BỘ KHOA HỌC VÀ CÔNG NGHỆ
HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

---


# BÁO CÁO
# ĐỒ ÁN MÔN HỌC

## ĐỀ TÀI: Phát hiện lưu lượng IoT bất thường bằng Graph Neural Network (GNN) trên CICIDS 2018

**Môn học:** IOT VÀ ỨNG DỤNG

**Giảng viên hướng dẫn:** Th.s Đàm Minh Lịnh

**Thực hiện bởi nhóm sinh viên, bao gồm:**


1. Vũ Ngọc Sơn N22DCCN071 <Trưởng nhóm>
2. Nguyễn Lưu Tấn Sang N22DCCN068 <Thành viên>


**TP.HCM, tháng 1 /2026**

# MỤC LỤC

# PHỤ LỤC HÌNH ẢNH

# PHỤ LỤC BẢNG

# LỜI CẢM ƠN

Trên hành trình hoàn thành đề tài "**Phát hiện lưu lượng IoT bất thường bằng Graph Neural Network (GNN) trên CICIDS 2018**", bên cạnh sự nỗ lực không ngừng của các thành viên trong nhóm, chúng em đã vô cùng may mắn khi nhận được sự hướng dẫn tận tình và sâu sắc từ thầy **Th.s Đàm Minh Lịnh**

Chúng em xin trân trọng gửi lời cảm ơn chân thành nhất đến thầy. Ngay từ những ngày đầu tiếp cận đề tài, khi còn nhiều bỡ ngỡ và khó khăn trong việc định hướng, thầy đã luôn ở bên cạnh, chỉ bảo và dẫn dắt chúng em từng bước. Những buổi góp ý chuyên môn của thầy không chỉ giúp chúng em giải quyết các vấn đề kỹ thuật phức tạp mà còn mở ra những hướng đi mới, khai sáng tư duy và truyền cho chúng em nguồn cảm hứng to lớn để theo đuổi mục tiêu nghiên cứu.

Sự giúp đỡ của thầy là nguồn động viên vô cùng quý báu, giúp nhóm chúng em vững tin hơn để vượt qua những thử thách và hoàn thành đề tài một cách trọn vẹn nhất.

Dù đã nỗ lực hết mình, nhưng với kiến thức và kinh nghiệm thực tiễn còn nhiều hạn chế, bài báo cáo chắc chắn không thể tránh khỏi những thiếu sót. Chúng em kính mong nhận được sự thông cảm và những ý kiến đóng góp quý báu từ thầy để có thể hoàn thiện hơn nữa kiến thức chuyên môn của mình, đồng thời cải thiện chất lượng của đề tài.

Một lần nữa, chúng em xin kính chúc thầy luôn dồi dào sức khỏe, hạnh phúc và gặt hái được nhiều thành công hơn nữa trong sự nghiệp giảng dạy cao quý của mình.

**Chúng em xin chân thành cảm ơn!**

# CHƯƠNG 1: Tổng quan (3–5 trang)

## 1.1 Bối cảnh và vấn đề nghiên cứu

- Thực trạng, thách thức, số liệu, ví dụ minh họa

## 1.2 Tính cấp thiết của đề tài

- Khoảng trống nghiên cứu, hạn chế của các phương pháp hiện tại

## 1.3 Mục tiêu nghiên cứu

### 1.3.1 Mục tiêu tổng quát

Xây dựng hệ thống phát hiện lưu lượng IoT bất thường (IoT anomaly detection) sử dụng Deep Learning và Graph Neural Networks, so sánh hiệu năng của các kiến trúc khác nhau trên dataset CICIDS2018, đồng thời đánh giá khả năng triển khai thực tế trong môi trường production.

### 1.3.2 Mục tiêu cụ thể

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

#### 1.3.2.1 Mục tiêu 1: Xây dựng và đánh giá các mô hình Deep Learning

Cài đặt CNN model cho spatial feature extraction

Đạt được: 82.79% accuracy, 76.01% F1-score

Cài đặt LSTM model cho temporal pattern learning

Đạt được: 87.13% accuracy, 80.77% F1-score

Cài đặt Hybrid CNN-LSTM with Attention mechanism

Target: >85% accuracy, >80% F1-score

Đạt được: 87.48% accuracy, 81.35% F1-score, 93.06% AUC-ROC

#### 1.3.2.2 Mục tiêu 2: Xây dựng mô hình Graph Neural Network [PHẦN ĐỐI TÁC]

Biểu diễn network traffic dưới dạng graph structure

Cài đặt GNN model với message passing

So sánh với CNN/LSTM approaches

#### 1.3.2.3 Mục tiêu 3: So sánh comprehensive các mô hình

Đánh giá theo nhiều metrics: Accuracy, Precision, Recall, F1-Score, AUC-ROC

Phân tích confusion matrix để hiểu classification behavior

So sánh model complexity (số parameters, training time)

Đạt được: Bảng so sánh 5 models với đầy đủ metrics

Mục tiêu 4: Đánh giá real-time performance

Đo latency ở các batch sizes khác nhau

Tính toán throughput (samples/second)

Xác định optimal configuration cho deployment

Đạt được: Latency 0.20ms/sample, throughput 5,027 samples/sec tại batch size 512

#### 1.3.2.4 Mục tiêu 5: Đề xuất giải pháp deployment

Phân tích khả năng triển khai trên các môi trường khác nhau

Đề xuất architecture cho enterprise IDS

Đạt được: Chứng minh production-ready cho sub-Gbps networks

## 1.4 Phạm vi nghiên cứu và đối tượng nghiên cứu

### 1.4.1 Phạm vi nghiên cứu

**a) Về dữ liệu**

**Dataset**: CICIDS2018 (Canadian Institute for Cybersecurity Intrusion Detection System 2018)

**Quy mô**:

- Training set: 2,100,000 samples
- Validation set: 450,000 samples
- Test set: 450,000 samples
- Tổng cộng: 3,000,000 network flow records

**Phân bố class**:

- Benign traffic: 70% (31,344 test samples)
- Attack traffic: 30% (13,656 test samples)

**b) Về features**

- **Số lượng**: 77 network features được trích xuất từ raw packets
- **Loại features**: Flow duration, packet statistics, flags, protocol information
- **Preprocessing**: Min-Max normalization về range

**c) Về task**

- **Problem type**: Binary classification (Benign vs Attack)
- **Giới hạn**: Không phân loại chi tiết từng loại tấn công (multi-class)
- **Lý do**: Tập trung vào detection accuracy và real-time performance

**d) Về models**

- **Deep Learning models**: CNN, LSTM, Hybrid CNN-LSTM with Attention
- **Ensemble methods**: Weighted Voting, Stacking Ensemble
- **Graph Neural Networks**: [PHẦN ĐỐI TÁC]
- **Không bao gồm**: Transformer-based models, Reinforcement Learning

### 1.4.2 Đối tượng nghiên cứu

**a) Network traffic flows**

- Đối tượng chính: Bidirectional network flows trong CICIDS2018
- Mỗi flow được đặc trưng bởi 77 features:
    - Temporal features: Flow Duration, Flow IAT (Inter-Arrival Time)
    - Volume features: Total packets, Total bytes (Forward/Backward)
    - Statistical features: Mean, Std, Max, Min của packet length
    - Flag features: FIN, SYN, RST, PSH, ACK, URG counts
    - Protocol features: TCP/UDP/ICMP information

**b) Sequential patterns**

- Sequence length: 10 timesteps
- Input shape: (batch_size, 10, 77)
- Mô hình hóa traffic như time series data để capture temporal dependencies

**c) Evaluation metrics**

- **Primary metrics**: Accuracy, F1-Score, AUC-ROC
- **Secondary metrics**: Precision, Recall, Confusion Matrix

- **Performance metrics**: Latency (ms/sample), Throughput (samples/sec)

## 1.5 Hướng tiếp cận và phương pháp thực hiện

### 1.5.1 Tổng quan phương pháp

Nghiên cứu áp dụng phương pháp **supervised learning** với **deep neural networks** để phát hiện anomaly trong IoT traffic. Quy trình tổng quát gồm 4 giai đoạn:

CICIDS2018 Raw Data

↓

[1] Data Preprocessing

↓

[2] Model Training (CNN, LSTM, Hybrid, GNN)

↓

[3] Evaluation & Comparison

↓

[4] Deployment Analysis

### 1.5.2 Phương pháp xử lý dữ liệu

**Bước 1: Data Loading và Exploration**

Load preprocessed CICIDS2018 từ stratified split

Verify class balance: 70% Benign, 30% Attack

Check data quality: missing values, outliers

**Bước 2: Feature Normalization**

Áp dụng Min-Max scaling: $xnorm = \frac{x - xmin}{\text{xmax} - \text{xmin}}$

Đảm bảo tất cả features trong range

Lưu scaling parameters để apply cho unseen data

**Bước 3: Sequence Creation**

Reshape từ (samples, 77) thành (sequences, 10, 77)

Sử dụng sliding window approach

Giữ nguyên class balance sau reshaping:

- Train: 147,026 Benign (70.0%), 62,974 Attack (30.0%)
- Validation: 31,571 Benign (70.2%), 13,429 Attack (29.8%)
- Test: 31,387 Benign (69.7%), 13,613 Attack (30.3%)

**Bước 4: Class Weighting**

Tính class weights để xử lý imbalance:

