Deep Learning for Network Intrusion Detection on NSL-KDD

Aykhan Mammadli, Electrical and Computer Engineering, New York University
am15226@nyu.edu

Abstract

Intrusion Detection Systems classify network connections as benign or malicious. We study binary intrusion detection on the NSL-KDD benchmark and compare a classical baseline like logistic regression with a multilayer perceptron as known as MLP. Because missed attacks are costly and IDS datasets are often class-imbalanced, we emphasize attack-class recall and F1-score and test imbalance-aware training (balanced class weights / weighted loss). On the NSL-KDD test set, the best model is an MLP with `pos_weight` achieving Recall=0.688 and F1=0.793, narrowly edging the unweighted MLP (Recall=0.680, F1=0.791) and outperforming logistic regression (Recall=0.591, F1=0.718). Source code: (`https://github.com/AykhanJ/DeepLearningProject/tree/main/IDS_Project`)

1. Introduction

Networked systems face attacks such as probing, denial-of-service, and credential abuse. An Intrusion Detection System aims to identify malicious activity from observed traffic while keeping false alarms manageable. The most costly errors are considered false negatives in many security settings. This motivates evaluating models with attack-class recall and F1-score rather than accuracy alone. Another practical challenge is data imbalance. Data imbalance means, benign traffic may dominate, and certain attack patterns can be under-represented. As a result, IDS evaluation typically emphasizes recall/F1 for the attack class and tests robustness techniques that mitigate imbalance.

Our project formulates IDS as a supervised binary classification task using the NSL-KDD dataset, a widely used benchmark derived from KDD Cup 99 [Tavallaee et al.(2009)Tavallaee, Bagheri, Lu, and Ghorbani].

Additionally, Our repository includes a README with step-by-step commands to reproduce preprocessing, training, and table/figure generation.

We compare a strong classical baseline such as logistic regression against a simple deep learning model, MLP. To strengthen the methodology, we add an imbalance-handling extension and empirically evaluate its effect on both model families. Our main finding is that the MLP improves test-set recall and F1 over logistic regression. But imbalance weighting provides only small gains for logistic regression and only marginal gains for the MLP in this setting.

2. Background and Related Work

2.1 Benchmark Datasets for IDS

KDD Cup 99 and its variant NSL-KDD have been widely used in IDS research. However, the original KDD Cup 99 dataset contains redundant records that can bias training and inflate evaluation scores. NSL-KDD was proposed to mitigate these issues by reducing redundancy and providing a more meaningful benchmark for comparing methods [Tavallaee et al.(2009)Tavallaee, Bagheri, Lu, and Ghorbani].

NSL-KDD still remains a common reference point for controlled evaluations, despite it is not a modern enterprise traffic dataset.

2.2 Classical Machine Learning Approaches

Traditional IDS approaches often rely on classical machine learning models trained on engineered connection features, including decision trees, support vector machines, and logistic regression. These models are attractive because they are relatively simple, fast to train, and can be easier to interpret than deep networks. Logistic regression in particular is a strong baseline for linearly separable problems and is commonly used to establish a minimum-performance reference.

2.3 Deep Learning for IDS

Deep learning approaches aim to capture non-linear feature interactions and reduce reliance on hand-crafted decision rules. Shone et al. report competitive results with deep-learning-based intrusion detection and highlight benefits of learned representations for detection [Shone et al.(2018)Shone, Ngoc, Phai, and Shi].

Other work studies deep and recurrent architectures for IDS, including RNN-based intrusion detection [Yin et al.(2017)Yin, Zhu, Fei, and He], and

deep models applied to network traffic features [Javaid et al.(2016)Javaid, Niyaz, Sun, and Alam].

