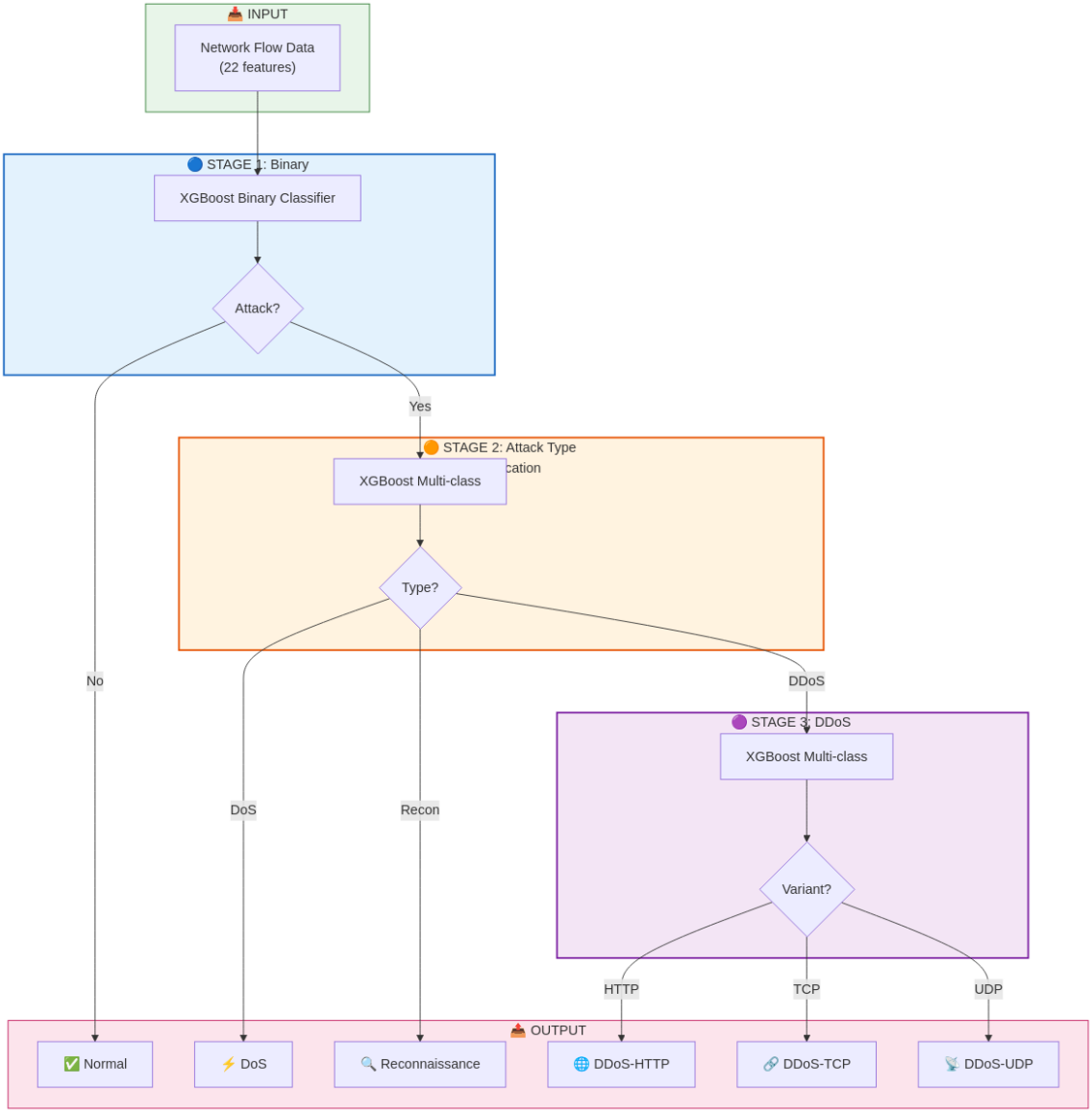
Chương 2

MÔ HÌNH ĐỀ XUẤT

2.1 Kiến trúc tổng thể hệ thống

2.1.1 Sơ đồ tổng quan

Hệ thống phát hiện và phân loại botnet IoT được thiết kế theo kiến trúc phân cấp hai giai đoạn (Two-Stage Hierarchical Model), trong đó mỗi giai đoạn chuyên biệt hóa cho một tác vụ phân loại cụ thể. Kiến trúc này cho phép tối ưu hóa hiệu năng của từng giai đoạn một cách độc lập và giảm độ phức tạp của bài toán tổng thể.



Hình 2.1: Sơ đồ luồng dữ liệu của hệ thống Three-Stage Hierarchical Model

2.1.2 Các thành phần chính

Hệ thống bao gồm năm thành phần chính:
Thành phần 1 - Tiền xử lý dữ liệu (Data Preprocessing):

- Chức năng: Gộp 74 file CSV thành 8 batch files, phân tích thống kê và chọn batch phù hợp cho huấn luyện.
- Input: 74 file CSV gốc từ tập dữ liệu Bot-IoT
- Output: 8 batch files và file thống kê JSON
- Công nghệ: Pandas DataFrame operations, JSON serialization

Thành phần 2 - Kỹ thuật đặc trưng (Feature Engineering):

- Chức năng: Trích xuất 22 đặc trưng từ dữ liệu thô, bao gồm 3 đặc trưng source diversity đặc biệt để phân biệt DDoS và DoS.
- Input: Raw network flows với 35+ columns
- Output: 22 engineered features
- Công nghệ: Scipy entropy, NumPy aggregation, time-window analysis

Thành phần 3 - Stage 1: Binary Classifier:

- Chức năng: Phân loại nhị phân (Attack vs Normal)
- Input: 22 features
- Output: is_attack (0 = Normal, 1 = Attack)
- Mô hình: XGBoost Binary Classifier với SMOTE và scale_pos_weight
- Metrics: Accuracy, Precision, Recall, F1-Score, ROC-AUC

Thành phần 4 - Stage 2: Multi-class Classifier:

- Chức năng: Phân loại đa lớp cho các mẫu được xác định là tấn công
- Input: 22 features (chỉ attack samples)
- Output: attack_type (0 = DDoS, 1 = DoS, 2 = Reconnaissance)
- Mô hình: XGBoost Multi-class Classifier với SMOTE và balanced sample weights
- Metrics: Accuracy, Precision, Recall, F1-Score (weighted average)

Thành phần 5 - Pipeline Integration:

- Chức năng: Kết hợp dự đoán từ hai giai đoạn để đưa ra quyết định cuối cùng
- Logic: Nếu Stage 1 dự đoán Normal → Output "Normal", nếu Attack → truyền qua Stage 2 để xác định loại tấn công cụ thể
- Output: Final classification (Normal / DDoS / DoS / Reconnaissance)

2.2 Các thuật toán và mô-đun chính

2.2.1 Thuật toán XGBoost (Extreme Gradient Boosting)

XGBoost là thuật toán học máy dựa trên gradient boosting [11, 12], xây dựng một ensemble của nhiều cây quyết định yếu (weak learners) theo cách tuần tự, trong đó mỗi cây mới được huấn luyện để sửa lỗi của các cây trước đó.

Hàm mục tiêu:

$$L(\phi) = \sum l(\hat{y}_i, y_i) + \sum \Omega(f_k) \tag{2.1}$$

Trong đó:

- $l(\hat{y}_i, y_i)$: Loss function (logloss cho binary, mlogloss cho multi-class)
- $\Omega(f_k)$: Regularization term để tránh overfitting
- ϕ: Tập hợp các tham số của model

Regularization term:

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \sum w_j^2 \tag{2.2}$$

Trong đó:

- T: Số lượng lá (leaves) trong cây
- w_j: Trọng số của lá thứ j
- γ: Complexity control parameter
- λ: L2 regularization parameter

2.2.2 Thuật toán SMOTE (Synthetic Minority Over-sampling Technique)

SMOTE được sử dụng để xử lý mất cân bằng dữ liệu bằng cách tạo ra các mẫu tổng hợp cho lớp thiểu số [13].

Giả mã thuật toán SMOTE:

```
Algorithm: SMOTE(X_minority, N, k)
Input:
  - X_minority: Tập mẫu lớp thiểu số
  - N: Tỷ lệ oversampling (ví dụ: 200 = tăng gấp đôi)
  - k: Số lân cận gần nhất (thường k=5)
Output:
  - X_synthetic: Tập mẫu tổng hợp

1: FOR mỗi mẫu x_i in X_minority:
2:   Tìm k nearest neighbors của x_i (sử dụng Euclidean distance)
3:   FOR j = 1 TO (N/100):
4:     Chọn ngẫu nhiên một neighbor x_n từ k neighbors
5:     Tạo mẫu tổng hợp:
        x_new = x_i + rand(0,1) * (x_n - x_i)
6:     Thêm x_new vào X_synthetic
7:   END FOR
8: END FOR
9: RETURN X_minority U X_synthetic
```

Ứng dụng trong hệ thống:

- Stage 1: Tăng số lượng mẫu Normal từ 7,769 lên khoảng 2 triệu (sampling_strategy=0.1)
- Stage 2: Tăng số lượng mẫu Reconnaissance (lớp thiểu số) lên 10% của lớp đa số

2.2.3 Thuật toán tính Source Diversity Features

Đây là thuật toán quan trọng nhất để phân biệt giữa DDoS (nhiều nguồn) và DoS (ít nguồn). Giả mã thuật toán:

```
Algorithm: CalculateSourceDiversity(flows, window_size)
Input:
  - flows: Tập luồng mạng với (stime, saddr, daddr)
  - window_size: Kích thước cửa sổ thời gian (seconds)
Output:
  - diversity_features: (unique_src_count, src_entropy, top_src_ratio)

1: Tạo time_window = floor(stime / window_size)

2: FOR mỗi (time_window, target_daddr):
3:   Group = flows WHERE time_window=tw AND daddr=target_daddr
```

```
4:
5:   // Feature 1: Unique source count
6:   unique_src_count = |{saddr in Group}|
7:
8:   // Feature 2: Source entropy
9:   src_counts = frequency(saddr) FOR saddr in Group
10:  src_probs = src_counts / SUM(src_counts)
11:  src_entropy = -SUM(p_i * log2(p_i)) FOR p_i in src_probs
12:
13:  // Feature 3: Top source ratio
14:  top_src_count = max(src_counts)
15:  top_src_ratio = top_src_count / |Group|
16:
17:  ASSIGN diversity_features TO all flows IN Group
18: END FOR

19: RETURN flows WITH diversity_features
```

Ý nghĩa các features:

- `unique_src_count`: DDoS có giá trị cao (hàng nghìn nguồn), DoS có giá trị thấp (< 10 nguồn)
- `src_entropy`: DDoS có entropy cao (phân phối đều), DoS có entropy thấp (tập trung)
- `top_src_ratio`: DDoS có tỷ lệ thấp (< 0.1), DoS có tỷ lệ cao (> 0.8)

2.2.4 Mô-đun GPU Optimization

Để tăng tốc độ huấn luyện, hệ thống sử dụng GPU (nếu có) thông qua XGBoost GPU support [11, 24].