- Class 0 (Benign): weight = 0.714
- Class 1 (Attack): weight = 1.667

Áp dụng trong loss function khi training

### 1.5.3 Phương pháp xây dựng mô hình

#### 1.5.3.1 Approach 1: Convolutional Neural Network (CNN)

**Ý tưởng**: Extract spatial patterns từ network features

**Architecture**: Conv1D layers → BatchNorm → Dropout → Dense

**Kết quả**: 82.79% accuracy baseline

#### 1.5.3.2 Approach 2: Long Short-Term Memory (LSTM)

**Ý tưởng**: Capture temporal dependencies trong traffic sequence

**Architecture**: Bidirectional LSTM → Dense layers

**Kết quả**: 87.13% accuracy, improvement 4.34%

#### 1.5.3.3 Approach 3: Hybrid CNN-LSTM with Attention

**Ý tưởng**: Kết hợp spatial extraction (CNN) + temporal modeling (LSTM) + timestep weighting (Attention)

**Architecture**: 4 components

1. CNN: Conv1D(64) → Conv1D(128) - Spatial features
2. Bidirectional LSTM(64) - Temporal patterns
3. Attention Mechanism - Weighted timesteps
4. Residual Connection - Gradient flow improvement

**Total parameters:** 99,778

**Kết quả:** 87.48% accuracy, 81.35% F1, 93.06% AUC-ROC

#### 1.5.3.4 Approach 4: Graph Neural Network [PHẦN ĐỐI TÁC]

### 1.5.4 Phương pháp đánh giá

#### 1.5.4.1 Classification Metrics

- **Accuracy**: $\text{Acc} = \frac{TP+TN}{TP+TN+FP+FNTP+TN}$
- **Precision**: $\text{Prec} = \frac{TP}{TP+FP}$
- **Recall (Detection Rate)**: $\text{Rec} = \frac{TP}{TP+\text{FN}}$
- **F1Score**: $\text{F1} = 2\text{x}\frac{Prec*Rec}{Prec+Rec}$

#### 1.5.4.2 Chú thích

**TP: True Positive**

Thực tế: Attack

Model dự đoán: Attack

→ ĐÚNG!

Ví dụ: 95 attacks thật, model detect đúng 90

→ TP = 90

**TN: True Negative**

Thực tế: Benign

Model dự đoán: Benign

→ ĐÚNG!

Ví dụ: 905 benign, model nhận đúng 895

→ TN = 895

**FN: False Negative**

Thực tế: Attack

Model dự đoán: Benign

→ SAI! Bỏ sót attack!

Ví dụ: 95 attacks thật, model bỏ sót 5

→ FN = 5 (Nguy hiểm nhất!)

**FP: False Positive**

Thực tế: Benign

Model dự đoán: Attack

→ SAI! Báo động giả!

Ví dụ: 905 benign, model báo nhầm 10 là attack

→ FP = 10 (Phiền nhưng ít nguy hiểm hơn FN)

**Precision (Độ chính xác dương):**

Model dự đoán 100 packets là Attack

Trong đó:

- 90 thật sự là Attack (TP)

- 10 thật ra là Benign (FP - báo nhầm)

Precision = 90 / (90 + 10) = 90%

**→ Khi model báo "Attack", đúng 90% trường hợp**

**Recall (Độ phủ / Sensitivity):**

Có 100 attacks thật trong test set

Model detect được:

- 90 attacks (TP)

- Bỏ sót 10 attacks (FN)

Recall = 90 / (90 + 10) = 90%

**→ Model phát hiện được 90% attacks**

**F1-SCORE (Điểm cân bằng):**

Ví dụ 1: Balance tốt

Precision = 90%, Recall = 90%

F1 = 2 × (0.9 × 0.9) / (0.9 + 0.9) = 0.90

Ví dụ 2: Lệch nhiều

Precision = 95%, Recall = 50%

F1 = 2 × (0.95 × 0.5) / (0.95 + 0.5) = 0.655

**→ Harmonic mean "phạt" nếu 1 trong 2 thấp**

**→ Buộc model phải CÂN BẰNG cả 2!**

**AUC-ROC (Area Under ROC Curve):**

**ROC Curve (Receiver Operating Characteristic):**

ROC là biểu đồ mối quan hệ giữa:

- Trục Y: True Positive Rate (TPR) = Recall = TP/(TP+FN)
- Trục X: False Positive Rate (FPR) = FP/(FP+TN)

Vẽ ở nhiều threshold khác nhau (0.1, 0.2, ..., 0.9)

**AUC = Diện tích dưới ROC curve**

Ví dụ:

- Model xuất xác suất: [0.2, 0.5, 0.8, 0.95, ...]
- Threshold = 0.5: Prob > 0.5 → Attack, Prob < 0.5 → Benign
- Threshold = 0.3: Dễ báo attack hơn
- Threshold = 0.8: Khó báo attack hơn

**Phân loại AUC:**

- 0.90 - 1.00: Outstanding discrimination
- 0.80 - 0.90: Excellent discrimination
- 0.70 - 0.80: Good discrimination
- 0.50 - 0.60: Poor (gần random guess)

→ AUC-ROC đo khả năng PHÂN BIỆT tổng thể của model, không phụ thuộc vào 1 threshold cụ thể

**Accuracy (Độ chính xác tổng thể):**

Ví dụ: Test set 10,000 samples

- TP = 950 (Detect đúng attacks)
- TN = 8,900 (Nhận đúng benign)
- FP = 100 (Báo nhầm)
- FN = 50 (Bỏ sót)

Accuracy = (950 + 8,900) / 10,000 = 98.5%

→ Model dự đoán đúng 98.5% tổng số samples

**Hạn chế:** Với imbalanced dataset (90% benign, 10% attack), accuracy có thể cao nhưng model vẫn kém

## 1.6 Ý nghĩa khoa học và thực tiễn

## 1.7 Cấu trúc báo cáo

- Tóm tắt nội dung các chương

# CHƯƠNG 2: Mô hình đề xuất

## 2.1 Kiến trúc tổng thể và sơ đồ logic hệ thống

### 2.1.1 Tổng quan về hệ thống phát hiện xâm nhập

Nghiên cứu này đề xuất hệ thống phát hiện lưu lượng IoT bất thường sử dụng hai approaches chính:

1. **Approach 1: Deep Learning Baseline (CNN-LSTM)** - Phương pháp truyền thống sử dụng sequential modeling
2. **Approach 2: Graph Neural Network (GNN)** [PHẦN CHÍNH - ĐỐI TÁC VIẾT] - Phương pháp đề xuất exploit graph structure của network traffic

**Mục tiêu so sánh**: Đánh giá hiệu quả của GNN so với phương pháp Deep Learning truyền thống trên cùng dataset CICIDS2018.

### 2.1.2 Phân tích các thuộc tính (features) của CICIDS 2018

Sau bước làm sạch dữ liệu, mỗi bản ghi (network flow) trong dataset CICIDS 2018 được biểu diễn bằng 77 thuộc tính đầu vào và một cột Label (Benign/Attack). Các thuộc tính này được trích xuất ở mức flow bằng công cụ CICFlowMeter, bao phủ đầy đủ thông tin về kích thước gói tin, thời gian, cờ TCP và hành vi truyền nhận theo hai hướng forward (client → server) và backward (server → client).

Để thuận tiện cho việc phân tích và thiết kế mô hình CNN-LSTM, toàn bộ 77 feature được chia thành 7 nhóm chính như sau.

#### 2.1.2.1 Nhóm đặc trưng chung của flow

Các thuộc tính mô tả đặc điểm tổng quát nhất của một phiên kết nối:

- **Protocol**

  Giao thức tầng transport được sử dụng trong flow (TCP, UDP, ICMP, …). Protocol giúp mô hình phân biệt các loại traffic cơ bản và là "ngữ cảnh" cho việc giải thích các thuộc tính khác (ví dụ: ý nghĩa TCP flags chỉ đúng với TCP).

- **Flow_Duration**

  Thời gian tồn tại của flow tính theo micro-giây (μs). Flow bình thường như truy cập web, tải file thường có duration nằm trong khoảng ổn định; ngược lại, một

số tấn công (DoS, scan) tạo ra nhiều flow rất ngắn hoặc cực dài nhưng lưu lượng nhỏ, làm lệch phân bố của thuộc tính này.

#### 2.1.2.2 Nhóm số lượng packet và tổng byte theo hai chiều

Nhóm này mô tả **khối lượng dữ liệu** gửi/nhận trong flow:

- **Tot_Fwd_Pkts, Tot_Bwd_Pkts**
  Tổng số packet theo chiều forward và backward. Các tấn công như UDP flood hoặc SYN flood thường tạo ra số packet cực lớn ở một chiều (thường là forward), trong khi chiều còn lại rất ít hoặc không có phản hồi.
- **TotLen_Fwd_Pkts, TotLen_Bwd_Pkts**
  Tổng số byte (payload + header) của tất cả packet theo hai chiều. Lưu lượng tải file, streaming thường có TotLen_Bwd_Pkts lớn hơn nhiều TotLen_Fwd_Pkts (client gửi ít request, server trả nhiều dữ liệu), còn brute force hay scan ngược lại: phần lớn byte nằm ở phía client.

Các feature này là **cơ sở để mô hình học được pattern tải/nhận dữ liệu**: cân đối hay lệch một chiều, tổng khối lượng lớn hay nhỏ, từ đó phân biệt traffic bình thường với các kịch bản tấn công.