2.4 Architectural Choice

We use an MLP because NSL-KDD provides fixed-length, connection-level tabular features after one-hot encoding and standardization. An MLP is a natural baseline for learning non-linear interactions among these heterogeneous attributes. In contrast, models designed for sequences are most appropriate when raw packet streams or time-ordered flow records are available, which this benchmark does not directly provide. A strong non-neural alternative for tabular IDS is tree-based ensembles (e.g., Random Forest or gradient boosting), which often perform competitively on structured features. We leave this comparison for future work.

Autoencoder ensembles have also been proposed for online anomaly detection. It emphasizes adaptive representations and streaming settings [Mirsky et al.(2018)Mirsky, Doitshman, Elovici, and Shabtai].

2.5 Imbalance and Evaluation Metrics

A persistent issue in IDS datasets is class imbalance. Imbalance-aware learning (class weighting, resampling, or loss reweighting) is often used to improve detection of rare malicious events. Because missed attacks can be costly, IDS papers frequently report recall and F1 for the attack class in addition to accuracy, and confusion matrices are important to understand false negatives and false positives.

3. Dataset and Preprocessing

We use NSL-KDD, where each instance corresponds to a network connection described by mixed feature types, including categorical attributes (for example, protocol type, service, flag) and numeric attributes (byte counts, error rates, traffic statistics and etc.). We convert original labels into a binary target: normal (0) versus attack (1). The official dataset provides separate train and test files; we further split the training file to create a validation set.

Preprocessing Pipeline

We apply the following preprocessing steps:

- **Categorical encoding:** one-hot encode categorical attributes. Unknown categories are ignored at inference.
- **Feature scaling:** standardize numeric features using training statistics only (mean 0, variance 1).
- **Data splits:** create a stratified validation split from the training file (15% validation). The official test file is used only for final evaluation to prevent tuning on test data.

After preprocessing, each example becomes a fixed-dimensional numeric vector used by both logistic regression and the MLP. We also save the fitted preprocessor to ensure reproducibility.

4. Methodology

Baseline: Logistic Regression

We train logistic regression on the processed feature vectors using the SAGA solver with a maximum of 500 iterations. We evaluate two variants: (a) unweighted training and (b) balanced class weights via `class_weight=balanced`, which reweights samples inversely proportional to class frequency.

4.1 Deep Model: Multilayer Perceptron

Our deep model is a feedforward MLP with hidden layer sizes [256, 128, 64], ReLU activations, and dropout 0.2. The network outputs a single logit and is trained with binary cross-entropy with logits. We use Adam with learning rate 10^{-3} and batch size 512. Early stopping is applied based on validation F1-score with patience 5; the best checkpoint by validation F1 is used for test evaluation.

4.2 Imbalance Handling Extension

To address class imbalance as a methodological extension, we evaluate (i) balanced class weights for logistic regression and (ii) a weighted loss for the MLP using a positive-class weight `pos_weight` computed from the training split as the ratio of negative to positive examples. This increases the penalty for misclassifying attacks.

4.3 Hyperparameter Selection and Training Details

Hyperparameters were selected using the validation set with attack-class F1 as the primary model-selection metric. For logistic regression we fixed common stable settings (SAGA solver, max_iter=500). For the MLP we chose a small architecture suitable for tabular data and tuned only lightweight parameters (dropout and early stopping). We use a fixed random seed for reproducibility. All models share the same preprocessing, splits, and evaluation procedure to ensure a fair comparison.

4.4 Evaluation Metrics

We report accuracy, precision, recall, and F1-score for the positive class (attack). Confusion matrices are also included to interpret types of errors. Predictions use a 0.5 threshold on sigmoid probability (equivalently, logit ≥ 0).

5. Experiments and Results

5.1 Main Results

Table 1 reports test-set performance. Logistic regression achieves F1=0.718 and Recall=0.591. Applying balanced class weights yields a modest improvement (F1=0.724, Recall=0.599). The unweighted MLP substantially outperforms both logistic regression variants (F1=0.791, Recall=0.680).