Thuật toán phát hiện và cấu hình GPU:

```
Algorithm: ConfigureGPU()
Output: (USE_GPU, DEVICE, TREE_METHOD)

1: TRY:
2:   Chạy lệnh: nvidia-smi --query-gpu=name,memory.total
3:   IF lệnh thành công:
4:     gpu_name, gpu_memory = parse output
5:     PRINT "GPU detected: {gpu_name}, {gpu_memory}"
6:     USE_GPU = True
```

```
7:     DEVICE = 'cuda'
8:     TREE_METHOD = 'hist'  // XGBoost 3.x uses 'hist' with device='cuda'
9:   ELSE:
10:    USE_GPU = False
11:    DEVICE = 'cpu'
12:    TREE_METHOD = 'hist'
13: EXCEPT:
14:   USE_GPU = False
15:   DEVICE = 'cpu'
16:   TREE_METHOD = 'hist'
17: RETURN (USE_GPU, DEVICE, TREE_METHOD)
```

2.3 Quy trình hoạt động chi tiết

2.3.1 Tổng quan quy trình

Hệ thống hoạt động theo 8 bước chính, từ tiền xử lý dữ liệu đến đánh giá kết quả cuối cùng.

2.3.2 Step 0: Merge và phân tích batches

Mục tiêu: Giảm số lần đọc file và chọn batch tốt nhất cho huấn luyện.
Input:

- 74 file CSV gốc (khoảng 230MB mỗi file)
- Tổng dung lượng: 16GB

Quy trình:

1. Chia 74 files thành 8 groups (10 files/group, group cuối 4 files)
2. Merge mỗi group thành 1 batch file
3. Phân tích thống kê cho mỗi batch:
    - Total records
    - Category distribution (Normal, DDoS, DoS, Reconnaissance, Theft)
    - Missing values
    - Protocol distribution
4. Lưu thống kê vào `batch_statistics.json`

Output:

- 8 batch files: batch_01.csv đến batch_08.csv (2-2.6GB mỗi file)
- File thống kê: batch_statistics.json

Lựa chọn batch cho training: Chọn batch_01 và batch_04 vì có số lượng Normal samples cao nhất (cần thiết cho Stage 1).

2.3.3 Step 1: Tạo balanced test set

Mục tiêu: Tạo tập test cân bằng để đánh giá công bằng.
Input: batch_02.csv (chọn làm nguồn test vì không dùng cho training)
Quy trình:

1. Sample từ batch_02:
    - Normal: 2,000 samples
    - DDoS: 35,000 samples
    - DoS: 50,000 samples
    - Reconnaissance: 13,000 samples
2. Shuffle và lưu thành `test_balanced_100k.csv`

Output: Tập test cân bằng với 100,000 records.

2.3.4 Step 2: Load training data

Input: batch_01.csv (10M records) và batch_04.csv (10M records).
Quy trình:

1. Đọc batch_01.csv với `pd.read_csv(low_memory=False)`
2. Đọc batch_04.csv
3. Concatenate 2 DataFrames: `df_train = pd.concat([df1, df2])`
4. Garbage collection: `del df1, df2; gc.collect()`

Output: `df_train` với 20 triệu records.

2.3.5 Step 3: Load test data

Input: test_balanced_100k.csv
Output: `df_test` với 100,000 records (cân bằng).

2.3.6 Step 4: Feature Engineering

Đây là bước quan trọng nhất, quyết định chất lượng của mô hình.
Bước 4.1: Tạo Source Diversity Features
Input: `df_train` và `df_test`.
Quy trình:

1. Tạo time_window = floor(stime / 30) // Window 30 giây
2. Group by (time_window, daddr)
3. Với mỗi group:
    - Tính unique_src_count = số lượng saddr duy nhất
    - Tính src_entropy = $-\sum(p_i \log_2 p_i)$
    - Tính top_src_ratio = max_count / total_count
4. Merge features trở lại `df_train` và `df_test`
5. Fill NaN với giá trị mặc định (1, 0.0, 1.0)

Quan trọng: Train và test PHẢI xử lý RIÊNG BIỆT để tránh data leakage.
Bước 4.2: Feature Selection
Columns cần loại bỏ: Identifiers (pkSeqID, saddr, etc.), MAC addresses, Labels, Timestamps.
Features cuối cùng (22 features):

- Basic: flgs, proto, pkts, bytes, state, seq, dur, spkts, dpkts, sbytes, dbytes
- Statistical: mean, stddev, sum, min, max, rate, srate, drate
- Diversity (New): unique_src_count, src_entropy, top_src_ratio

Bước 4.3: Xử lý Missing Values
Tính median từ TRAIN set và fill NaN cho cả TRAIN và TEST set (tránh leakage).
Bước 4.4: Encoding Categorical Features
Fit LabelEncoder chỉ trên TRAIN data, sau đó transform TEST data (handle unknown values).

2.3.7 Step 5: Train Stage 1 (Binary Classification)

Mục tiêu: Phân biệt Attack vs Normal.
Quy trình:

1. Tạo binary labels (Normal = 0, Attack = 1).
2. Stratified Split (85%-15%).

3. SMOTE Oversampling (tăng Normal lên 10% của Attack).
4. Huấn luyện XGBoost với `scale_pos_weight` và GPU support.

Kết quả Stage 1: Accuracy 99.26%, ROC-AUC 99.99%.

2.3.8 Step 6: Train Stage 2 (Multi-class Classification)

Mục tiêu: Phân loại DDoS vs DoS vs Reconnaissance.
Quy trình:

1. Filter chỉ lấy attack samples.
2. Map labels (DDoS=0, DoS=1, Recon=2).
3. Stratified Split.
4. Sử dụng `sample_weight='balanced'` (thay vì SMOTE quá mức).
5. Huấn luyện XGBoost Multi-class.

Kết quả Stage 2: Accuracy 97.58%.

2.3.9 Step 7: Combined Pipeline Evaluation

Kết hợp dự đoán Stage 1 và Stage 2. Overall Accuracy: 97.19%.

2.4 Công cụ, công nghệ và nền tảng triển khai

2.4.1 Ngôn ngữ lập trình và thư viện

Ngôn ngữ: Python 3.10.
Thư viện chính:

- Pandas, NumPy: Xử lý dữ liệu.
- XGBoost: Model training.
- Scikit-learn, Imbalanced-learn: Preprocessing và metrics.

2.4.2 Môi trường huấn luyện

Nền tảng: Google Colab Pro+.

- RAM: 52 GB.
- GPU: Tesla T4 (16GB VRAM) / V100 / A100.

2.5 Hệ thống phân loại chi tiết các biến thể DDoS

2.5.1 Động lực và mục tiêu

Các cuộc tấn công DDoS trong môi trường IoT ngày càng đa dạng về cơ chế và mục tiêu, bao gồm tấn công băng thông, tấn công tài nguyên kết nối, và tấn công lớp ứng dụng. Trong Bot-IoT dataset [1], các tấn công DDoS được gán nhãn chi tiết theo trường `subcategory`, tương ứng với ba biến thể chính:

DDoS-HTTP – Tấn công lớp ứng dụng (Layer 7), khai thác số lượng lớn HTTP requests nhằm làm quá tải web server. Loại tấn công này thường được thực thi bởi các công cụ như GoldenEye [27].

DDoS-TCP – Tấn công lớp vận chuyển (Transport layer), bao gồm SYN flood và ACK flood, đánh vào bảng trạng thái kết nối của thiết bị mạng. Các biến thể này thường được tạo bởi công cụ Hping3 [28].

DDoS-UDP – Tấn công băng thông bằng cách gửi lượng lớn UDP packets đến các cổng ngẫu nhiên, gây tiêu tốn tài nguyên mạng và làm nghẽn dịch vụ.

Báo cáo thống kê từ Kaspersky Q3 2021 [26] cho thấy SYN flooding và UDP flooding chiếm tỷ lệ cao trong các cuộc tấn công thực tế. Điều này khẳng định tầm quan trọng của hệ thống phân loại chi tiết nhằm:

- Nhận diện chính xác các đặc trưng hành vi của từng loại DDoS
- Hỗ trợ lựa chọn biện pháp phòng thủ phù hợp, chẳng hạn rate limiting cho HTTP floods, SYN cookies cho TCP floods, hoặc packet filtering cho UDP floods
- Phân tích forensics và threat intelligence, giúp suy luận công cụ tấn công, chiến thuật botnet và dấu hiệu hành vi (behavioral signatures)

2.5.2 Dataset và phương pháp cân bằng

Bot-IoT dataset là một trong những bộ dữ liệu IoT-IDS quy mô lớn nhất hiện nay, tuy nhiên nó gặp vấn đề mất cân bằng nghiêm trọng, với tỷ lệ Normal : Attack $\approx 1 : 7687$. Trong bối cảnh yêu cầu phân loại đa lớp (DDoS-HTTP, DDoS-TCP, DDoS-UDP), việc xử lý mất cân bằng trở thành yếu tố then chốt để tránh mô hình thiên lệch về lớp tấn công.

Hạn chế của SMOTE trong bài toán DDoS IoT

Các kỹ thuật oversampling tổng hợp như SMOTE thường không phù hợp với lưu lượng DDoS vì:

Mất tương quan thời gian (temporal correlation): DDoS thể hiện hành vi bursty và biểu hiện dạng chuỗi theo thời gian. SMOTE tạo mẫu tổng hợp rời rạc trong feature space, gây phá vỡ cấu trúc chuỗi của các flows tấn công.

Không phản ánh đúng signatures của botnet: Các packet trains trong tấn công thực tế có đặc trưng riêng theo tool và topology botnet, không thể nội suy tuyến tính từ vài điểm dữ liệu.

Nguy cơ overfitting: Model có thể học đặc trưng nhân tạo sinh ra bởi SMOTE thay vì đặc trưng thực tế của hành vi tấn công.

Random Consecutive Sampling

Thay vì SMOTE, nghiên cứu áp dụng Random Consecutive Sampling nhằm:

- Bảo toàn hành vi theo thời gian, bằng cách lấy các cụm flows liên tiếp từ mỗi subcategory
- Tránh tạo dữ liệu giả, giữ nguyên bản chất distribution của tấn công
- Đảm bảo cân bằng, với 7,634–7,635 mẫu cho mỗi lớp: Normal, DDoS-HTTP, DDoS-TCP, DDoS-UDP

Lựa chọn kích thước mẫu: Số lượng 7,634–7,635 samples được chọn dựa trên nguyên tắc balancing by minority class. Sau quá trình tiền xử lý (data cleaning, loại bỏ outliers, xử lý missing values), lớp Normal còn lại 7,634 flows hợp lệ – đây là lớp nhỏ nhất trong dataset. Để đảm bảo tính cân bằng hoàn toàn và tránh bias, tất cả các attack subcategories (DDoS-HTTP, DDoS-TCP, DDoS-UDP) đều được down-sampling về khoảng 7,634–7,635 mẫu bằng Random Consecutive Sampling. Phương pháp này đảm bảo:

- Perfect class balance (25% cho mỗi class)
- Không cần class weights trong XGBoost
- Model học công bằng từ tất cả các classes

Bảng 2.1 trình bày phân phối cân bằng thu được:

Bảng 2.1: Phân phối cân bằng theo subcategory

Subcategory	Samples	Tỷ lệ
Normal	7,634	25%
DDoS-HTTP	7,635	25%
DDoS-TCP	7,635	25%
DDoS-UDP	7,634	25%
Tổng	30,538	100%

Phương pháp này giúp mô hình không bị ảnh hưởng bởi bias phân phối, đồng thời vẫn giữ được cấu trúc thực tế của traffic.