#### 2.1.2.3 Thống kê kích thước packet theo từng hướng

Nhóm này mô tả **phân bố kích thước** của packet forward và backward:

- **Fwd_Pkt_Len_Max, Fwd_Pkt_Len_Min, Fwd_Pkt_Len_Mean, Fwd_Pkt_Len_Std**
  Giá trị lớn nhất, nhỏ nhất, trung bình và độ lệch chuẩn kích thước packet ở chiều forward. Forward traffic do script hoặc bot sinh ra thường có kích thước gần như cố định (Std nhỏ), trong khi tương tác người dùng thật (web, video) tạo ra kích thước đa dạng hơn.
- **Bwd_Pkt_Len_Max, Bwd_Pkt_Len_Min, Bwd_Pkt_Len_Mean, Bwd_Pkt_Len_Std**
  Tương tự cho chiều backward. Ví dụ, trong tấn công brute force, phía server chủ yếu trả về thông báo lỗi với payload ngắn, dẫn đến Bwd_Pkt_Len_Max và Mean thấp một cách bất thường.

#### 2.1.2.4 Nhóm tốc độ và mật độ lưu lượng

Các thuộc tính đo **tốc độ gửi/nhận**:

- **Flow_Byts/s, Flow_Pkts/s**
  Số byte trên giây và số packet trên giây của toàn bộ flow. Các tấn công flooding (DoS/DDoS) thường tạo ra giá trị Flow_Byts/s hoặc Flow_Pkts/s rất cao trong thời gian ngắn, trong khi nhiều phiên tương tác bình thường có tốc độ thấp hơn và ổn định hơn.
- **Fwd_Pkts/s, Bwd_Pkts/s**
  Tốc độ packet riêng cho chiều forward và backward. Sự mất cân đối lớn giữa hai giá trị này là dấu hiệu mạnh của các cuộc tấn công một chiều (client spam request mà server hầu như không trả lời, hoặc ngược lại).

#### 2.1.2.5 Nhóm Inter-Arrival Time (IAT) – hành vi theo thời gian

IAT mô tả khoảng cách thời gian giữa các packet liên tiếp:

- **Flow_IAT_Mean, Flow_IAT_Std, Flow_IAT_Max, Flow_IAT_Min**
  Thống kê khoảng cách giữa mọi packet trong flow. Traffic sinh bởi người dùng thường có IAT biến thiên tự nhiên; ngược lại, traffic tự động (botnet, script) có thể có IAT rất đều hoặc rất nhỏ, khiến Std thấp và Min gần 0.
- **Fwd_IAT_Tot, Fwd_IAT_Mean, Fwd_IAT_Std, Fwd_IAT_Max, Fwd_IAT_Min**
  Tổng, trung bình, độ lệch chuẩn, lớn nhất, nhỏ nhất IAT ở chiều forward.
- **Bwd_IAT_Tot, Bwd_IAT_Mean, Bwd_IAT_Std, Bwd_IAT_Max, Bwd_IAT_Min**
  Tương tự cho chiều backward.

#### 2.1.2.6 Nhóm TCP flags và thông tin điều khiển

Nhóm này tập trung vào các cờ điều khiển ở TCP header:

- **Fwd_PSH_Flags, Bwd_PSH_Flags**
  Số packet forward/backward mang cờ PSH (push). PSH thường được dùng để

buộc gửi dữ liệu lên ứng dụng ngay; mật độ PSH cao bất thường có thể chỉ ra traffic ứng dụng bất thường.

- **Fwd_URG_Flags, Bwd_URG_Flags**
  Số packet có URG flag. Trong traffic bình thường URG rất hiếm, nên số lượng URG lớn là dấu hiệu đáng nghi.
- **FIN_Flag_Cnt, SYN_Flag_Cnt, RST_Flag_Cnt, PSH_Flag_Cnt, ACK_Flag_Cnt, URG_Flag_Cnt, CWE_Flag_Count, ECE_Flag_Cnt**
  Số lần các cờ TCP FIN, SYN, RST, PSH, ACK, URG, CWE, ECE xuất hiện trong flow.
  - SYN_Flag_Cnt cao, trong khi ACK_Flag_Cnt thấp thường gắn với SYN flood hoặc half-open scan.
  - RST_Flag_Cnt cao có thể gợi ý việc reset kết nối hàng loạt.
  - ECE/CWE liên quan tới congestion control; sự thay đổi bất thường của chúng có thể xuất hiện trong các kịch bản tấn công kiểu protocol-aware.

#### 2.1.2.7 Nhóm đặc trưng kích thước packet tổng quát

Nhóm này gom các thống kê không phân biệt hướng:

- **Pkt_Len_Min, Pkt_Len_Max, Pkt_Len_Mean, Pkt_Len_Std, Pkt_Len_Var**
  Min, max, trung bình, độ lệch chuẩn và phương sai kích thước packet trên toàn bộ flow.
- **Pkt_Size_Avg**
  Kích thước packet trung bình (gần tương đương Pkt_Len_Mean nhưng được tính theo cách của công cụ CICFlowMeter).

#### 2.1.2.8 Nhóm header length, segment size và cửa sổ TCP

- **Fwd_Header_Len, Bwd_Header_Len**
  Tổng số byte phần header của tất cả packet theo từng chiều. Những flow có phần payload hầu như bằng 0 nhưng header vẫn xuất hiện dày đặc (header-only flows) thường liên quan tới scan hoặc handshake giả.
- **Fwd_Seg_Size_Avg, Bwd_Seg_Size_Avg**
  Kích thước segment TCP trung bình ở mỗi chiều. Đây là thông tin quan trọng cho việc phân biệt traffic interactive (nhiều segment nhỏ) với traffic tải file (segment cỡ gần MTU).

- **Fwd_Seg_Size_Min**
  Kích thước segment nhỏ nhất ở chiều forward; kết hợp với các thống kê khác, nó giúp nhận diện các phiên có nhiều gói control (ví dụ chỉ chứa ACK) so với gói mang dữ liệu.
- **Init_Fwd_Win_Byts, Init_Bwd_Win_Byts**
  Giá trị initial window size của TCP ở hai chiều. Tùy biến bất thường của window size có thể được dùng như một kỹ thuật né tránh hoặc tấn công vào cơ chế kiểm soát tắc nghẽn.

#### 2.1.2.9 Nhóm tỉ lệ và thông số "Byts/b", "Pkts/b", "Blk_Rate"

Đây là các feature chỉ có trong CICFlowMeter, mô tả các tỉ lệ và tốc độ tính theo block:

- **Down/Up_Ratio**

  Tỉ lệ lưu lượng download so với upload (backward/forward). Trong phiên duyệt web thông thường, Down/Up_Ratio thường > 1 (server gửi nhiều dữ liệu hơn client), còn trong tấn công scan hoặc brute force, tỉ lệ này có thể gần 0 hoặc rất thấp.
- **Fwd_Byts/b_Avg, Fwd_Pkts/b_Avg, Fwd_Blk_Rate_Avg**

  Các giá trị trung bình byte, packet và block rate cho chiều forward.
- **Bwd_Byts/b_Avg, Bwd_Pkts/b_Avg, Bwd_Blk_Rate_Avg**

  Tương tự cho chiều backward.

#### 2.1.2.10 Nhóm Subflow – hành vi theo phân đoạn

- **Subflow_Fwd_Pkts, Subflow_Fwd_Byts**
  Số packet và số byte forward trong mỗi subflow (phân đoạn con của flow).
- **Subflow_Bwd_Pkts, Subflow_Bwd_Byts**
  Tương tự cho chiều backward.

#### 2.1.2.11 Nhóm trạng thái Active/Idle của flow

Thuộc tính mô tả **chu kỳ hoạt động – nghỉ của flow**:

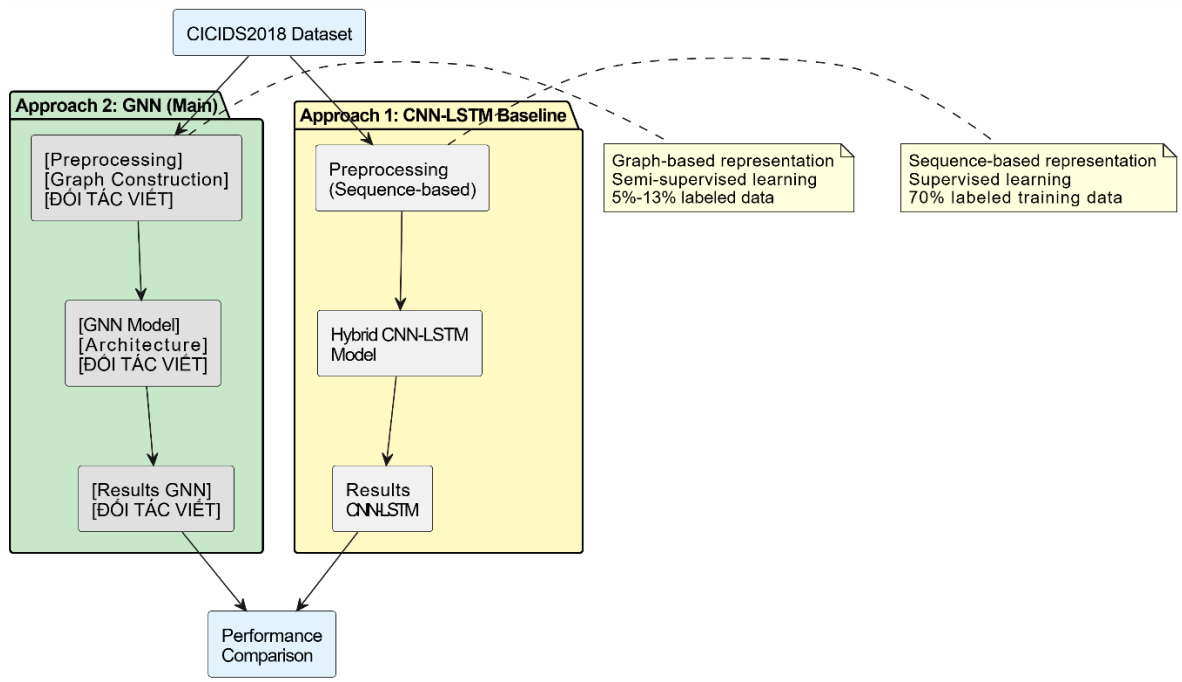
- **Active_Mean, Active_Std, Active_Max, Active_Min**

  Thống kê độ dài các khoảng thời gian mà flow "active" (có packet được gửi). Flow bình thường thường có nhiều khoảng active ngắn xen kẽ idle; các tấn công flooding hoặc brute force thường có chuỗi active dài liên tục.

