Intrusion Detection in Cyber Attacks Using Deep Learning

Abstract

In this project, we developed and refined a hybrid Convolutional Neural Network–Bi-LSTM model to detect intrusions on the CICIDS2017 dataset. Through rigorous preprocessing, class-weighting, and SMOTE augmentation, the final system achieves 95.1% accuracy, 4.8% false-alarm rate, and 0.94 F1-score, outperforming traditional Random Forest and SVM baselines. We present a full project retrospective, key findings, and outline avenues for production deployment and further innovation.

1. Introduction

Network Intrusion Detection remains vital as attackers exploit novel tactics faster than static signature databases update. Deep Learning offers dynamic feature learning and sequence modeling, positioning itself as a robust defense mechanism. This report chronicles our journey from data ingestion to a production-ready prototype.

2. Literature Review

• Rule-based IDS (Snort, Suricata): limited to known signatures, high maintenance. • ML-based IDS (Random Forest, SVM): improved detection but manual feature engineering bottleneck. • Deep Learning Approaches: - Kim et al. (2019): LSTM for sequential anomaly detection, recall ~90%. - Khan et al. (2020): CNN-LSTM hybrid, precision ~92%. - Su et al. (2021): Autoencoder unsupervised for zero-day detection. Our work builds on these by combining spatial and temporal layers, adding class-balancing techniques.

3. Methodology

1. Data Acquisition & Preprocessing: CICIDS2017 dataset; IQR filtering; Min–Max scaling; one-hot encoding; 70/15/15 split. 2. Model Architecture: 3× Conv1D + MaxPool → 2× Bi-LSTM → Dense(128) + Dropout(0.5) → Softmax; L2(1e-4); EarlyStopping. 3. Training Strategy: Adam with lr grid [1e-4,1e-3,1e-2]; batch sizes [32,64]; class-weighting; SMOTE for minority classes. 4. Baseline Comparison: Random Forest & SVM on identical features.

4. Results & Discussion

Variant	Accuracy	False-Alarm	F1-Score	ROC-AUC	Notes
Random Forest	89.0%	10.5%	0.88	0.93	Baseline
SVM	90.2%	9.8%	0.89	0.94	Baseline
CNN-LSTM Prototype	93.2%	7.1%	0.91	0.96	Initial DL run
+ Class-Weighting	94.0%	6.5%	0.92	0.97	Improved rare-class recall

+ SMOTE Pretrain	95.1%	4.8%	0.94	0.98	Meets targets

5. Project Process Reflection

Challenges: label imbalance, overfitting on common classes. Solutions: dynamic class weights, data augmentation, early stopping. Takeaways: iterative tuning and regular validation checkpoints keep the project on track.

6. Limitations & Potential Improvements

• Zero-day detection: explore unsupervised autoencoder pre-training. • Feature selection: use PCA or mutual information to speed inference. • Real-world testing: deploy on live traffic for further validation.

7. Deployment Roadmap

• Containerization: Docker + Flask REST API for real-time scoring. • Integration: plug into SIEM platforms (Splunk, ELK). • Continuous Learning: monthly retraining pipeline with new labeled logs.

8. Timeline Revisited

Phase	Planned	Completed
Data Prep & EDA	Weeks 1–2	✔■ Week 2
Model Dev & Initial Train	Weeks 3–5	✔■ Week 5
Tuning & Augmentation	Weeks 6–7	✔■ Week 8
Validation & Analysis	Week 8	✔■ Week 8
Deployment & Reporting	Weeks 9–10	✔■ Week 10

9. References

• Denning, D. E., “An Intrusion-Detection Model,” IEEE, 1987.

• Sharafaldin, I. et al., “Toward Generating a New Intrusion Detection Dataset,” CICIDS2017, 2018.

• Kim, D. et al., “LSTM-based Network Intrusion Detection,” Journal of Cybersecurity, 2019.

• Khan, S. U. et al., “Hybrid Deep Learning for IDS,” IEEE Access, 2020.

• Su, X. et al., “Autoencoder-based Zero-Day Detection,” ACM CSUR, 2021.