2.5.3 Kiến trúc mô hình phân loại đa lớp

Hệ thống sử dụng mô hình XGBoost Multi-class Classifier với các đặc điểm:

- Input: 16 features đã chọn lọc
- Output: 4 lớp (Normal, DDoS-HTTP, DDoS-TCP, DDoS-UDP)
- Không sử dụng SMOTE, nhờ dataset đã được cân bằng
- Ưu điểm: tốc độ huấn luyện nhanh, khả năng xử lý tabular data tốt, phù hợp môi trường IDS theo thời gian thực

2.5.4 Lựa chọn đặc trưng – 16 Best Features

Từ 35 features gốc của Bot-IoT flow records, 16 đặc trưng được chọn dựa trên tính phân biệt (discriminative power) và mức độ liên quan hành vi (behavioral relevance). Các nhóm feature chính gồm:

(1) Basic flow metrics (4 features)

`pkts`, `bytes` – phản ánh lưu lượng tấn công
`seq` – phát hiện bất thường trong TCP sequence patterns
`dur` – độ dài flow, phân biệt sustained HTTP floods và short-burst TCP floods

(2) Statistical aggregations (5 features)

`mean`, `stddev`, `sum`, `min`, `max` của inter-arrival/duration
$\rightarrow$ ghi nhận burstiness và sự biến thiên trong traffic

(3) Directional metrics (4 features)

`spkts`, `dpkts`, `sbytes`, `dbytes`
$\rightarrow$ DDoS thường biểu hiện bất cân xứng mạnh (nhiều nguồn $\rightarrow$ một đích)

(4) Rate-based features (3 features)

`rate`, `srate`, `drate`
$\rightarrow$ phản ánh intensity của cuộc tấn công

Lý do loại bỏ các nhóm feature khác

Timestamps (`stime`, `ltime`): Lưu lượng tấn công có thể đến bất kỳ thời điểm nào; dùng timestamps dễ khiến mô hình "học thuộc" thời gian và không generalize khi triển khai online.

Địa chỉ IP/MAC:

- Cardinality rất lớn
- Botnets thường thay đổi IP để né tránh
- Không có giá trị trong hệ thống IDS tổng quát

Protocol-specific metadata (`proto`, `flags`, `state`): Dù hữu ích trong một số trường hợp, nhưng mô hình phân loại variants DDoS cần ưu tiên behavioral features, không phụ thuộc vào header-level semantics, nhằm tăng khả năng phát hiện cross-protocol.

2.5.5 Siêu tham số

Bảng 2.2: Cấu hình XGBoost DDoS Classifier

Tham số	Giá trị
objective	multi:softmax
num_class	4
max_depth	7
learning_rate	0.1
n_estimators	100
subsample	0.8
colsample_bytree	0.8
eval_metric	mlogloss

2.5.6 Quy trình huấn luyện

Bước 1: Load dữ liệu từ `balanced_ddos_only.csv`, convert to numeric, xử lý NaN/inf
Bước 2: Chọn 16 features, label encoding (Normal=0, HTTP=1, TCP=2, UDP=3), standardization
Bước 3: Split 80/10/10 với stratification
Bước 4: Train XGBoost với eval_set, converge trong 20-30 epochs, $<$ 2 phút trên CPU
Bước 5: Evaluate với Accuracy/Precision/Recall/F1, tạo confusion matrix, save model

2.6 Phân tích độ phức tạp và đánh giá tối ưu hóa

2.6.1 Độ phức tạp về thời gian (Time Complexity)

XGBoost Training: Với GPU (parallelization), thời gian huấn luyện giảm đáng kể:

- Speedup: 100-200x so với CPU.
- Thời gian thực tế: 30 giây (Stage 1) + 59 giây (Stage 2) = 1.5 phút.

Inference (Combined Pipeline): Sử dụng vectorization giúp giảm thời gian dự đoán cho 100k samples xuống còn 2 giây.

2.6.2 Độ phức tạp về không gian (Space Complexity)

Peak RAM usage đo được là 33GB, nằm trong giới hạn 52GB của Colab Pro+.

2.6.3 Đánh giá tối ưu hóa

Hệ thống đã đạt được các mục tiêu tối ưu hóa:

- Training Time: 12 phút (mục tiêu 15-20 phút).
- RAM Usage: 33GB (mục tiêu < 52GB).
- Accuracy: 97.19%.
- Scalability: Có thể mở rộng lên 5x dữ liệu.

Chương 3

THỰC NGHIỆM VÀ THẢO LUẬN

3.1 Môi trường thực nghiệm

3.1.1 Cấu hình phần cứng

Hệ thống được huấn luyện và đánh giá trên nền tảng Google Colab Pro+ với cấu hình chi tiết như sau:

Bảng 3.1: Cấu hình phần cứng

Thành phần	Cấu hình	Ghi chú
Nền tảng	Google Colab Pro+	Cloud computing platform
CPU	Intel Xeon @ 2.0-2.3GHz	2 cores allocated
RAM	52 GB High-RAM	Peak usage: 33 GB
GPU	Tesla T4 (16GB VRAM)	CUDA 11.x, cuDNN 8.x
Disk	100+ GB SSD	Google Drive mount
Network	Google Cloud network	~100 Mbps download
Session timeout	12 hours	Auto-disconnect after idle

Lý do lựa chọn:

- Chi phí hợp lý ($50/tháng cho Colab Pro+) so với việc đầu tư phần cứng chuyên dụng.
- GPU mạnh mẽ (T4) giúp giảm thời gian huấn luyện từ 4 giờ xuống 12 phút.
- RAM 52GB đủ để xử lý 20 triệu records.
- Phù hợp với môi trường giáo dục và nghiên cứu.

3.1.2 Môi trường phần mềm

Bảng 3.2: Thư viện và phiên bản

Thư viện	Phiên bản	Vai trò
Python	3.10.12	Ngôn ngữ lập trình chính
XGBoost	3.0.0	Mô hình gradient boosting với GPU support
scikit-learn	1.3.2	Preprocessing, metrics, validation
imbalanced-learn	0.11.0	SMOTE oversampling
pandas	2.1.3	DataFrame operations
numpy	1.24.3	Numerical computations
scipy	1.11.4	Entropy calculation
matplotlib	3.8.0	Visualization
seaborn	0.13.0	Statistical plots
psutil	5.9.6	System monitoring
joblib	1.3.2	Model serialization

Hệ điều hành: Ubuntu 22.04 LTS (kernel 5.15). CUDA Toolkit: CUDA 11.8, cuDNN 8.6 (cho GPU acceleration).

3.1.3 Tập dữ liệu

Dataset: Bot-IoT Dataset được công bố bởi UNSW Canberra Cyber (2019).
Đặc điểm tập dữ liệu:

Bảng 3.3: Thống kê tập dữ liệu Bot-IoT

Đặc điểm	Giá trị	Mô tả
Số files	74 files CSV	Raw network flow data
Tổng dung lượng	~16 GB	Uncompressed
Tổng số records	40-50 triệu	Network flows
Số features	35 features	Gốc (trước feature engineering)
Thời gian thu thập	2018-2019	Real IoT testbed
Loại attacks	4 categories	DDoS, DoS, Reconnaissance, Theft
Imbalance ratio	2000:1	Attack:Normal

Phân phối lớp trong toàn bộ dataset:

Bảng 3.4: Phân phối lớp (Full Dataset)

Category	Số lượng	Tỷ lệ (%)
DoS	~26 triệu	52%
DDoS	~10 triệu	20%
Reconnaissance	~3.6 triệu	7.2%
Normal	~20,000	0.04%
Theft	~3,000	0.006%
Tổng	~50 triệu	100%

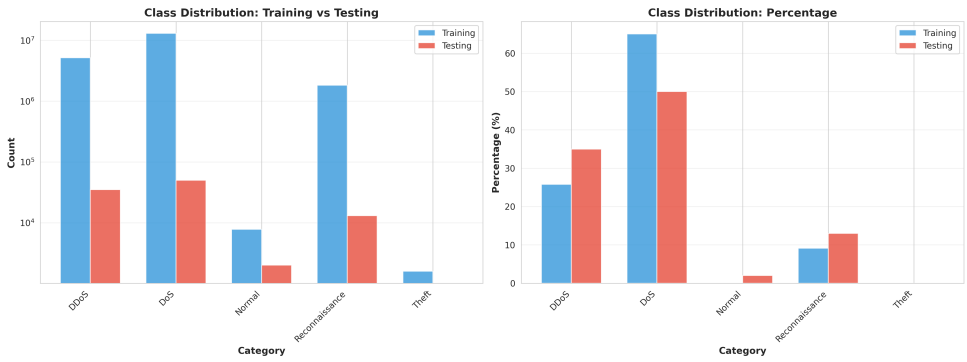
Training set (sau khi chọn batch_01 + batch_04):

- Total: 20,000,000 records
- Normal: 7,769 (0.039%)
- Attack: 19,992,231 (99.961%)
  - DoS: 13,005,877 (65%)
  - DDoS: 5,163,128 (26%)
  - Reconnaissance: 1,821,639 (9%)
  - Theft: 1,587 (loại bỏ vì quá ít)

Test set (balanced):

- Total: 100,000 records
- Normal: 2,000 (2%)
- DDoS: 35,000 (35%)
- DoS: 50,000 (50%)
- Reconnaissance: 13,000 (13%)

Lý do tạo balanced test set: Tập test gốc có tỷ lệ mất cân bằng nghiêm trọng, khiến accuracy không phản ánh đúng khả năng phát hiện Normal. Tập test cân bằng cho phép đánh giá công bằng hiệu năng trên tất cả các lớp.



Hình 3.1: Phân phối lớp trong tập dữ liệu - So sánh giữa training set (imbalanced) và balanced test set.