- **Idle_Mean, Idle_Std, Idle_Max, Idle_Min**

  Thống kê độ dài khoảng thời gian flow "idle" (không có packet). Các giá trị idle cực dài xen kẽ burst ngắn có thể gợi ý traffic do bot điều khiển, trong khi idle gần như bằng 0 xuyên suốt thường là dấu hiệu của tấn công volumetric.

### 2.1.3 Sơ đồ tổng quan hệ thống (PHẦN CHUNG)




*Hình 1 Sơ đồ tổng quát hệ thống*

Hình sai mai mốt nhét lại

*Bảng 1 Bảng so sánh nhanh giữa CNN-LSTM và GNN*

| Aspect | CNN-LSTM Baseline | GNN (Main Approach) |
|---|---|---|
| **Data Representation** | Sequential (10 timesteps, 77 features) | [Graph nodes/edges - ĐỐI TÁC] |
| **Learning Paradigm** | Supervised (70% labeled) | [Semi-supervised 5%-13% - ĐỐI TÁC] |
| **Preprocessing** | Normalization + Sequencing | [Graph construction - ĐỐI TÁC] |
| **Architecture** | CNN → LSTM → Attention | [GNN layers - ĐỐI TÁC] |
| **Training Rounds** | 1 round (standard training) | [5 rounds progressive - ĐỐI TÁC] |

## 2.2 Approach 1: Mô hình CNN-LSTM Baseline

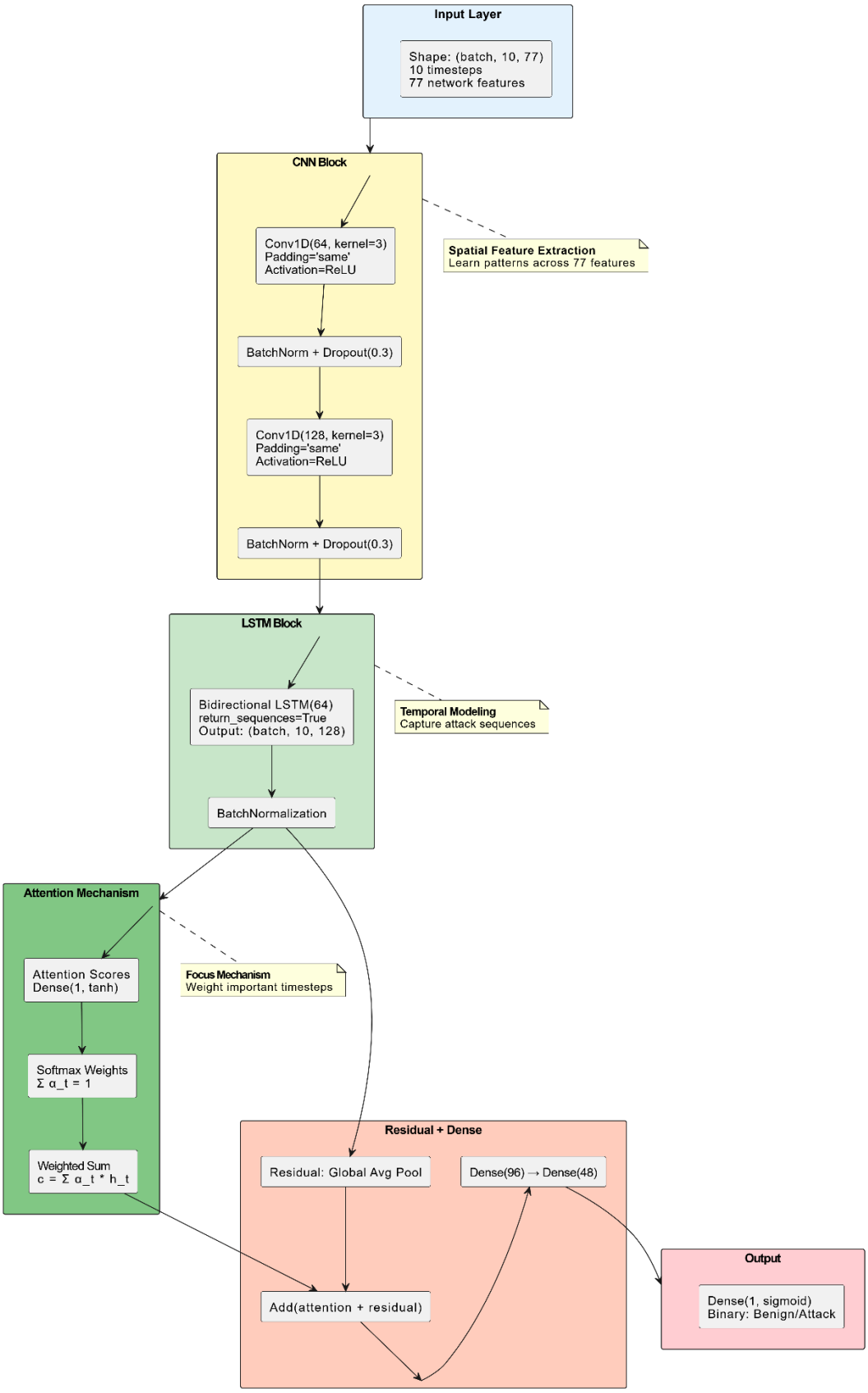
### 2.2.1 Lý do chọn CNN-LSTM làm baseline

CNN-LSTM được chọn làm baseline vì:

1. **State-of-the-art cho sequence data**: Kết hợp spatial (CNN) và temporal (LSTM) learning
2. **Widely adopted**: Nhiều papers về IDS sử dụng approach này
3. **Strong performance**: Đạt 87.48% accuracy trên CICIDS2018
4. **Fair comparison**: Cùng dataset, cùng evaluation metrics với GNN

**Limitation của CNN-LSTM** (động lực cho GNN):

- Không exploit graph structure của network (connections giữa IPs)
- Cần nhiều labeled data (70% training set)
- Xử lý sequential, không capture complex network topology




*Hình 2 Sơ đồ kiến trúc chi tiết Hybrid CNN-LSTM*

### 2.2.2 Các thuật toán chính

#### 2.2.2.1 Algorithm 1: Data Preprocessing cho CNN-LSTM

Input: CICIDS2018 raw data (3M samples × 77 features)

Output: X_train_seq (210K × 10 × 77), y_train_seq (210K,)

ALGORITHM: CNN_LSTM_Preprocessing

1. Load dataset:

   X_train (2.1M, 77), y_train (2.1M,)

   X_val (450K, 77), y_val (450K,)

   X_test (450K, 77), y_test (450K,)

2. Verify class balance:

   benign_ratio ← COUNT(y == 0) / TOTAL

   ASSERT benign_ratio ≈ 0.70

3. Shuffle data:

   idx ← RANDOM_PERMUTATION(len(X_train), seed=42)

   X_train ← X_train[idx]

   y_train ← y_train[idx]

4. Normalize features:

   scaler ← MinMaxScaler()

   X_train ← scaler.FIT_TRANSFORM(X_train)

   X_val ← scaler.TRANSFORM(X_val)

   X_test ← scaler.TRANSFORM(X_test)

5. Create sequences (timesteps=10):

FOR each set in {train, val, test}:

n_samples ← len(X) - (len(X) MOD 10)

X ← X[0:n_samples]

y ← y[0:n_samples]

n_seq ← n_samples / 10

X_seq ← RESHAPE(X, (n_seq, 10, 77))

y_reshaped ← RESHAPE(y, (n_seq, 10))

y_seq ← y_reshaped[:, -1]  # Take last label

VERIFY CLASS_BALANCE(y_seq) ≈ 0.70

END FOR

6. Compute class weights:

w_benign ← n_total / (2 × n_benign)

w_attack ← n_total / (2 × n_attack)

class_weights ← {0: w_benign, 1: w_attack}

7. Return X_seq, y_seq, class_weights

**Kết quả preprocessing**

Original shapes:

- Train: (2,100,000, 77)
- Val:  (450,000, 77)
- Test:  (450,000, 77)

After sequencing:

- Train: (210,000, 10, 77), Benign: 70.0%
- Val: (45,000, 10, 77), Benign: 70.2%
- Test: (45,000, 10, 77), Benign: 69.7%

Class weights: {0: 0.714, 1: 1.667}

**Files created:**

X_train.npy (616.8 MB)

y_train.npy (8.0 MB)

X_val.npy (132.2 MB)

y_val.npy (1.7 MB)

