Encrypted Traffic Classification

Deep Learning-Based VPN Detection System

Team Members:
Aravindh P - AM.SC.U4CYS23011
Jaifin B Aloor - AM.SC.U4CYS23022
Richu James - AM.SC.U4CYS23036

Research Question

How can deep learning models effectively classify encrypted mobile service traffic without payload inspection?

Can hybrid CNN-LSTM architectures outperform standalone models in capturing spatial and temporal traffic patterns?

How do packet sequences and flow statistics enable accurate VPN/Non-VPN service identification?

Problem Statement

Modern mobile apps use encryption (HTTPS, TLS, VPN)

Payload-based classification is impossible due to encryption

Network operators need real-time traffic identification for:

- Quality of Service (QoS) management
- Security monitoring and anomaly detection
- Network resource allocation

Challenge: Distinguishing services with similar encrypted traffic patterns

ISCX VPN-NonVPN Dataset 2016

Scenario A1: Pure VPN traffic (Skype, YouTube, Facebook over VPN)

Scenario A2: Mixed VPN and Non-VPN applications

Scenario B: Comprehensive real-world scenarios

Total Samples: 192,279 network flows

Features: Flow duration, packet counts, inter-arrival times, byte counts

Training: 153,823 flows | Testing: 38,456 flows

Deep Learning Models

CNN Architecture

Conv1d (32 filters)

Max Pooling

Conv1d (64 filters)

Max Pooling

Fully Connected

Learns spatial packet patterns

LSTM Architecture

2-layer LSTM

128 hidden units

Processes sequences

Captures temporal dynamics

Final hidden state

Understands flow behavior over time

Hybrid CNN-LSTM Model

Combines spatial and temporal feature extraction

CNN Stage: Extracts local packet patterns from flow sequences

LSTM Stage: Analyzes temporal evolution of traffic characteristics

Architecture: CNN → Permute → LSTM → Dense

Expected: Better accuracy than standalone models

Feature Engineering

Flow Duration: Total time span of network flow

Inter-Arrival Times: Time gaps between consecutive packets

Packet Counts: Forward and backward packet counts

Byte Lengths: Total bytes transmitted in each direction

Flow Statistics: Aggregated metrics per flow

All features extracted from encrypted headers (no payload needed)

Training Configuration

Device: Apple M4 Pro with MPS acceleration

Batch Size: 64 samples per iteration

Epochs: 20 training iterations

Learning Rate: 0.001 (Adam optimizer)

Loss Function: CrossEntropyLoss

Train-Test Split: 80/20 with stratification

Preprocessing: StandardScaler normalization

Tools & Technologies

ML/DL Frameworks

PyTorch (Deep Learning)

scikit-learn (Preprocessing)

SciPy (Data Loading)

Pandas & NumPy (Data Processing)

Matplotlib (Visualization)

Environment & Hardware

Python 3.14

macOS (M4 Pro)

GPU Acceleration (MPS)

Jupyter-compatible

Open-source libraries

Implementation Pipeline

1. Data Loading: Parse 20 ARFF files from 3 scenarios
2. Feature Extraction: 8 numeric flow-based features
3. Preprocessing: StandardScaler normalization
4. Reshape: Convert to (batch, 1, features) for Conv1d
5. Train: CNN, LSTM, Hybrid models sequentially
6. Evaluate: Accuracy, F1-Score, Loss curves
7. Compare: Identify best performing model

Model Performance Metrics

Accuracy: Percentage of correct predictions

Precision: True positives / (True positives + False positives)

Recall: True positives / (True positives + False negatives)

F1-Score: Harmonic mean of precision and recall

Training Loss: Decreasing curve indicates successful learning

Confusion Matrix: Class-wise performance breakdown

Results Summary

CNN Model: Captures spatial patterns in packet sequences

LSTM Model: Captures temporal dynamics of traffic flows

Hybrid Model: Superior performance combining both approaches

Comparative Analysis:

- Training loss convergence across epochs
- Test set accuracy and F1-scores
- Per-class performance metrics

Research Gaps Addressed

Gap 1: Limited reproducible implementations in open literature

→ Provided complete end-to-end pipeline code

Gap 2: Underutilization of hybrid CNN-LSTM architectures

→ Demonstrated effectiveness of combined approach

Gap 3: Lack of practical deployment examples

→ Created live traffic demo capability

Demo: Live Traffic Classification

Real-time packet capture using Scapy

Feature extraction from captured packets

Hybrid model inference on live flows

Visual alerts for encrypted (VPN) traffic detection

Displays: Flow count, confidence scores, classification labels

Synchronized with Wireshark packet monitoring

Shows practical applicability of trained model

Conclusion

Successfully implemented production-ready encrypted traffic classifier

Demonstrated feasibility of statistical feature-based classification

Hybrid CNN-LSTM combines complementary analysis approaches

Validated on large real-world ISCX dataset

Framework extensible for modern protocols and applications

Ready for deployment in network security systems

Future Work & Recommendations

Transfer Learning: Pre-trained models on diverse traffic datasets

Real-time Edge Deployment: Optimize for mobile/IoT devices

Modern Protocols: Support QUIC, HTTP/3, emerging protocols

Adversarial Robustness: Test against traffic obfuscation techniques

Federated Learning: Distributed training preserving privacy

Zero-shot Classification: Handle unseen applications with few examples

Key Takeaways

Statistical features enable effective encrypted traffic classification

Deep learning models require careful architecture selection

Hybrid approaches leverage strengths of multiple model types

Public datasets enable reproducible security research

Implementation complexity manageable with modern frameworks

Practical applications in network monitoring and security

Thank You!

Department of Cyber Security
Amrita School of Computing