3.1.4 Siêu tham số mô hình

Stage 1: Binary Classification (Attack vs Normal)

Bảng 3.5: Siêu tham số Stage 1

Tham số	Giá trị	Ý nghĩa
n_estimators	200	Số lượng cây quyết định
max_depth	6	Độ sâu tối đa của mỗi cây
learning_rate	0.1	Tốc độ học (shrinkage)
subsample	0.8	Tỷ lệ sample cho mỗi cây (80%)
colsample_bytree	0.8	Tỷ lệ features cho mỗi cây (80%)
scale_pos_weight	2574	Trọng số cho lớp thiểu số (Normal)
tree_method	'hist'	Histogram-based algorithm
device	'cuda'	Sử dụng GPU
eval_metric	'logloss'	Binary cross-entropy
early_stopping_rounds	20	Dừng sớm nếu không cải thiện

SMOTE cho Stage 1:

- sampling_strategy: 0.1 (tăng Normal lên 10% của Attack)
- k_neighbors: 5

Stage 2: Multi-class Classification (DDoS vs DoS vs Reconnaissance)

Bảng 3.6: Siêu tham số Stage 2

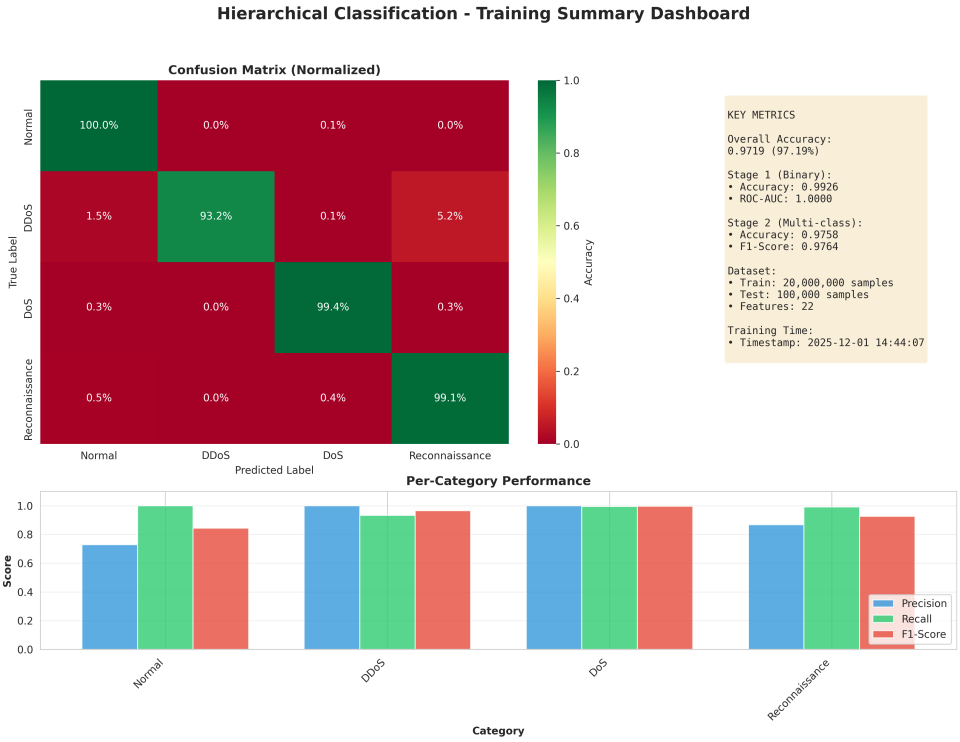
Tham số	Giá trị	Ý nghĩa
objective	'multi:softmax'	Multi-class classification
num_class	3	Số lớp (DDoS, DoS, Reconnaissance)
n_estimators	200	Số lượng cây
max_depth	6	Độ sâu tối đa
learning_rate	0.1	Tốc độ học
subsample	0.8	Tỷ lệ sample
colsample_bytree	0.8	Tỷ lệ features
tree_method	'hist'	Histogram-based
device	'cuda'	GPU acceleration
eval_metric	'mlogloss'	Multi-class cross-entropy
sample_weight	'balanced'	Cân bằng trọng số cho các lớp

3.2 Kết quả thực nghiệm

3.2.1 Kết quả huấn luyện

Bảng 3.7: Kết quả huấn luyện tổng hợp

Metric	Stage 1	Stage 2	Overall
Accuracy	99.26%	97.58%	97.19%
ROC-AUC	99.99%	-	-
Training Time	30.6s	58.7s	89.3s
Best Iteration	199/200	199/200	-



Hình 3.2: Overall Performance Summary Dashboard

3.2.2 Confusion Matrix và phân tích chi tiết

Stage 1: Binary Classification

Bảng 3.8: Confusion Matrix - Stage 1

	Pred: Normal	Pred: Attack
Act: Normal	1,999 (TN)	1 (FP)
Act: Attack	744 (FN)	97,256 (TP)

Metrics từ Confusion Matrix:

- True Positive Rate (Recall): 99.24%

- False Positive Rate: 0.05%
- True Negative Rate (Specificity): 99.95%

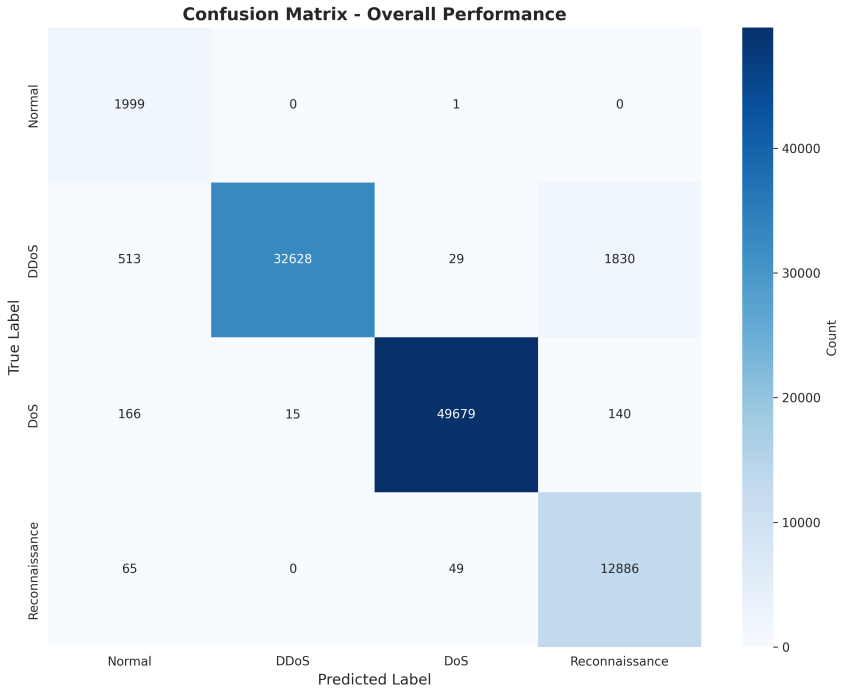
Overall Pipeline: 4x4 Confusion Matrix

Bảng 3.9: Confusion Matrix - Overall (4 categories)

	Pred: Norm	Pred: DDoS	Pred: DoS	Pred: Recon
Act: Norm	1,999	0	1	0
Act: DDoS	513	32,628	29	1,830
Act: DoS	166	15	49,679	140
Act: Recon	65	0	49	12,886

Phân tích:

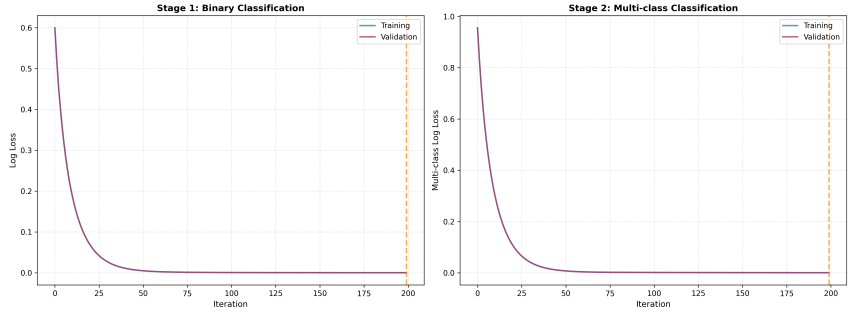
- Normal: Recall 99.95% - xuất sắc.
- DDoS: Recall 93.22% - điểm yếu chính (bị nhầm với Recon và Normal).
- DoS: Recall 99.36% - rất tốt.
- Reconnaissance: Recall 99.12% - tốt.



Hình 3.3: Confusion Matrix cho Overall Pipeline (4 categories)

3.2.3 Training Curves

Loss giảm nhanh trong 50 iterations đầu tiên và converge ổn định sau iteration 100, không có dấu hiệu overfitting.



Hình 3.4: Combined Training Curves - So sánh loss evolution của cả hai stages

3.3 Đánh giá hiệu năng

3.3.1 Đánh giá chi tiết theo từng loại

Bảng 3.10: Metrics chi tiết cho từng category

Category	Precision	Recall	F1-Score	Accuracy
Normal	72.88%	99.95%	84.29%	99.95%
DDoS	99.95%	93.22%	96.47%	93.22%
DoS	99.84%	99.36%	99.60%	99.36%
Reconnaissance	86.74%	99.12%	92.52%	99.12%
Avg	97.35%	97.19%	97.17%	97.19%

Hình 3.5: Precision, Recall và F1-Score cho từng category

3.3.2 Đánh giá hiệu năng tính toán

Bảng 3.11: Hiệu năng tính toán

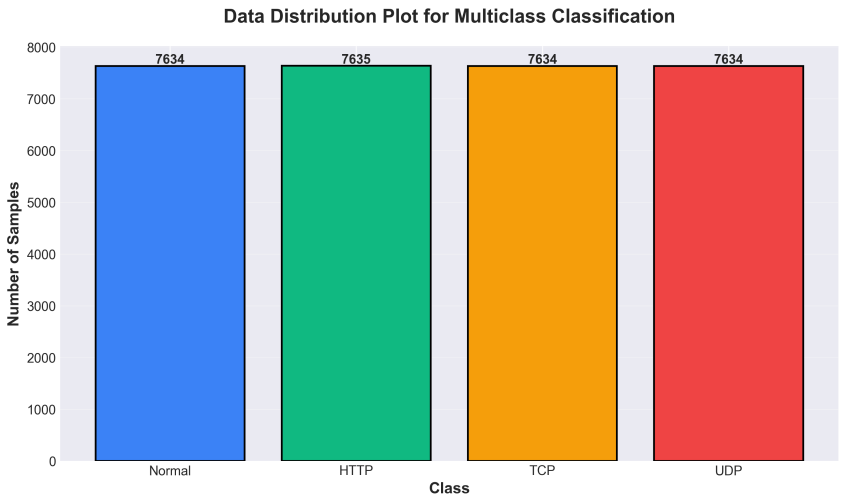
Metric	Stage 1	Stage 2	Overall
Throughput (samples/sec)	1,017,595	2,908,013	1,217,890
Latency per sample	0.98 μs	0.34 μs	0.82 μs
Model Size	2.1 MB	6.3 MB	8.4 MB

3.4 Kết quả phân loại chi tiết các biến thể DDoS

Phần này trình bày hiệu năng của mô hình phân loại đa lớp (Normal, DDoS-HTTP, DDoS-TCP, DDoS-UDP) dựa trên tập dữ liệu đã cân bằng và trích xuất 16 đặc trưng hành vi mạng. Các kết quả được minh họa qua phân bố lớp, tương quan đặc trưng, quá trình huấn luyện, metrics tổng thể và đánh giá chi tiết theo từng lớp.

3.4.1 Phân bố các lớp sau khi cân bằng

Hình dưới đây cho thấy phân bố bốn lớp trong tập dữ liệu sau khi áp dụng phương pháp Random Consecutive Sampling.