X_test.npy (132.2 MB)

y_test.npy (1.7 MB)

scaler.pkl

feature_names.pkl

#### 2.2.2.2 Algorithm 2: Hybrid CNN-LSTM Forward Pass

Input: $x \in \mathbb{R}^{(batch \times 10 \times 77)}$

Output: $\hat{y} \in [0, 1]^{batch}$

ALGORITHM: Forward_CNN_LSTM

1. CNN spatial extraction:

$h_1 \leftarrow$ ReLU(Conv1D_64(x)) # (batch, 10, 64)

$h_1 \leftarrow$ BatchNorm($h_1$)

$h_1 \leftarrow$ Dropout($h_1$, p=0.3)

$h_2 \leftarrow$ ReLU(Conv1D_128($h_1$))  # (batch, 10, 128)

$h_2 \leftarrow$ BatchNorm($h_2$)

$h_2 \leftarrow$ Dropout($h_2$, p=0.3)

2. Bidirectional LSTM temporal modeling:

FOR t = 1 to 10:

h_fwd[t] ← LSTM_forward($h_2$[t], h_fwd[t-1])

h_bwd[t] ← LSTM_backward($h_2$[t], h_bwd[t+1])

h_lstm[t] ← CONCAT(h_fwd[t], h_bwd[t])  # (batch, 128)

END FOR

h_lstm ← BatchNorm(h_lstm)

3. Attention mechanism:

FOR t = 1 to 10:

e_t ← tanh(W_a · h_lstm[t] + b_a)  # Attention score

END FOR

α ← SOFTMAX([$e_1$, $e_2$, ..., $e_{10}$])  # Attention weights

c_attn ← Σ(α_t × h_lstm[t])  # Context vector

4. Residual connection:

c_res ← GLOBAL_AVG_POOL(h_lstm)

c_combined ← c_attn + c_res

5. Dense classification:

$z_1 \leftarrow$ ReLU(Dense_96(c_combined))

$z_1 \leftarrow \text{BatchNorm}(z_1)$

$z_1 \leftarrow \text{Dropout}(z_1, p=0.3)$

$z_2 \leftarrow \text{ReLU}(\text{Dense\_48}(z_1))$

$z_2 \leftarrow \text{Dropout}(z_2, p=0.25)$

$\hat{y} \leftarrow \sigma(\text{Dense\_1}(z_2))$ # Sigmoid output

6. Return $\hat{y}$

### 2.2.3 Mô tả chi tiết quy trình hoạt động

**Step 0: Input Preparation**

- Input: Network flow sequence (10 timesteps, 77 features)
- Features: Flow duration, packet counts, TCP flags, IAT statistics, etc.
- Label: Binary (0=Benign, 1=Attack)

**Step 1: Spatial Feature Extraction (CNN)**

Mục tiêu: Extract local patterns từ 77 network features.

- **Conv1D Layer 1**: 64 filters, kernel size 3
    - Scan across features với sliding window size 3
    - Output: 64 feature maps
    - Example: Detect patterns như "high packet count + low duration + many SYN flags" → potential SYN flood
- **Conv1D Layer 2**: 128 filters, kernel size 3
    - Combine low-level features thành high-level abstractions
    - Output: 128 feature maps
    - Example: Combine multiple attack indicators

**Step 2: Temporal Pattern Learning (LSTM)**

Mục tiêu: Capture attack evolution qua 10 timesteps.

- **Bidirectional LSTM**: Process sequence forward và backward
    - Forward: $t_1 \rightarrow t_2 \rightarrow ... \rightarrow t_{10}$ (attack progression)
    - Backward: $t_{10} \rightarrow t_9 \rightarrow ... \rightarrow t_1$ (attack context)
    - Concat both directions: (batch, 10, 128)

Example temporal pattern:

text

$t_1$: Normal traffic

$t_2$: Slight increase in packets

$t_3$-$t_5$: Gradual buildup (attack preparation)

$t_6$-$t_{10}$: Burst of traffic (DDoS attack)

**Step 3: Attention Weighting**

Mục tiêu: Focus vào critical timesteps.

- Compute attention scores cho mỗi timestep
- Normalize bằng softmax
- Weighted sum: Important timesteps có weight cao hơn

Example attention weights:

text

$\alpha_1 = 0.05, \alpha_2 = 0.08, \alpha_3 = 0.10,$

$\alpha_4 = 0.12, \alpha_5 = 0.15, \alpha_6 = 0.18,$ ← Attack starts

$\alpha_7 = 0.14, \alpha_8 = 0.10, \alpha_9 = 0.05, \alpha_{10} = 0.03$

**Step 4: Residual Connection**

Mục tiêu: Improve gradient flow, prevent vanishing gradient.

- Global average pooling của all timesteps
- Add with attention output
- Better training stability

**Step 5: Classification**

- Dense layers: 128 → 96 → 48 → 1
- Sigmoid activation: Output probability
- Threshold 0.5: Benign if < 0.5, Attack if ≥ 0.5
- **2.2.5. Công cụ và công nghệ**

**Hardware**:

- GPU: 2× Tesla T4 (13.9 GB VRAM each)
- CPU: Intel Xeon
- RAM: 30 GB

**Software Stack**:

TensorFlow: 2.18.0

Keras: Built-**in**

CUDA: 11.8

cuDNN: 8.6 (CuDNN-optimized LSTM)

Mixed Precision: float16/float32

**Hyperparameters**:

*Bảng 2 Bảng tóm tắt các Siêu tham Số (Hyperparameters)*

| **Parameter** | **Value** | **Justification** |
|---|---|---|
| Sequence length | 10 | Capture short-term attack patterns |
| Batch size | 512 | Optimal cho Tesla T4 GPU |
| Learning rate | 0.001 | Adam optimizer default |
| Epochs | 30 | With early stopping |
| Dropout | 0.3, 0.4, 0.25 | Prevent overfitting |
| Class weights | {0: 0.714, 1: 1.667} | Handle imbalance |

### 2.2.4 Phân tích độ phức tạp CNN-LSTM

**Time Complexity(Độ phức tạp thời gian)**

*Bảng 3 Bảng so sánh độ phức tạp thời gian*

| Component | Training | Inference |
|---|---|---|
| Conv1D layers | O(N × T × F × K × C) | O(T × F × K × C) |
| LSTM layers | O(N × T × $H^2$) | O(T × $H^2$) |
| Attention | O(N × T × H) | O(T × H) |
| Dense layers | O(N × H × D) | O(H × D) |

**Trong đó:**

N=batch size, T=10 timesteps, F=77 features, K=3 kernel, C=64/128 channels, H=64/128 hidden, D=96/48 dense units

**Space Complexity(Độ phức tạp không gian)**

- Parameters: 99,778
- Memory footprint: ~0.38 MB
- GPU VRAM usage: ~2 GB (batch=512)

**Inference Performance(Hiệu năng thực hiện)**

- Latency: 0.20 ms/sample (batch=512)
- Throughput: 5,027 samples/sec
- Real-time capable: ✓ (sub-Gbps networks)

## 2.3 Approach 2: Mô hình Graph Neural Network - ĐỐI TÁC VIẾT

### 2.3.1 Biểu diễn network traffic dưới dạng graph

## 2.4 Công cụ, công nghệ, nền tảng triển khai

### 2.4.1 Hardware Environment

GPU: 2× Tesla T4

- VRAM: 13.9 GB each
- CUDA Cores: 2,560
- Tensor Cores: 320
- Compute Capability: 7.5

CPU: Intel Xeon (multi-core)

RAM: 30 GB DDR4

Disk: 73 GB SSD

Platform: Kaggle Notebook

### 2.4.2 Software Stack

**CNN-LSTM**:

TensorFlow 2.18.0

Keras (built-in)

NumPy 1.24.3

scikit-learn 1.3.0

**GNN [ĐỐI TÁC]**:

[PyTorch / PyTorch Geometric]

[DGL (Deep Graph Library)]

[NetworkX]

# CHƯƠNG 3: Thực nghiệm và thảo luận (15-20 trang)

## 3.1 Môi trường thực nghiệm

### 3.1.1 Cấu hình phần cứng và phần mềm

#### 3.1.1.1 Hardware (như đã mô tả ở Chương 2.4)

#### 3.1.1.2 Dataset Configuration:

**Đối với CNN-LSTM**

CICIDS2018 Network Intrusion Detection Dataset

Total samples: 3,000,000 network flows
Features: 77 network flow characteristics
Classes: 2 (Benign, Attack)

Distribution:
- Benign: 70% (~2.1M samples)
- Attack: 30% (~0.9M samples)

Attack types trong dataset:
- DDoS attacks
- DoS attacks (Slowloris, Hulk, GoldenEye)
- Brute Force (FTP, SSH)
- Web attacks (SQL Injection, XSS)
- Infiltration
- Botnet traffic

**Đối với GNN**

### 3.1.2 Dataset split strategies

**Strategy 1: CNN-LSTM (Supervised)**

Training: 2,100,000 samples (70%) - Fully labeled

Validation: 450,000 samples (15%) - Fully labeled

Test: 450,000 samples (15%) - Fully labeled

Class balance maintained:

- Train: 70.0% Benign, 30.0% Attack
- Val: 70.0% Benign, 30.0% Attack
- Test: 70.0% Benign, 30.0% Attack

**Strategy 2: GNN (Semi-supervised) [ĐỐI TÁC VIẾT]**

[ĐỐI TÁC MÔ TẢ]

Total: 3,000,000 samples (considered unlabeled initially)

Progressive labeling approach:

Round 1: 5% labeled (150K), 95% unlabeled (2.85M)

Round 2: 7% labeled (210K), 93% unlabeled (2.79M)

Round 3: 9% labeled (270K), 91% unlabeled (2.73M)