Adding `pos_weight` to the MLP yields a very small improvement in Recall and F1 (Recall=0.688, F1=0.793),

Model	Acc	Prec	Recall	F1
LogReg	0.736	0.915	0.591	0.718
LogReg + class_weight	0.740	0.915	0.599	0.724
MLP	0.795	0.944	0.680	0.791
MLP + pos_weight	0.795	0.936	0.688	0.793

Table 1: Performance on NSL-KDD (attack is positive class).

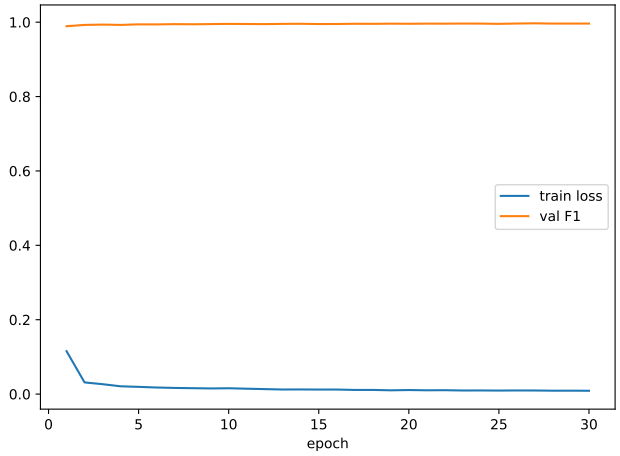

Figure 1: MLP training loss and validation F1 across epochs (early stopping selects the best validation F1).

while slightly reducing precision (0.936 vs. 0.944). This suggests the weighted loss mildly shifts predictions toward the positive class, improving missed-attack rate at the cost of a small increase in false positives.

5.2 Discussion: Validation vs. Test Gap

Validation performance is substantially higher than test performance. The behavior is consistent with known differences between NSL-KDD training and test distributions as the test set includes harder and less frequent patterns. Therefore, we emphasize the official test-set results for final comparison and keep model selection restricted to the validation split.

6. Conclusion

We originally intended to determine if a simple deep learning model could outperform a classical baseline on imbalanced network traffic data. We implemented an end-to-end IDS classification pipeline on NSL-KDD and compared logistic regression against a simple deep MLP. The strongest test performance was achieved by the MLP with `pos_weight` (Recall=0.688, F1=0.793), improving over the baseline (LogReg: Recall=0.591, F1=0.718). We also evaluated imbalance-aware training as a methodological extension: it provided small gains for logistic regression and only marginal gains for the MLP under the NSL-KDD test distribution. Future work could explore threshold tuning to trade off recall vs. false alarms, alternative imbalance methods, and cross-dataset evaluation to better estimate real-world generalization.

References

[Javaid et al.(2016)Javaid, Niyaz, Sun, and Alam] Javaid, A.; Niyaz, Q.; Sun, W.; and Alam, M. 2016. A Deep Learning Approach for Network Intrusion Detection System. *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS)*.

[Mirsky et al.(2018)Mirsky, Doitshman, Elovici, and Shabtai] Mirsky, Y.; Doitshman, D.; Elovici, Y.; and Shabtai, A. 2018. Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection. *Network and Distributed System Security Symposium (NDSS)*.

[Shone et al.(2018)Shone, Ngoc, Phai, and Shi] Shone, N.; Ngoc, T.; Phai, V. D.; and Shi, Q. 2018. A Deep Learning Approach to Network Intrusion Detection. *IEEE Transactions on Emerging Topics in Computational Intelligence*.

[Tavallaee et al.(2009)Tavallaee, Bagheri, Lu, and Ghorbani] Tavallaee, M.; Bagheri, E.; Lu, W.; and Ghorbani, A. A. 2009. A Detailed Analysis of the KDD CUP 99 Data Set. In *Proceedings of the 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA)*.

[Yin et al.(2017)Yin, Zhu, Fei, and He] Yin, C.; Zhu, Y.; Fei, J.; and He, X. 2017. A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks. *IEEE Access*.