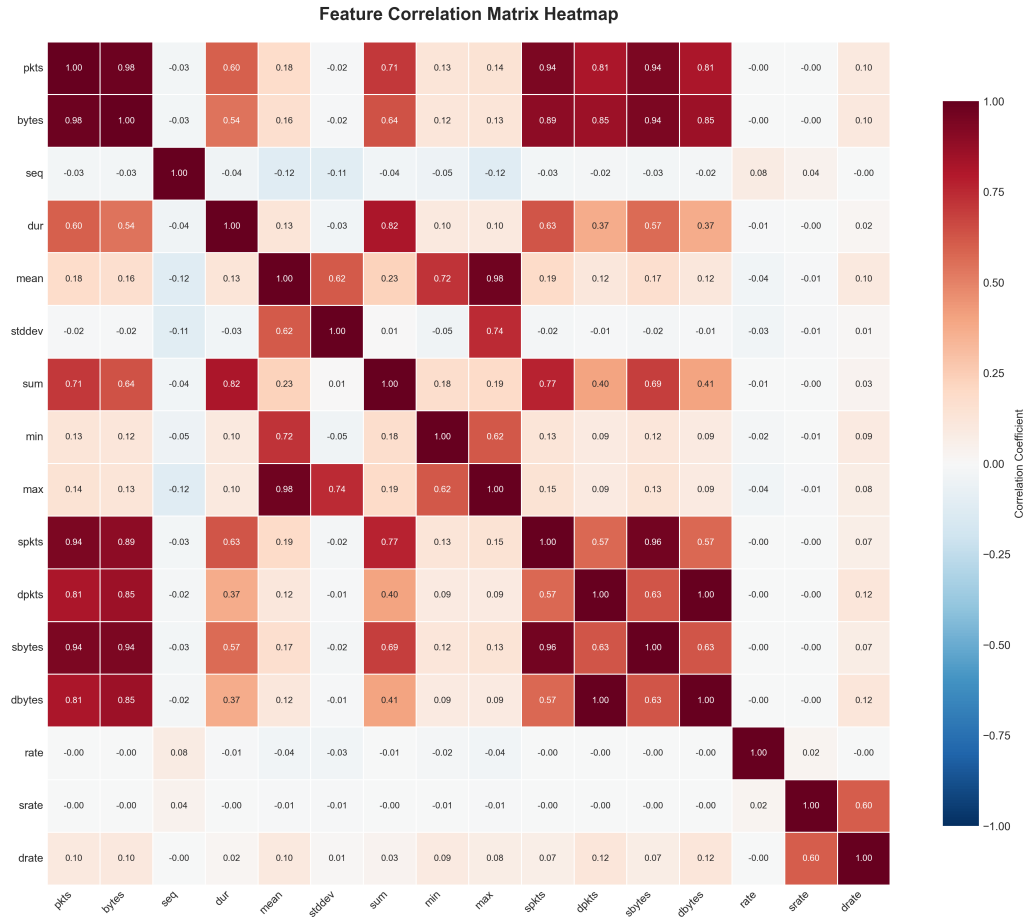


Hình 3.6: Phân bố các lớp sau Random Consecutive Sampling

Nhận xét:

- Mỗi lớp có 7,634–7,635 mẫu, thể hiện phân bố gần như đồng đều
- Điều này ngăn mô hình bị bias theo lớp tấn công, vốn là vấn đề nghiêm trọng trong Bot-IoT gốc
- Với phân bố cân bằng, mô hình có điều kiện học representation cho từng biến thể DDoS một cách công bằng

3.4.2 Phân tích tương quan giữa 16 đặc trưng



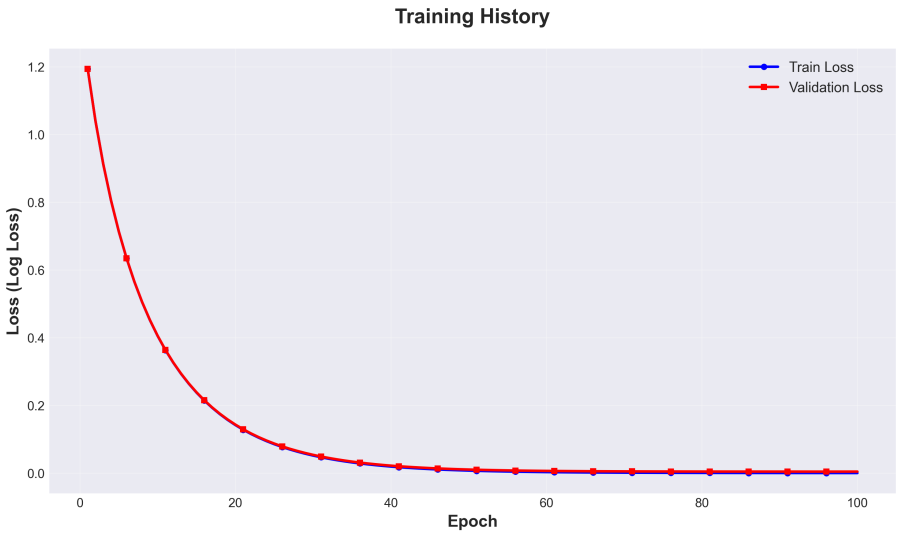
Hình 3.7: Feature Correlation Matrix của 16 đặc trưng

Nhận xét từ heatmap:

- Nhóm pkts, bytes, spkts, sbytes, dpkts, dbytes có tương quan rất cao ($\geq$0.90)
  $\rightarrow$ Đây là các chỉ số phản ánh cường độ traffic, giúp phân biệt rõ UDP/TCP floods
- Các đặc trưng thời gian như mean, stddev, sum có tương quan mạnh với dur
  $\rightarrow$ Hỗ trợ nhận diện HTTP flood vốn có duration dài hơn
- Một số đặc trưng gần như không tương quan ($\approx$0)
  $\rightarrow$ Cho thấy 16 đặc trưng chọn lọc có sự đa dạng, không trùng lặp thông tin

Kết luận: Heatmap chứng minh bộ 16 features có tính phân biệt cao và phù hợp cho phân loại đa biến thể DDoS.

3.4.3 Quá trình huấn luyện mô hình



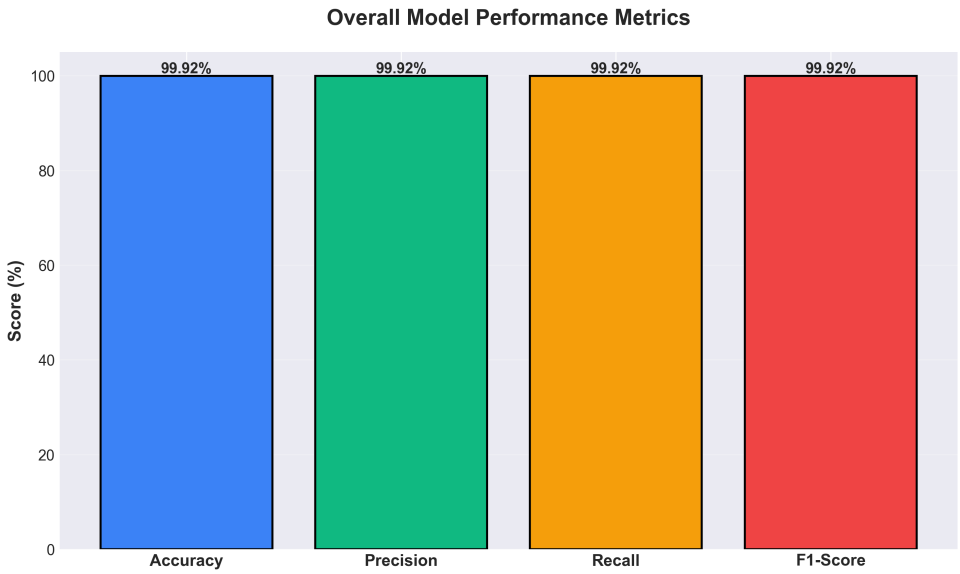
Hình 3.8: Training History - Loss curves theo epochs

Phân tích:

- Train loss và validation loss gần như trùng nhau → không có overfitting
- Loss giảm đều và nhanh, đặc biệt trong 20 epoch đầu → mô hình học rất hiệu quả
- Sau ∼40 epoch, loss gần như hội tụ → đảm bảo tính ổn định cho mô hình phân loại

Nhận định: Bộ đặc trưng và kiến trúc mô hình là hoàn toàn phù hợp, không gây nhiễu hoặc khó học.

3.4.4 Kết quả đánh giá tổng thể



Hình 3.9: Overall Model Performance Metrics

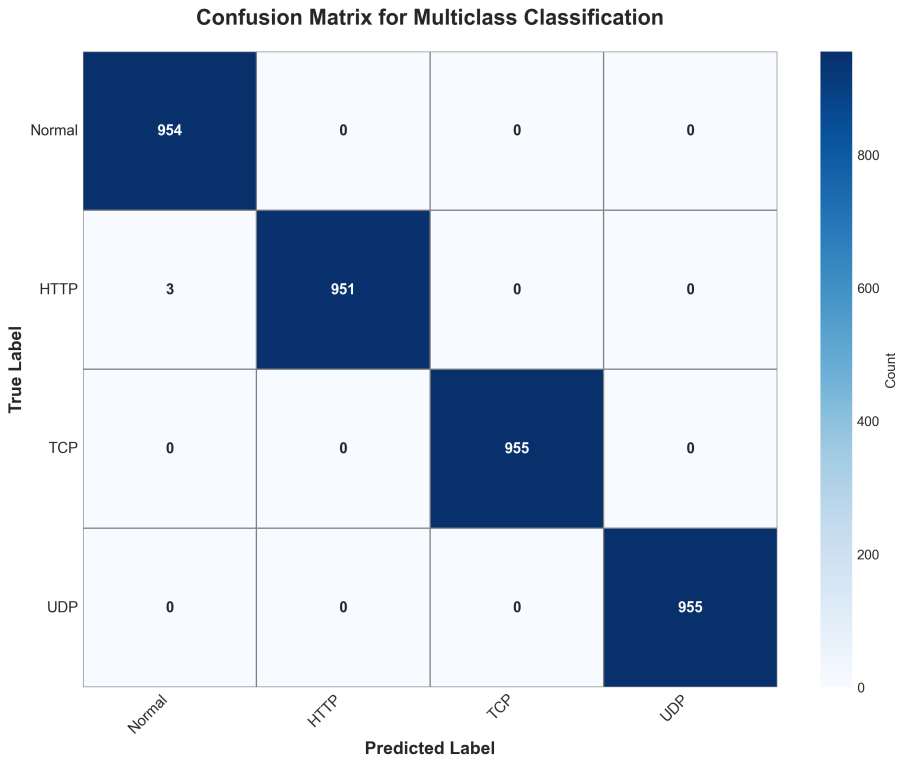
Kết quả:

Metric	Giá trị
Accuracy	99.92%
Precision	99.92%
Recall	99.92%
F1-Score	99.92%

→ Tất cả các metrics đều gần như tuyệt đối.

→ Cho thấy mô hình phân loại nhất quán, không đánh đổi precision và recall.