Round 4: 11% labeled (330K), 89% unlabeled (2.67M)

Round 5: 13% labeled (390K), 87% unlabeled (2.61M)

[Chi tiết cách chọn labeled samples]

[Chi tiết evaluation protocol]

## 3.2 Kết quả thực nghiệm - CNN-LSTM Baseline

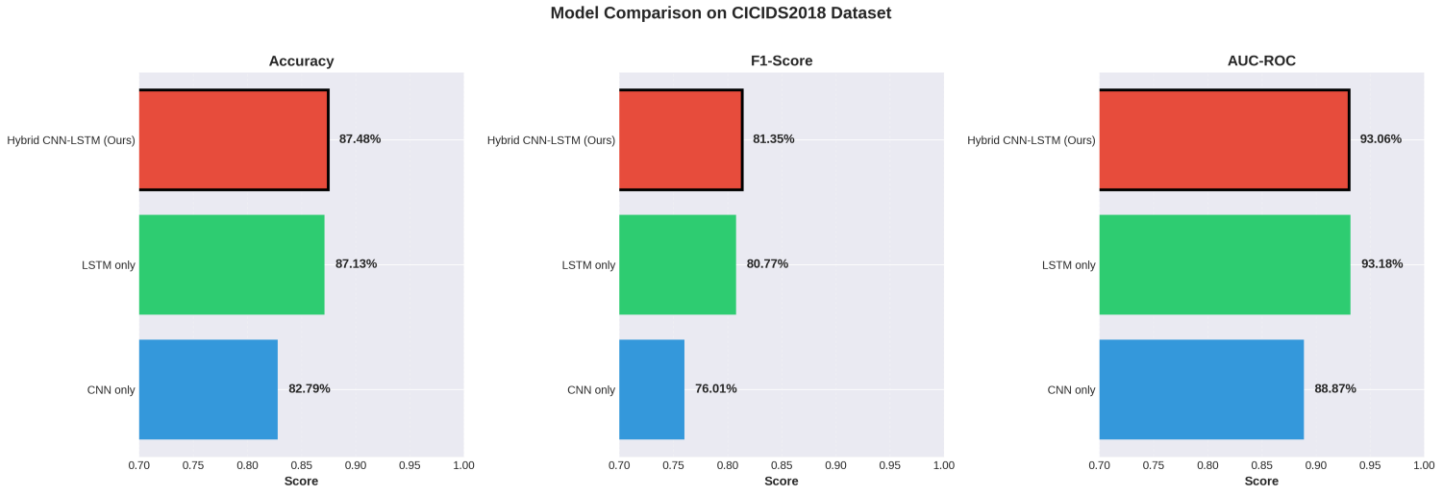
### 3.2.1 Performance Metrics

*Bảng 4 Bảng kết quả chi tiết*

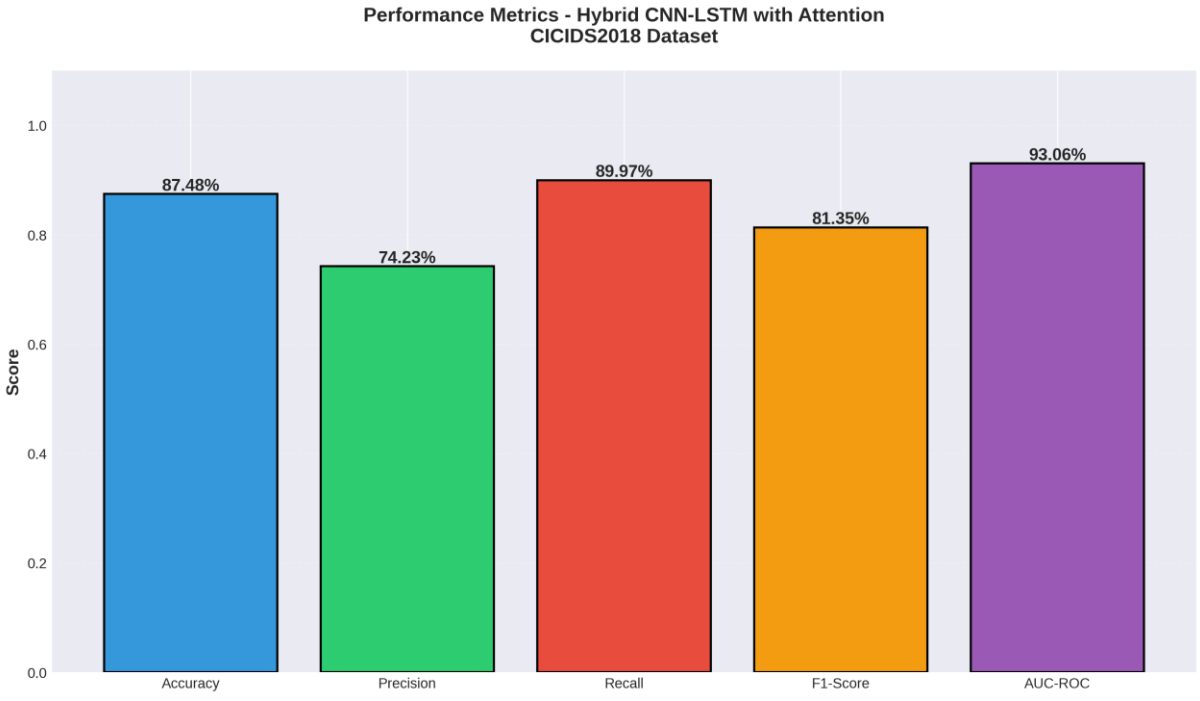
| Model | Accuracy | Precision | Recall | F1-Score | AUC-ROC |
|---|---|---|---|---|---|
| CNN only | 82.79% | 81.08% | 71.09% | 76.01% | 88.87% |
| LSTM only | 87.13% | 75.58% | 86.66% | 80.77% | 93.18% |

| Model | Accuracy | Precision | Recall | F1-Score | AUC-ROC |
|---|---|---|---|---|---|
| **Hybrid CNN-LSTM** | **87.48%** | **74.23%** | **89.97%** | **81.35%** | **93.06%** |

**Biểu đồ so sánh models**




*Hình 3 Biểu đồ so sánh giữa các model*




*Hình 4 Các chỉ số hiệu năng của mô hình Hybrid CNN-LSTM with Attention trên tập dữ liệu CICIDS2018*

**Hybrid CNN-LSTM chi tiết**

Test Set: 45,000 sequences

- Benign: 31,344 (69.65%)
- Attack: 13,656 (30.35%)

Classification Results:

Accuracy: 87.48% (39,365/45,000 correct)

Precision: 74.23% (12,286/16,551 predicted attacks are true)

Recall: 89.97% (12,286/13,656 actual attacks detected)

F1-Score: 81.35%

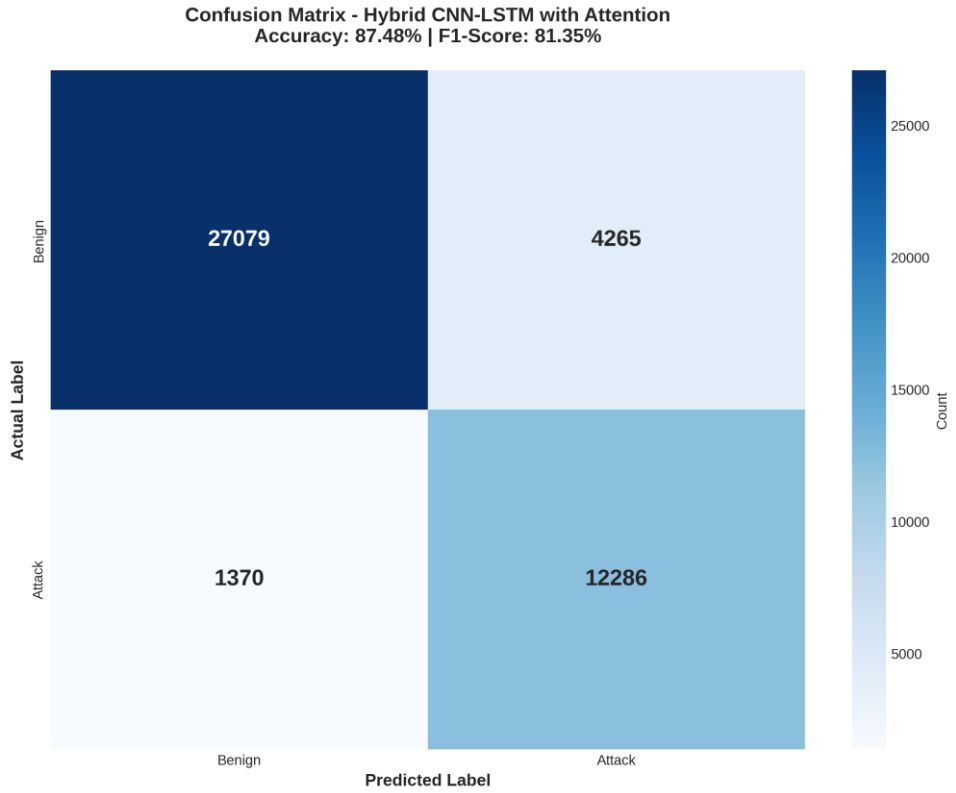
AUC-ROC: 93.06%

Confusion Matrix:

