Cyber Attack Detection Using Deep Neural Networks

Abstract

This project presents a deep neural network approach for detecting cyber attacks in network traffic. The dataset contains 15 different attack types with significant class imbalance, where rare attacks comprise less than 0.01% of samples. We implemented a multi-layer neural network with dropout and batch normalization, comparing different activation functions (ReLU vs ELU) and optimization strategies. By incorporating a learning rate scheduler, we achieved 91% overall accuracy with dramatically improved recall for minority classes (from 5% to 100% for certain attack types). This demonstrates the importance of adaptive learning strategies in handling imbalanced cybersecurity datasets.

1. Introduction

In today's digital age, cyber attacks pose a significant threat to organizations worldwide. Traditional security methods often fail to detect sophisticated attacks, especially rare ones. This project implements a deep learning approach to identify 15 different types of cyber attacks in network traffic.

I chose this project because it presents a unique and challenging problem that is not commonly addressed in typical deep learning courses. The main challenge lies in the severe class imbalance - while normal traffic comprises 85% of the data, some attack types have as few as 2-5 samples, making them extremely difficult to detect.

2. Dataset Description

The dataset contains network traffic data with 15 different classes:

- Class 0: Normal traffic (284,500 samples - 85%)
- Classes 1-14: Various cyber attacks (ranging from 2 to 25,926 samples)

The dataset contains 50 features extracted from network traffic. The significant class imbalance presents a major challenge - while some attack types have thousands of samples, others have as few as 2-5 samples. This required implementing class balancing techniques to ensure the model learns to detect all attack types equally, rather than focusing only on the majority classes.

3. Methodology

3.1 Model Architecture

The implemented neural network consists of:

- Input layer: 50 features
- Hidden layers: [256, 224, 192, 160, 128, 96, 64] neurons
- Output layer: 15 classes

- Activation: ReLU
- Regularization: Dropout (0.3) and Batch Normalization

This architecture was designed with additional hidden layers to enable better learning between layers. The gradual decrease in layer sizes (rather than sharp drops) helps the model learn more smoothly and prevents it from missing important patterns during the transformation from input to output.

3.2 Training Configuration

- Optimizer: Adam
- Initial Learning Rate: 0.001
- Batch Size: 512
- Epochs: 100
- Loss Function: Cross-Entropy Loss

4. Experiments

4.1 Baseline Model

Initial training with standard parameters achieved 98% accuracy but failed to detect rare attacks:

- Class 1: 45% recall
- Class 12: 5% recall
- Classes 8, 9: 0% recall

4.2 Optimization Attempts

Several approaches were tested:

1. **ELU Activation Function**: Slightly improved accuracy (87%) but reduced precision
2. **Learning Rate Scheduler**: Implemented ReduceLROnPlateau scheduler

Initially, a fixed learning rate of 0.001 was used. To improve the model, we implemented a Learning Rate Scheduler that adjusts the learning rate gradually. When the model's performance plateaus or degrades, the scheduler reduces the learning rate, allowing for finer adjustments. This adaptive approach enables more precise learning, particularly helping the model learn patterns of rare attack types.

5. Results

5.1 Final Model Performance

- Overall Accuracy: 91%
- Macro-average ROC-AUC: 0.9958
- Training completed in ~2 hours on GPU (T4)

5.2 Significant Improvements

- Class 1: Recall improved from 45% to 94%
- Class 12: Recall improved from 5% to 100%
- Class 8: Recall improved from 0% to 100%
- Class 9: Recall 80% (from 0%)

5.3 Comparison: Before and After Scheduler

Before implementing the Learning Rate Scheduler, the model achieved high accuracy (98%) but suffered from low recall in several classes, indicating the model failed to identify many actual attack cases. After implementing the scheduler, while the accuracy slightly decreased to 91%, the recall improved dramatically across all minority classes. This trade-off is highly favorable in cybersecurity contexts, where missing an attack (false negative) is far more dangerous than raising a false alarm (false positive).

6. Conclusions

This project demonstrates that high accuracy alone is not sufficient for cyber attack detection. The significant improvement in recall rates shows that adaptive learning strategies are crucial for handling imbalanced datasets. The model now successfully identifies even the rarest attacks, making it practical for real-world deployment.

6.1 Key Learnings

The most important lesson learned was the significant impact of Learning Rate Scheduler on model performance, particularly for imbalanced datasets. Witnessing the dramatic improvements in minority class detection demonstrated how crucial adaptive learning strategies are in deep learning.

6.2 Future Improvements

Given more computational resources, I would experiment with smaller batch sizes to investigate their effect on learning dynamics. Due to time and resource constraints, the project was completed with the current configuration, which nevertheless achieved satisfactory results for practical deployment.

7. Technical Implementation

- **Framework**: PyTorch
- **Development Environment**: Lightning AI (Cloud GPU)
- **Hardware**: NVIDIA T4 GPU
- **Training Time**: ~2 hours for 100 epochs

8. Acknowledgments

This project was completed as part of my deep learning studies, representing my first professional implementation of a neural network for cybersecurity applications.