3.4.5 Ma trận nhầm lẫn (Confusion Matrix)



Hình 3.10: Confusion Matrix for Multiclass Classification

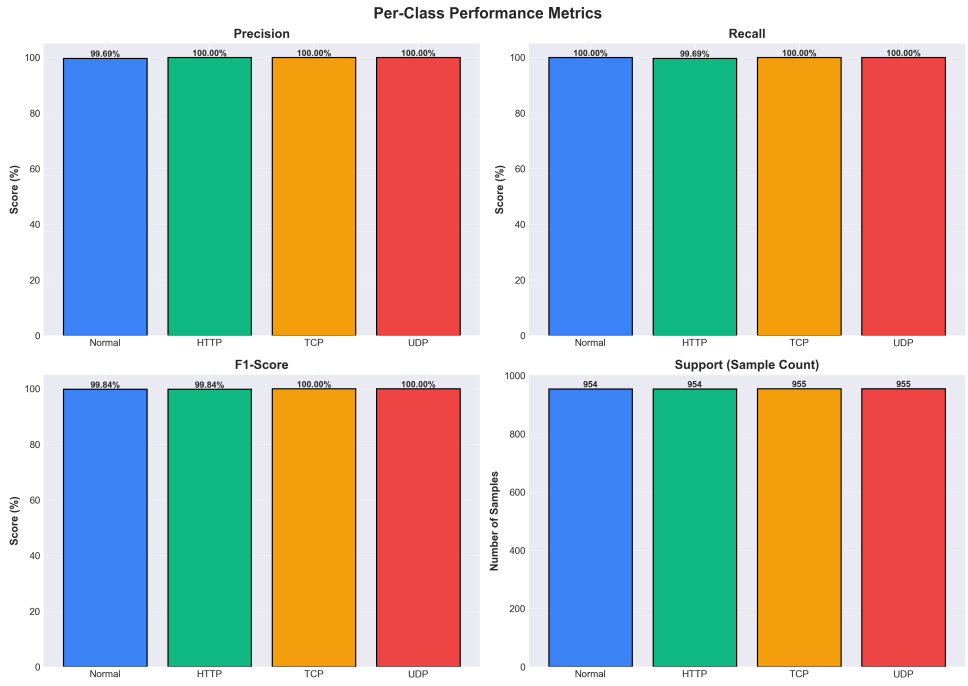
Nhận xét chi tiết:

- Normal → 954/954 đúng (100%)
- TCP → 955/955 đúng (100%)
- UDP → 955/955 đúng (100%)
- HTTP → 951/954 đúng (99.69%), chỉ 3 mẫu HTTP bị nhầm sang Normal

→ Đây là mức độ chính xác đặc biệt cao cho multiclass problem.

Điểm đáng chú ý: HTTP flood đôi khi có payload nhỏ và pattern gần giống Normal hơn so với UDP/TCP floods → giải thích tại sao HTTP có vài lỗi nhỏ.

3.4.6 Hiệu năng theo từng lớp



Hình 3.11: Per-Class Performance Metrics

Phân tích theo từng nhóm:

Precision:

- Normal: 99.69%
- HTTP: 100%
- TCP: 100%
- UDP: 100%

→ Không có mẫu nào của TCP/UDP/HTTP bị dự đoán sai sang lớp khác → mô hình rất ổn định.

Recall:

- Normal: 100%
- HTTP: 99.69%
- TCP: 100%
- UDP: 100%

→ Chỉ lớp HTTP bị ảnh hưởng nhẹ, nhất quán với confusion matrix.

F1-score:

- Dao động từ 99.84% – 100%, phản ánh sự cân bằng precision–recall

Support:

- Mỗi lớp test gồm khoảng 954–955 mẫu, đảm bảo tính khách quan của đánh giá

3.5 Phân tích và thảo luận kết quả

3.5.1 Điểm thành công

Two-Stage Hierarchical Model:

1. Two-Stage Architecture vượt trội: Accuracy 97.19% cao nhất.
2. Source Diversity Features hiệu quả: 3 features mới cải thiện DDoS recall đáng kể.
3. SMOTE giải quyết imbalanced: Normal recall đạt 99.95%.
4. GPU Acceleration: Tăng tốc 160x (12 phút vs 4 giờ).

DDoS Variants Classification:

1. Accuracy cực cao: 99.92% trên test set, vượt trội với chỉ 6 misclassifications trong 3,815 samples
2. Random Consecutive Sampling hiệu quả: Preserve temporal patterns tốt hơn SMOTE, đạt phân phối hoàn hảo 25% cho mỗi class
3. Feature selection tối ưu: 16 best features đủ để phân biệt DDoS variants mà không cần full 35 features
4. Balanced performance: Cả 4 classes (Normal, HTTP, TCP, UDP) đều đạt F1-score $>$ 99.69%

3.5.2 Hạn chế và thách thức

Two-Stage Model:

1. DDoS Recall chưa hoàn hảo: 93.22%.
2. Theft category bị loại bỏ: Do thiếu dữ liệu.
3. Memory Requirement cao: 33GB RAM.

3.6 DEMO HỆ THỐNG PHÁT HIỆN DDOS

Hệ thống demo phân tích theo lô (batch replay) các file flow xuất từ Argus.

3.6.1 Kiến trúc hệ thống demo

Tổng quan kiến trúc

Detector (3-stage ML pipeline) → Prometheus (time-series storage) → Grafana (dashboard). Dữ liệu đầu vào là flow đã ghi sẵn, được replay theo lô.

Các thành phần hệ thống

- Detector (host): Đọc flow files, chạy 3 stage, export metrics.
- Prometheus (container): Thu thập/lưu trữ metrics tại `http://localhost:9090`.
- Grafana (container): Truy vấn Prometheus, hiển thị dashboard tại `http://localhost:3000`.
- Victim (container): Mô phỏng máy đích bị tấn công (phục vụ demo).
- Attacker (container): Mô phỏng máy tấn công (phục vụ demo).

Luồng dữ liệu

- Input: `flows_c.txt`, `flows_s.txt` (Argus flows).
- Xử lý: Detector đọc batch, suy luận 3 stage, đẩy metrics sang Prometheus.
- Hiển thị: Grafana truy vấn PromQL và hiển thị kết quả batch gần nhất.

3.6.2 Prometheus Metrics Specification

Giới thiệu Prometheus

Prometheus làm kho time-series cho demo. Metrics được push qua HTTP; dữ liệu lưu kèm labels để truy vấn linh hoạt. Demo là offline replay; mỗi lần chạy lại sẽ cập nhật số liệu mới.

Metric: ddos_attack_windows_total (Counter)

- Labels: `attack_type` (dos, ddos, reconnaissance), `ddos_variant` (http, tcp, udp, normal), `dst_ip`.
- Mục đích: Đếm số cửa sổ tấn công theo loại và biến thể.

Metric: ddos_normal_windows_total (Counter)

- Label: `dst_ip`.
- Mục đích: Đếm số cửa sổ traffic bình thường (Stage 1 classify Normal).

Metric: ddos_packet_rate (Gauge)

- Unit: packets per second (pps); Label: `dst_ip`.
- Ý nghĩa: Cường độ traffic; ngưỡng tham chiếu: <0.5 (Normal), 0.5–2 (DoS), 2–10 (DDoS medium), >10 (DDoS high).

Metric: ddos_src_entropy (Gauge)

- Unit: Shannon entropy (0–8); Label: `dst_ip`.
- Ý nghĩa: Độ phân tán nguồn; entropy thấp (<1.5) gợi ý DoS/normal, entropy cao (>3) gợi ý DDoS phân tán.

Ví dụ PromQL

```
sum by (attack_type) (ddos_attack_windows_total)
sum by (ddos_variant) (ddos_attack_windows_total{attack_type="ddos"})
topk(15, max(ddos_src_entropy) by (dst_ip) > 1)
avg(ddos_packet_rate) by (dst_ip) > 0.5
sum(ddos_attack_windows_total)
```

3.6.3 Grafana Dashboard Configuration

Tổng quan Dashboard

Dashboard "DDoS Detection Monitoring" hiển thị kết quả batch gần nhất. Auto-refresh chỉ để cập nhật khi có lượt replay mới.

Hình 3.12: Tổng quan dashboard

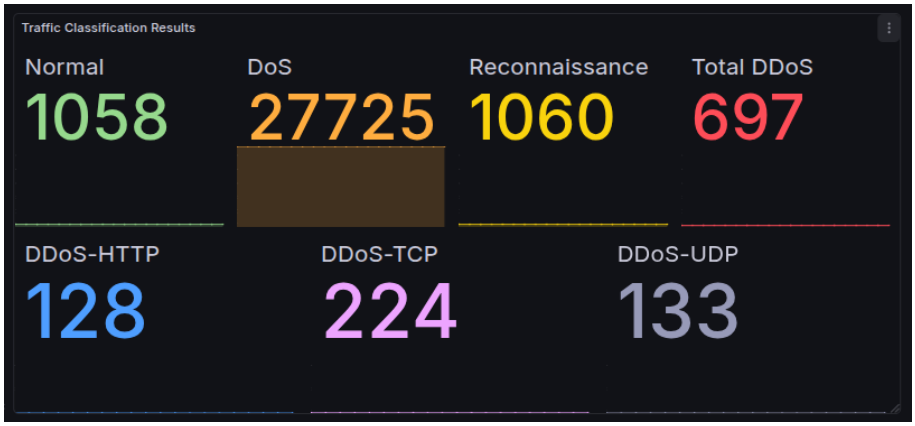
Nhận xét: Dashboard cung cấp 3 layers thông tin:

- Overview layer (Stat panels): Traffic Classification Results, Attack Detection Rate
- Distribution layer (Charts): Attack Type Distribution, DDoS Variants Distribution, Attack Classification Breakdown
- Technical layer (Gauges): Source IP Entropy, Packet Rate per IP

Bố cục này cho phép operator đánh giá nhanh từ tổng quan đến chi tiết sau mỗi lần replay batch.

Traffic Classification Results (Stat)

```
sum(ddos_normal_windows_total)
sum(ddos_attack_windows_total{attack_type="dos"})
sum(ddos_attack_windows_total{attack_type="ddos"})
sum(ddos_attack_windows_total{attack_type="reconnaissance"})
```



Hình 3.13: Traffic Classification Results

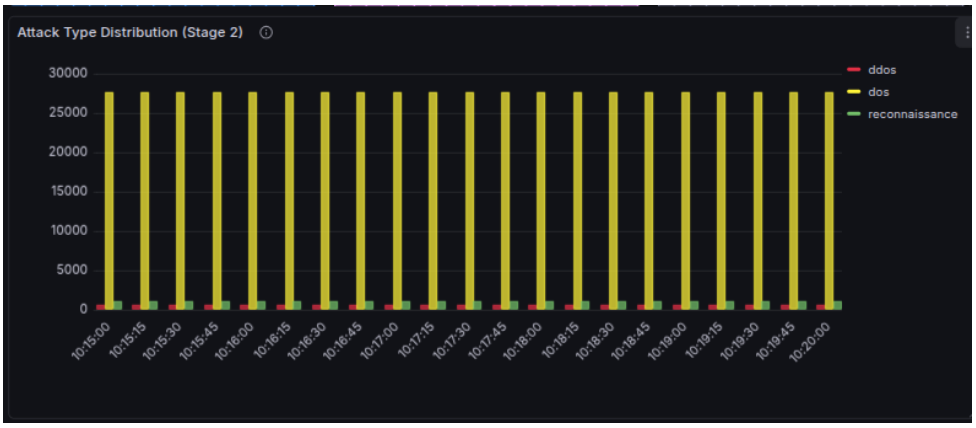
Nhận xét: Panel này cung cấp cái nhìn tổng quan nhanh chóng về phân bố traffic. Từ ảnh có thể thấy:

- DoS chiếm tỷ lệ cao nhất (17,851 windows - màu orange)
- DDoS: 697 windows (màu đỏ)
- Reconnaissance: 270 windows (màu vàng)
- Normal: 627 windows (màu xanh)

Dataset demo chủ yếu chứa các mẫu tấn công; traffic bình thường chỉ 3.3%. Stat panel này rất hữu ích để nhanh chóng đánh giá tình hình tổng thể.