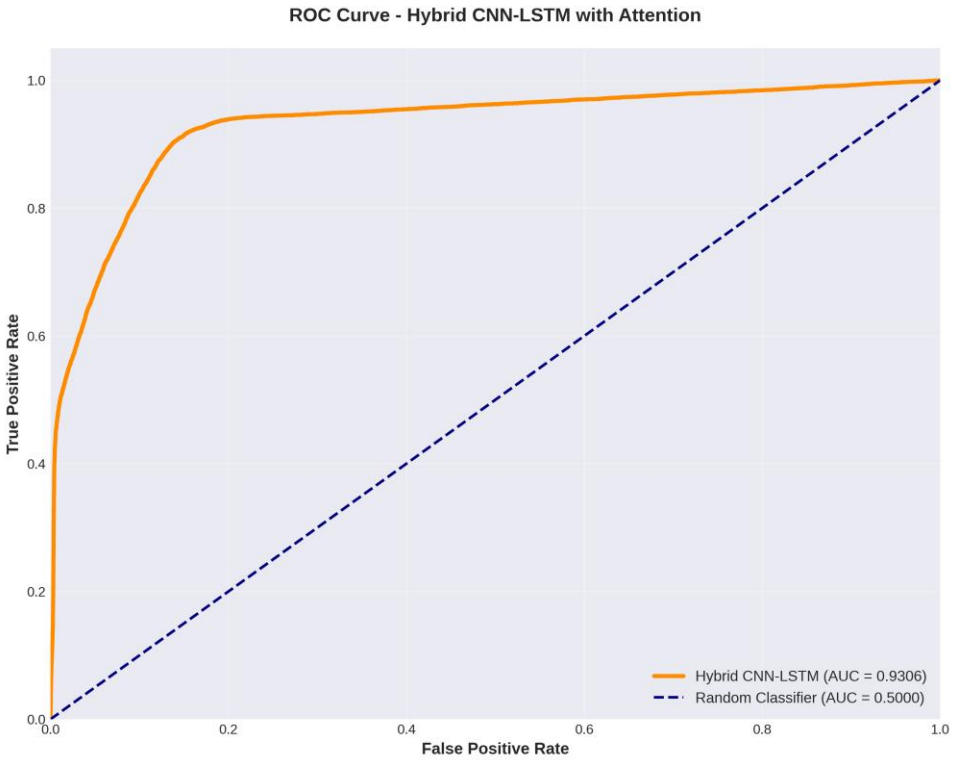
| | Predicted | | |
|---|---|---|---|
| | Benign | Attack | Total |
| Actual | | | |
| Benign | 27,079 | 4,265 | 31,344 |
| Attack | 1,370 | 12,286 | 13,656 |
| Total | 28,449 | 16,551 | 45,000 |

Error Analysis:

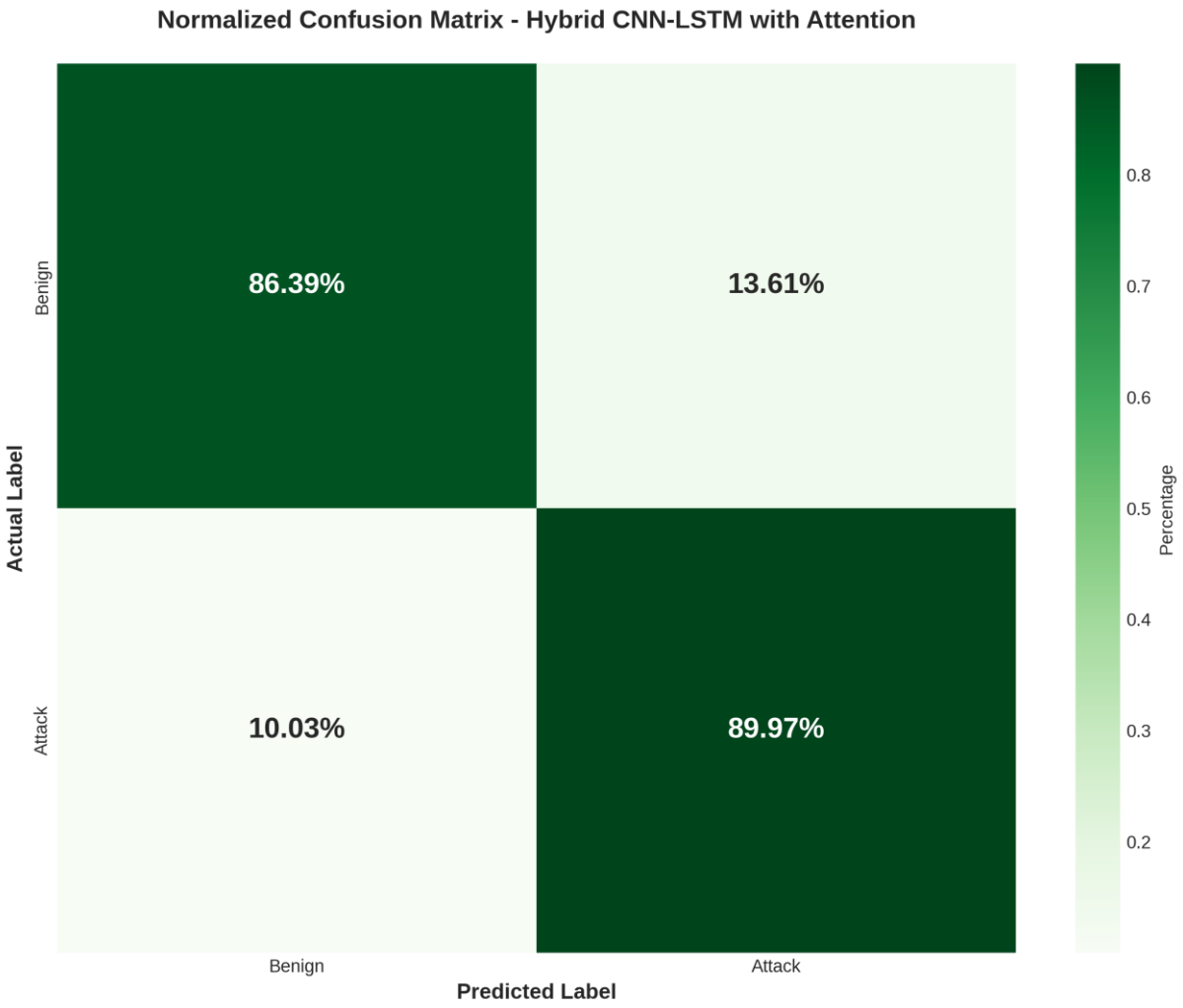
- True Negatives: 27,079 (86.39% specificity)
- False Positives: 4,265 (13.61% FPR)
- False Negatives: 1,370 (10.03% FNR)
- True Positives: 12,286 (89.97% recall)




*Hình 5 Ma trận nhầm lẫn của mô hình Hybrid CNN-LSTM with Attention*




*Hình 6 Đường cong ROC của mô hình Hybrid CNN-LSTM with Attention (AUC = 0.9306)*




*Hình 7 Ma trận nhầm lẫn chuẩn hóa của mô hình Hybrid CNN-LSTM with Attention (phần trăm)*

### 3.2.2 Real-time Performance Analysis

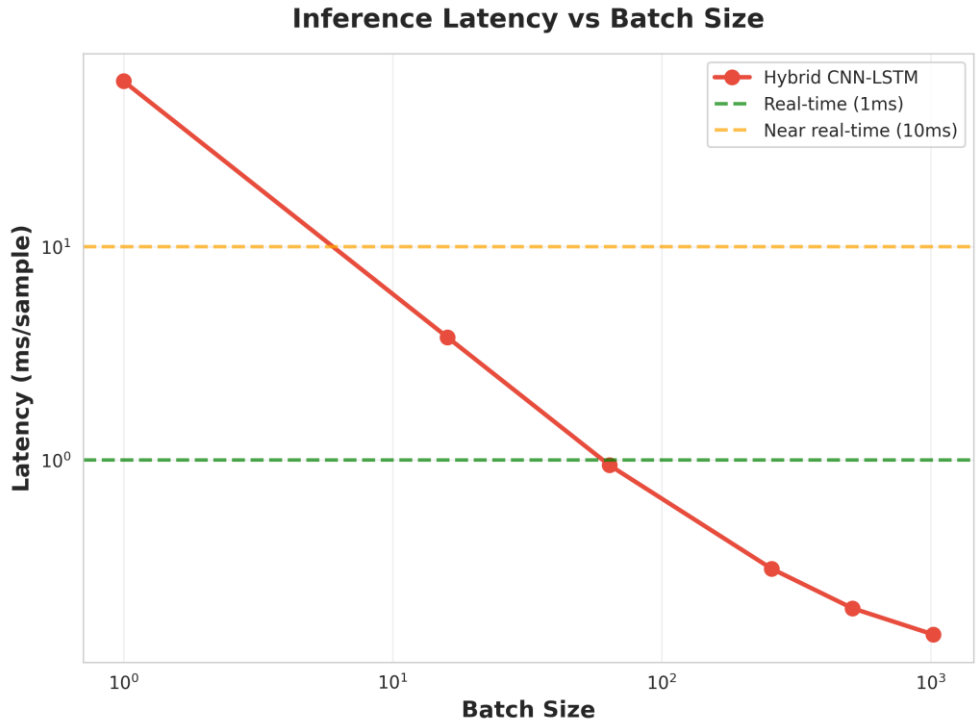
*Bảng 5 Bảng so sánh về Latency và Throughput*

| Batch Size | Latency (ms/sample) | Throughput (samples/sec) |
|---|---|---|
| 1 | 58.11 ± 1.01 | 17 |
| 16 | 3.70 ± 0.10 | 270 |
| 64 | 0.92 ± 0.02 | 1,081 |
| 256 | 0.30 ± 0.01 | 3,280 |