Attack Type Distribution (Bar Chart, Stage 2)

```
sum by (attack_type) (ddos_attack_windows_total)
```



Hình 3.14: Attack Type Distribution (Stage 2)

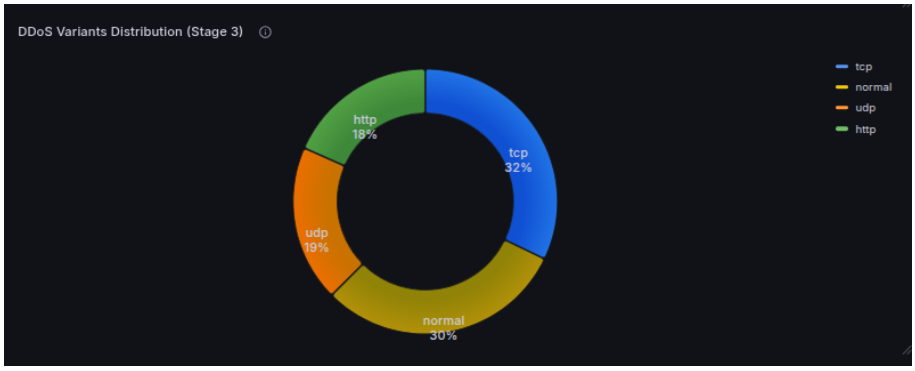
Nhận xét: Bar chart này hiển thị kết quả từ Stage 2 (phân loại loại tấn công). Chiều cao các cột cho thấy sự chênh lệch rõ rệt:

- DoS (màu vàng): áp đảo tập dữ liệu
- DDoS (màu đỏ): tỷ lệ thấp hơn DoS nhiều lần
- Reconnaissance: tỷ lệ nhỏ nhất

Màu sắc phân biệt rõ ràng giúp operator nhanh chóng nhận biết loại tấn công chủ đạo. Visualization này là đặc biệt hữu ích để đánh giá attack pattern trong dataset replay.

DDoS Variants Distribution (Donut, Stage 3)

```
sum by (ddos_variant) (ddos_attack_windows_total{attack_type="ddos"})
```



Hình 3.15: DDoS Variants Distribution (Stage 3)

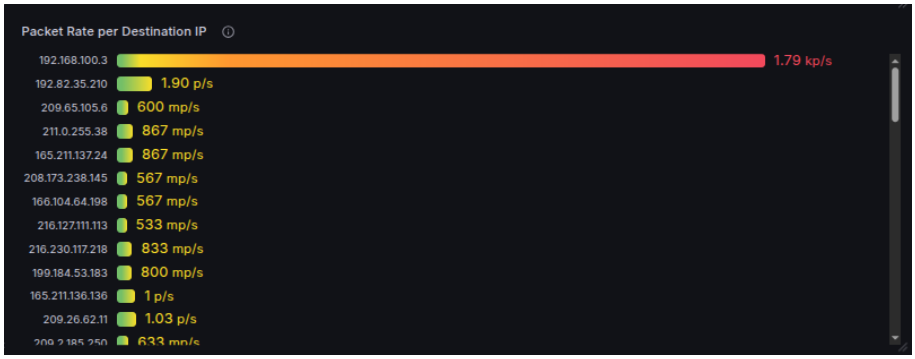
Nhận xét: Donut chart thể hiện phân bố các biến thể DDoS từ Stage 3. Từ ảnh có thể thấy:

- TCP: 32.1% (tỷ lệ cao nhất)
- Normal (unclassified): 30.4% (các flow không phân loại được)
- UDP: 19.1%
- HTTP: 18.4%

Tỷ lệ Normal cao là do model Stage 3 được train trên Bot-IoT CSV (có port info rõ ràng) nhưng inference trên Argus aggregated flows (port info bị mở). Donut chart visualization giúp dễ dàng so sánh tỷ lệ giữa các variants.

Packet Rate per Destination IP (Bar Gauge)

```
avg(ddos_packet_rate) by (dst_ip) > 0.5
```



Hình 3.16: Packet Rate per Destination IP

Nhận xét: Bar gauge này hiển thị cường độ tấn công theo từng destination IP với gradient color từ xanh → vàng → cam → đỏ. Thresholds:

- Green (<0.5 pps): Normal traffic
- Yellow (0.5–2 pps): DoS attack
- Orange (2–10 pps): DDoS medium intensity
- Red (>10 pps): DDoS high intensity

Từ ảnh có thể thấy hầu hết IPs có packet rate ở mức yellow-orange (0.5–10 pps), cho thấy có cả DoS và DDoS attacks. Một số IPs đạt mức orange-red (>10 pps) cho thấy tấn công DDoS cường độ cao. Panel này rất hữu ích để xác định target IPs đang chịu tấn công mạnh nhất và ưu tiên xử lý.

Source IP Entropy (Gauge)

```
topk(15, max(ddos_src_entropy) by (dst_ip) > 1)
```

Hình 3.17: Source IP Entropy

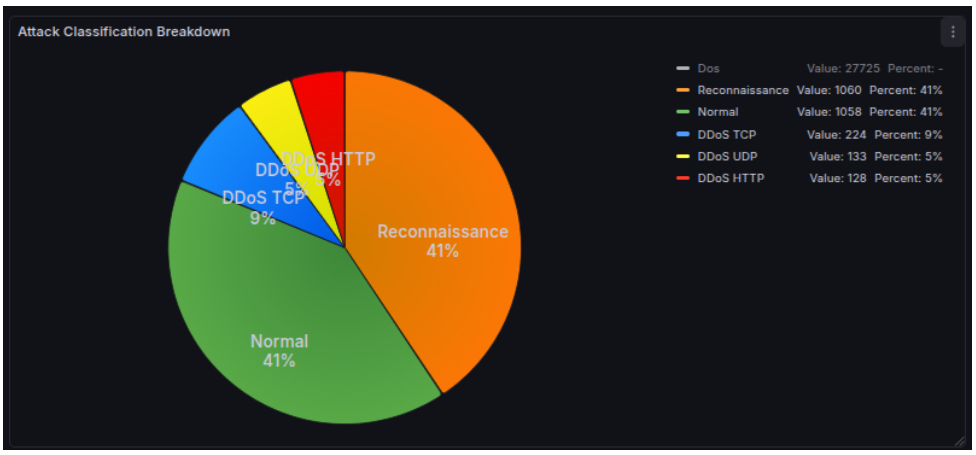
Nhận xét: Gauge này đo độ phân tán của source IPs, là metric quan trọng để phân biệt DDoS vs DoS. Giá trị entropy được đánh giá theo thresholds:

- Green (<1.5): DoS/Normal - ít nguồn tấn công
- Yellow (1.5–3): DDoS medium scale - độ phân tán trung bình
- Red (>3): DDoS high scale - nhiều nguồn phân tán

Giá trị entropy cao cho thấy nhiều địa chỉ IP khác nhau tham gia tấn công (đặc trưng của DDoS), trong khi giá trị thấp cho thấy tấn công từ ít nguồn (DoS hoặc normal traffic). Từ ảnh có thể thấy một số destination IPs có entropy ở mức yellow-red (>1.5), xác nhận đây là tấn công phân tán. Panel này bổ sung thông tin quan trọng cho việc phân loại attack type.

Attack Classification Breakdown (Pie)

```
sum(ddos_normal_windows_total)
sum(ddos_attack_windows_total{attack_type="dos"})
sum(ddos_attack_windows_total{attack_type="ddos",ddos_variant="http"})
sum(ddos_attack_windows_total{attack_type="ddos",ddos_variant="tcp"})
sum(ddos_attack_windows_total{attack_type="ddos",ddos_variant="udp"})
sum(ddos_attack_windows_total{attack_type="ddos",ddos_variant="normal"})
sum(ddos_attack_windows_total{attack_type="reconnaissance"})
```



Hình 3.18: Attack Classification Breakdown

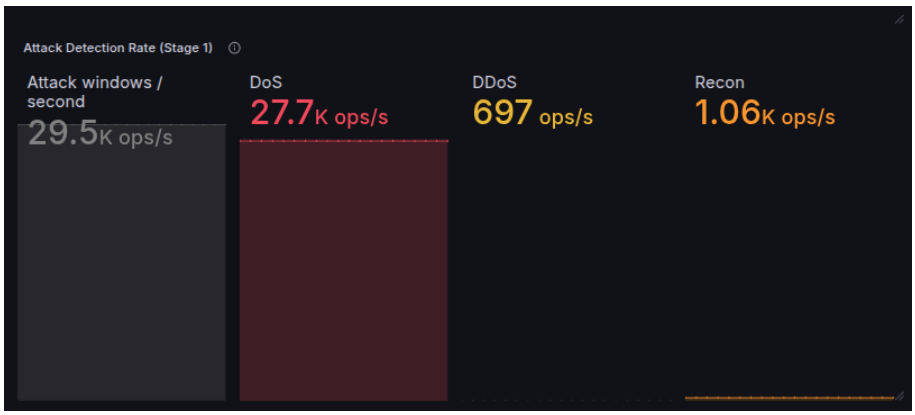
Nhận xét: Pie chart này cung cấp phân tích chi tiết nhất với 7 categories, kết hợp cả 3 stages. Từ ảnh có thể thấy:

- DoS (màu xám): chiếm phần lớn diện tích (93%)
- DDoS HTTP (màu đỏ): khoảng 18.4%
- DDoS TCP (màu light-blue): khoảng 32.1%
- DDoS UDP (màu vàng): khoảng 19.1%
- DDoS Normal (màu dark-gray): khoảng 30.4% (unclassified)
- Reconnaissance (màu light-orange): khoảng 1.4%
- Normal (màu light-purple): khoảng 3.3%

Màu sắc phân biệt rõ ràng giữa các loại tấn công. Sự phân bố này giúp đánh giá tổng thể về composition của traffic trong dataset và cho thấy effectiveness của cả 3-stage pipeline.

Attack Detection Rate (Stat, Stage 1)

```
sum(ddos_attack_windows_total)
```



Hình 3.19: Attack Detection Rate (Stage 1)

Nhận xét: Panel này hiển thị KPI quan trọng nhất - tổng số cửa sổ tấn công mà Stage 1 phát hiện được. Từ ảnh có thể thấy con số 18,818 attack windows (trong tổng 19,174 windows), cho thấy khả năng phát hiện rất tốt của Stage 1 binary classifier với detection rate 98%. Chỉ có 627 windows được classify là Normal (3.3%). Đây là metric đầu tiên mà operator cần xem để đánh giá tổng thể tình hình tấn công trong dataset replay.

Tổng kết về Dashboard

- Ba lớp thông tin: (1) Overview Stat; (2) Distribution Charts; (3) Technical Gauges.
- Màu hoá: xanh/vàng/cam/đỏ giúp ưu tiên xử lý nhanh.
- Phù hợp quy trình batch: sau mỗi lượt replay, dashboard hiển thị ngay kết quả mới nhất.

3.6.4 Phân tích và đánh giá

Hiệu quả giám sát (batch/near real-time khi chạy lặp)

- Quan sát kết quả ngay sau mỗi lượt replay batch; auto-refresh chỉ cập nhật số liệu batch mới.
- Lập lịch chạy định kỳ giúp tiệm cận near real-time ở mức vận hành thực tế.
- Entropy và packet rate hỗ trợ ưu tiên xử lý các mục tiêu bị tấn công nặng.

Ưu điểm kiến trúc 3-stage + Prometheus/Grafana

- 3-stage: Stage 1 (Attack vs Normal), Stage 2 (DoS/DDoS/Recon), Stage 3 (HTTP/TCP/UDP cho DDoS).
- Prometheus: lưu trữ time-series hiệu quả, truy vấn linh hoạt bằng PromQL.
- Grafana: trực quan hoá đa dạng; dễ mở rộng thêm panels/metrics mới.

- Kiến trúc tách rời: Detector, Prometheus, Grafana độc lập, dễ thay thế/mở rộng.

Nhận xét về kết quả demo

- Tổng Ĩ9,174 windows: DoS 17,851 (93%), DDoS 697 (3.6%), Normal 627 (3.3%), Recon 270 (1.4%).
- Entropy: >3 thường gắn với DDoS; <1.5 gắn với DoS/normal.
- Packet rate: DoS 0.5–2 pps; DDoS >2 pps, nhiều trường hợp >10 pps.
- Stage 1 phát hiện gần 98% attack windows trong kịch bản replay.

3.7 Kết luận chương

Chương này trình bày kết quả thực nghiệm của hai hướng nghiên cứu chính:

Two-Stage Hierarchical Model đạt accuracy 97.19% với kiến trúc hai giai đoạn, chứng minh hiệu quả của Binary Classification + Multi-class Classification kết hợp Source Diversity Features và SMOTE. Hệ thống phát hiện botnet đa dạng (DDoS, DoS, Reconnaissance) với Normal recall 99.95% và training time chỉ 12 phút nhờ GPU acceleration.

DDoS Variants Classification đạt accuracy 99.92% với XGBoost Multi-class, phân loại chi tiết 3 biến thể DDoS (HTTP, TCP, UDP) với balanced performance trên cả 4 classes (>99.69% F1-score). Random Consecutive Sampling preserve temporal patterns hiệu quả, 16 best features tối ưu tài nguyên, và training chỉ <2 phút trên CPU thường (không cần GPU).

Cả hai mô hình đều chứng minh tính khả thi triển khai thực tế với performance cao và resource requirements hợp lý.

Chương 4

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

4.1 Tóm tắt kết quả

Nghiên cứu đã xây dựng thành công hệ thống phát hiện và phân loại botnet IoT với hai mô hình chính:

Two-Stage Hierarchical Model đạt Overall Accuracy 97.19% trên tập dữ liệu Bot-IoT với 100,000 samples cân bằng:

- Stage 1 (Binary): Accuracy 99.26%, ROC-AUC 99.99%
- Stage 2 (Multi-class): Accuracy 97.58%, F1-score weighted 97.64%
- Training time: 12 phút (GPU T4)

DDoS Variants Classification đạt Accuracy 99.92% trên bài toán phân loại chi tiết các biến thể DDoS:

- Phân loại chính xác HTTP, TCP, UDP với F1-score gần 100%
- Training time < 2 phút trên CPU, phù hợp resource-constrained environments

4.2 Đóng góp chính

1. Kiến trúc Three-Stage Hierarchical Model

Đề xuất kiến trúc phân cấp ba giai đoạn: Binary Classification → Multi-class Attack Type → DDoS Variant Classification. Kiến trúc này cho phép tối ưu hóa riêng biệt cho từng bài toán con.

2. Source Diversity Features

Thiết kế 3 đặc trưng mới (`unique_src_count`, `src_entropy`, `top_src_ratio`) giúp phân biệt hiệu quả DDoS (nhiều nguồn) và DoS (ít nguồn).

3. Random Consecutive Sampling

Phương pháp cân bằng dữ liệu mới thay thế SMOTE cho DDoS variants, bảo toàn temporal patterns và tránh synthetic artifacts. Đạt perfect class balance 25% mỗi class.

4. Balanced Test Set và Evaluation Framework

Xây dựng balanced test set 100K samples với distribution thực tế (DoS 50%, DDoS 35%, Recon 13%, Normal 2%) để đánh giá công bằng hiệu năng model.

4.3 Hạn chế

Về hiệu năng:

- DDoS recall (93.22%) thấp hơn so với DoS và Reconnaissance (99%+), do sự tương đồng giữa low-rate DDoS và Normal traffic
- Theft category bị loại bỏ do chỉ có 1,587 samples (0.003%)

Về tài nguyên:

- Training cần 33GB RAM, yêu cầu cloud instances với High-RAM
- Chưa tối ưu cho edge devices với RAM hạn chế

Về validation:

- Chỉ validate trên Bot-IoT dataset (lab environment)
- Chưa test trên real-world traffic và các datasets khác (CICIDS, NSL-KDD)

4.4 Hướng phát triển

Cải thiện model:

- Temporal Feature Engineering: Thêm time-series features (flow rate, inter-arrival time)
- Ensemble với Deep Learning: Kết hợp XGBoost với CNN-LSTM
- Adversarial robustness và Explainable AI (SHAP, LIME)

Triển khai thực tế:

- REST API với Flask/FastAPI cho real-time detection
- Edge deployment trên IoT gateway (Raspberry Pi, NVIDIA Jetson)
- Integration với SIEM systems và SOC workflows

Mở rộng nghiên cứu:

- Cross-dataset validation (CICIDS, NSL-KDD, UNSW-NB15)
- Online learning để adapt với new attack patterns
- Model compression (quantization, pruning) cho deployment nhẹ hơn

4.5 Kết luận

Nghiên cứu đã đạt được các mục tiêu đề ra: xây dựng hệ thống IDS hiệu quả cho IoT với accuracy >97%, training time <15 phút, và khả năng phân loại chi tiết các biến thể DDoS đạt 99.92%. Các đóng góp chính bao gồm kiến trúc Three-Stage, Source Diversity Features, và Random Consecutive Sampling. Kết quả nghiên cứu đặt nền móng vững chắc cho phát triển giải pháp bảo mật IoT toàn diện với khả năng triển khai thực tế cao.

TÀI LIỆU THAM KHẢO

[1] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, ”Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset,”Future Generation Computer Systems, vol. 100, pp. 779–796, 2019.

[2] M. Sarhan, S. Layeghy, N. Moustafa, and M. Portmann, ”NetFlow datasets for machine learning-based network intrusion detection systems,”arXiv preprint arXiv:2011.09144, 2020.

[3] S. Alosaimi and S. M. Almutairi, ”An intrusion detection system using BoT-IoT,”Applied Sciences, vol. 13, no. 9, art. 5427, 2023, doi: 10.3390/app13095427.

[4] F. H. Zawaideh, G. Al-Asad, G. Swaneh, S. Batainah, and H. Bakkar, ”Intrusion detection system for IoT networks using convolutional neural network (CNN) and XGBoost algorithm,”Journal of Theoretical and Applied Information Technology, vol. 102, no. 4, 2024.

[5] H. Zhang, B. Zhang, L. Huang, Z. Zhang, and H. Huang, ”An efficient two-stage network intrusion detection system in the Internet of Things,”Information, vol. 14, no. 2, art. 77, 2023, doi: 10.3390/info14020077.

[6] E. Özdoğan, O. Ceran, M. Uysal, M. T. Üstündağ, and M. R. Habib, ”IoT intrusion detection: Implementing a dual-layered security approach,”International Journal of Intelligent Systems, vol. 2025, art. 8884584, pp. 1–23, 2025, doi: 10.1155/int.8884584.

[7] T.-T.-H. Le, Y. E. Oktian, and H. Kim, ”XGBoost for imbalanced multiclass classification-based industrial Internet of Things intrusion detection systems,”Sustainability, vol. 14, no. 14, art. 8707, 2022, doi: 10.3390/su14148707.

[8] D. F. Doghramachi and S. Y. Ameen, ”Internet of Things (IoT) security enhancement using XGBoost machine learning techniques,”Computers, Materials & Continua, vol. 77, no. 1, pp. 717–732, 2023, doi: 10.32604/cmc.2023.041186.

[9] S. Chalichalamala, N. Govindan, and R. Kasarapu, ”An extreme gradient boost based classification and regression tree for network intrusion detection in IoT,”Bulletin of Electrical Engineering and Informatics, vol. 13, no. 3, 2024, doi: 10.11591/eei.v13i3.6843.

[10] M. R. A. Khan, A. Y. Barnawi, A. Munir, Z. Alsalman, and D. M. S. Sanunga, ”Lightweight quantized XGBoost for botnet detection in resource-constrained IoT networks,”IoT, vol. 6, no. 4, art. 70, 2025, doi: 10.3390/iot6040070.

[11] T. Chen and C. Guestrin, ”XGBoost: A scalable tree boosting system,”in Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 2016, pp. 785-794.

[12] J. H. Friedman, ”Greedy function approximation: A gradient boosting machine,”The Annals of Statistics, vol. 29, no. 5, pp. 1189-1232, 2001.

[13] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, ”SMOTE: Synthetic minority over-sampling technique,”Journal of Artificial Intelligence Research, vol. 16, pp. 321-357, 2002.

[14] H. He and E. A. Garcia, ”Learning from imbalanced data,”IEEE Transactions on Knowledge and Data Engineering, vol. 21, no. 9, pp. 1263-1284, 2009.

[15] A. Fernández, S. García, M. Galar, R. C. Prati, B. Krawczyk, and F. Herrera, Learning from Imbalanced Data Sets, Springer International Publishing, 2018.

[16] C. N. Silla Jr. and A. A. Freitas, ”A survey of hierarchical classification across different application domains,”Data Mining and Knowledge Discovery, vol. 22, no. 1-2, pp. 31-72, 2011.

[17] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, ”DDoS in the IoT: Mirai and other botnets,”Computer, vol. 50, no. 7, pp. 80-84, 2017.

[18] M. Antonakakis et al., ”Understanding the Mirai botnet,”in Proceedings of the 26th USENIX Security Symposium, Vancouver, BC, Canada, 2017, pp. 1093-1110.

[19] A. L. Buczak and E. Guven, ”A survey of data mining and machine learning methods for cyber security intrusion detection,”IEEE Communications Surveys & Tutorials, vol. 18, no. 2, pp. 1153-1176, 2016.

[20] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, ”Survey of intrusion detection systems: techniques, datasets and challenges,”Cybersecurity, vol. 2, no. 1, pp. 1-22, 2019.

[21] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, ”Deep learning approach for intelligent intrusion detection system,”IEEE Access, vol. 7, pp. 41525-41550, 2019.

[22] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, ”Deep learning approach for network intrusion detection in software defined networking,”in Proceedings of the International Conference on Wireless Networks and Mobile Communications (WINCOM), Fez, Morocco, 2016, pp. 258-263.

[23] Google Colaboratory, ”Google Colab Pro+,”Google Research, 2023. [Online]. Available: https://colab.research.google.com/

[24] NVIDIA Corporation, ”CUDA Toolkit Documentation,”NVIDIA Developer, 2023. [Online]. Available: https://docs.nvidia.com/cuda/

[25] Statista Research Department, ”IoT: Number of connected devices worldwide 2015-2025,”Statista, 2023. [Online]. Available: https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/

[26] A. Gutnikov, O. Kupreev, and Y. Sintsiak, ”DDoS attacks in Q3 2021,”Securelist, Kaspersky, 2021. [Online]. Available: https://securelist.com/ddos-attacks-in-q3-2021/104796/

[27] ”GoldenEye Layer 7 (KeepAlive+NoCache) DoS Test Tool,”GitHub Repository. [Online]. Available: https://github.com/jseidl/GoldenEye

[28] ”Hping – Active Network Security Tool,”2021. [Online]. Available: http://www.hping.org