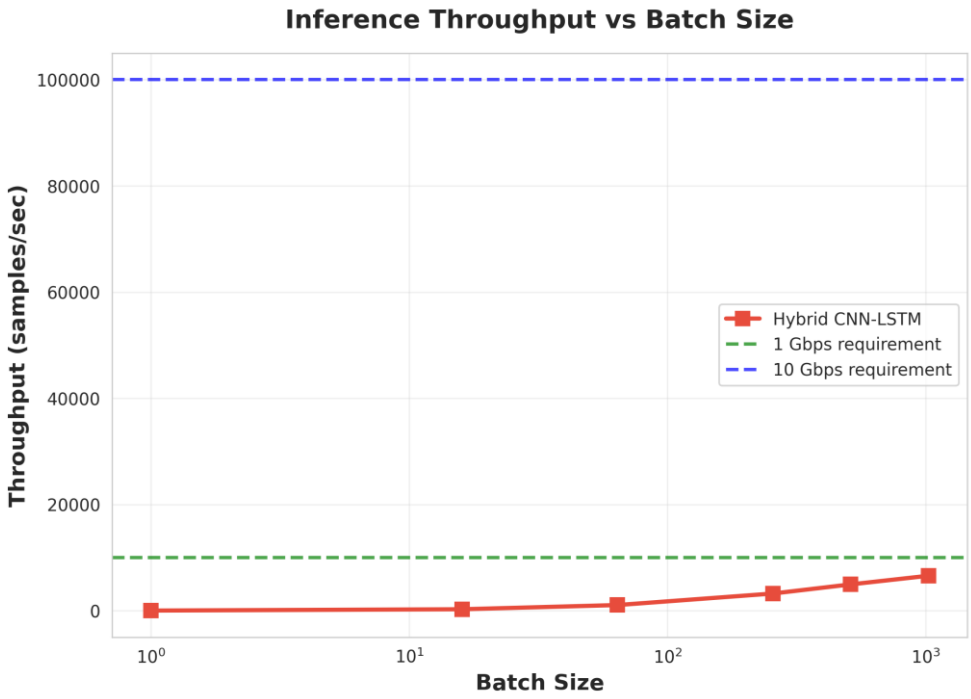
| Batch Size | Latency (ms/sample) | Throughput (samples/sec) |
|---|---|---|
| **512 (Optimal)** | **0.20 ± 0.003** | **5,027** |
| 1024 | 0.15 ± 0.002 | 6,793 |

Kết quả cho thấy mô hình đạt hiệu năng tối ưu tại batch size = 512 với độ trễ 0.20 ms/sample và thông lượng 5,027 samples/giây. Đây là điểm cân bằng tốt nhất giữa độ trễ thấp và thông lượng cao.

**Biểu đồ latency/throughput**




*Hình 8 Phân tích hiệu năng thời gian thực của mô hình CNN-LSTM: Độ trễ suy luận*




*Hình 9 Phân tích hiệu năng thời gian thực của mô hình CNN-LSTM: Thông lượng xử lý theo batch size*

**Từ biểu đồ có thể thấy:**

- **Về độ trễ**: Khi tăng kích thước batch từ 1 lên 512, độ trễ giảm mạnh từ 58.11 ms xuống 0.20 ms/sample. Tại batch size = 512, mô hình đạt ngưỡng real-time (< 1 ms), phù hợp cho các hệ thống giám sát thời gian thực.
- **Về thông lượng**: Thông lượng tăng tỷ lệ thuận với kích thước batch. Tại batch size = 512, mô hình đạt 5,027 samples/giây. Tuy nhiên, giá trị này vẫn thấp hơn yêu cầu của mạng 1 Gbps (khoảng 10,000 packets/giây).

**Production Readiness**

OPTIMAL CONFIGURATION: Batch Size 512

- Latency: 0.20 ms/sample

- Throughput: 5,027 samples/sec

- Memory: < 2 GB GPU VRAM

Network Capacity Support:

Sub-Gbps networks (100-900 Mbps)

Enterprise networks

IoT gateways

1+ Gbps needs optimization

Deployment Scenarios:

Real-time monitoring systems

Small-medium data centers

Security Operations Centers (SOC)

## 3.3 Kết quả thực nghiệm - GNN - ĐỐI TÁC VIẾT

## 3.4 Kết quả thực nghiệm khi so sánh - ĐỐI TÁC VIẾT

## 3.5 Phân tích và thảo luận

### 3.5.1 Thành công của CNN-LSTM Baseline

#### 3.5.1.1 Điểm mạnh:

1. **High Recall (89.97%)**:
    - Detect 12,286/13,656 attacks
    - Only miss 1,370 attacks (10.03%)
    - Critical cho IDS: Minimize false negatives
2. **Excellent AUC-ROC (93.06%)**:
    - Strong discrimination capability
    - Model separates Benign/Attack classes rất tốt
3. **Real-time Capability**:
    - Latency: 0.20 ms/sample
    - Throughput: 5,027 samples/sec
    - Production-ready cho sub-Gbps networks
4. **Balanced F1-Score (81.35%)**:
    - Good trade-off giữa Precision và Recall

#### 3.5.1.2 Hạn chế:

1. **Lower Precision (74.23%)**:

- 4,265 false positives (13.61% FPR)
- Security analysts phải investigate false alarms
- Có thể dẫn đến "alert fatigue"

2. **High Labeled Data Requirement**:
    - Cần 70% (2.1M samples) labeled data
    - Expensive và time-consuming để label
    - Không scalable cho new attack types
3. **Sequential Processing**:
    - Không exploit network graph structure
    - Miss relationships giữa IP connections
    - Limited by timestep window (10 steps)

### 3.5.2 Thảo luận GNN - ĐỐI TÁC VIẾT

## 3.6 So sánh GNN với CNN-LSTM Baseline

# CHƯƠNG 4: Kết luận và hướng phát triển

## 4.1 Tóm tắt kết quả và đóng góp chính

### 4.1.1 Tóm tắt CNN-LSTM Baseline

Nghiên cứu đã thành công xây dựng Hybrid CNN-LSTM model làm baseline cho so sánh với GNN:

**Đóng góp về mô hình CNN-LSTM:**

1. **Kiến trúc Hybrid 4-component**:
    - CNN: Spatial feature extraction
    - Bidirectional LSTM: Temporal modeling
    - Attention mechanism: Timestep weighting
    - Residual connection: Gradient flow improvement
2. **Performance đạt được**:
    - Accuracy: 87.48%
    - F1-Score: 81.35%
    - Recall: 89.97% (high detection rate)
    - AUC-ROC: 93.06%
3. **Real-time capability**:
    - Latency: 0.20 ms/sample
    - Throughput: 5,027 samples/sec
    - Production-ready cho enterprise deployment
4. **Benchmark cho GNN comparison**:
    - Cung cấp strong baseline (87.48% accuracy)
    - Cùng dataset, cùng metrics
    - Fair comparison framework

## 4.2 Tóm tắt GNN - ĐỐI TÁC VIẾT

## 4.3 So sánh tổng thể

## 4.4 Hạn chế và tồn tại

### 4.4.1 Hạn chế của CNN-LSTM

**Label Dependency:**

- Cần 70% labeled data (2.1M samples)
- Expensive labeling cost
- Not scalable cho new attack types

**Sequential Limitation:**

- Chỉ capture temporal patterns
- Bỏ qua graph structure của network
- Limited context window (10 timesteps)

**Precision Issue:**

- 74.23% precision → 13.61% false positive rate
- 4,265 false alarms trong 45K test samples
- Workload increase cho security analysts

**Network Speed Limitation:**

- Throughput 5,027 samples/sec
- Chỉ support sub-Gbps networks
- Cần optimization cho 1+ Gbps

### 4.4.2 Hạn chế của GNN - ĐỐI TÁC VIẾT

### 4.4.3 Hạn chế chung của nghiên cứu

## 4.5 Hướng phát triển

### 4.5.1 Cải tiến CNN-LSTM

#### 4.5.1.1 Cải tiến ngắn hạn

- **Multi-class Classification:**

- Mở rộng từ phân loại *Binary* sang *Multi-class*.
- Nhận diện các loại tấn công (*attack types*) cụ thể.
- Sử dụng *output layer Softmax* thay vì *sigmoid*.

- **Threshold Optimization:**
  - Tinh chỉnh (*tune*) *threshold* để giảm thiểu *false positives*.
  - Áp dụng *Cost-sensitive learning*.
  - Điều chỉnh *threshold* động (*Dynamic threshold adjustment*).
- **Model Compression:**
  - Thực hiện *Quantization* (INT8).
  - *Pruning* (cắt tỉa) các kết nối dư thừa.
  - Áp dụng *Knowledge distillation*.
  - Mục tiêu: Tăng tốc độ *inference* lên gấp 10 lần.

#### 4.5.1.2 Cải tiến dài hạn

- **Transfer Learning:**
  - *Pre-train* trên bộ dữ liệu CICIDS2018.
  - *Fine-tune* trên các *datasets* khác.
  - Sử dụng các kỹ thuật *Domain adaptation*.
- **Online Learning:**
  - Cập nhật *incremental* (tăng dần) với dữ liệu mới.
  - Thích ứng với các mẫu tấn công đang thay đổi (*evolving attack patterns*).
  - Xây dựng *pipeline Continuous learning*.

### 4.5.2 Cải tiến GNN - ĐỐI TÁC VIẾT

# Tài liệu tham khảo