Comprehensive Technical Report: Multi-Class DDoS Attack Detection

Contents

1 Introduction: The Challenge of DDoS Attack Classification

1.1 Problem Statement

This report details a comprehensive project undertaken to develop a high-accuracy, multi-class classification system capable of detecting and distinguishing between various types of Distributed Denial of Service (DDoS) attacks. The foundation of this work is the CICDDoS2019 benchmark dataset, a large-scale collection of network traffic data that realistically simulates modern DDoS threats. The project's core challenge was to navigate the significant complexities inherent in this dataset and engineer a robust detection pipeline.

The primary difficulties stemmed from several key characteristics of the data. The dataset is large, comprising 431,371 network flow samples, and possesses high dimensionality with an initial set of 88 features. This multi-class classification problem originally involved 13 distinct categories (benign traffic and 12 attack types), which was later refined to 12. Most critically, the dataset suffers from a severe class imbalance, with a documented ratio of 176.9-to-1 between the majority and minority classes. This imbalance poses a substantial risk of developing models that are highly accurate on dominant classes but fail to detect rare and potentially novel attack vectors.

To address these challenges, the project employed a systematic approach, beginning with meticulous data preparation and feature engineering.

1.2 Project Objectives and Scope

The project was guided by a set of clear objectives designed to address the problem statement systematically and deliver a validated solution.

Primary Objectives:

- To systematically preprocess and clean the raw CICDDoS2019 dataset to create a reliable foundation for modeling.
- To conduct rigorous feature analysis and selection to reduce dimensionality and improve model efficiency without sacrificing performance.
- To implement, train, and evaluate a diverse range of machine learning and deep learning models to identify the most effective architectures for this task.
- To explore advanced ensemble techniques to potentially improve upon the performance of individual models.
- To deliver a reproducible pipeline with saved models and preprocessed data for future collaboration.

This report outlines the complete project lifecycle, covering the foundational data preparation and feature engineering stages, the implementation details of all modeling methodologies, a comparative analysis of their performance, and a discussion of the validation techniques employed to ensure the robustness of the final results. This structured documentation provides a transparent overview of the methods and findings that underpin the project's conclusions.

2 Data Preparation and Feature Engineering

2.1 Dataset Overview and Preprocessing

A robust data preprocessing pipeline is critical for building accurate machine learning models, particularly when dealing with the inherent noise and complexity of network traffic data. The initial state of the CICDDoS2019 dataset required a multi-step cleaning process to prepare it for effective modeling.

The dataset's initial characteristics are summarized in Table 1.

Table 1: Initial Dataset Characteristics

Characteristic	**Value**
Total Samples	431,371
Initial Feature Count	88
Initial Class Count	13
Source	CICDDoS2019

A systematic data cleaning process was executed to address quality issues and refine the dataset for the classification task:

1. **Label Normalization:** Inconsistent class labels between training and testing files (e.g., "MSSQL" vs. "DrDoS_MSSQL") were standardized to ensure uniformity across the entire dataset.
2. **Handling of Data Quality Issues:** Infinity values, present in flow rate columns like "Flow Bytes/s", were replaced with NaN and subsequently imputed using the median value from the training data. Missing values were also imputed using the same strategy.
3. **Removal of Non-Predictive Columns:** Columns such as Flow ID, source/destination IP addresses, and timestamps were removed. These features are unique to specific sessions or hosts and would lead to overfitting rather than the learning of generalizable attack patterns.
4. **Handling of Severe Class Imbalance:** The 'WebDDoS' class, containing only 51 samples, was dropped entirely. Its extreme underrepresentation made it statistically insignificant and a source of potential model instability. This action reduced the classification task from 13 to 12 classes.

The reduction in total samples from 431,371 to 431,320 is a direct result of dropping the 'WebDDoS' class, which contained 51 samples. After these preprocessing steps, the cleaned dataset was split into training, validation, and test sets to facilitate model development and evaluation (Table 2).

Table 2: Dataset Split

Split	**Samples**
Training	276,044
Validation	69,012
Test	86,264
Total	431,320

With a clean and structured dataset, the focus shifted to reducing its high dimensionality through a scientific feature selection process.

2.2 Scientific Feature Selection

Following the initial data cleaning, a rigorous feature analysis was conducted to reduce model complexity, decrease training time, and enhance generalization by retaining only the most predictive features. This analysis employed four complementary methods: variance analysis, correlation analysis, Random Forest feature importance, and mutual information.

The key findings from this analysis were instrumental in refining the feature set:

- **Identification of Non-Discriminative Features:** A total of 17 features were identified as having no discriminative value. This group included zero-variance features (e.g., Fwd/Bwd URG Flags which were always zero), bulk transfer features irrelevant to DDoS traffic, and rarely used TCP flags.
- **Discovery of Redundant Features:** The analysis uncovered 10 pairs of perfectly correlated features ($r > 0.95$), such as "Total Fwd Packets" and "Subflow Fwd Packets," which were mathematically identical and thus redundant.
- **Identification of High-Value Feature Categories:** The most powerful predictors were found to be related to packet length (e.g., Fwd Packet Length Min), flow rate (e.g., Flow Bytes/s), and Inter-Arrival Time, which effectively capture the volumetric and timing characteristics of automated attack traffic.

The outcome of this scientific process was a 32% dimensionality reduction, trimming the feature set from 77 to 52 without compromising predictive power. This optimized feature set formed the basis for all subsequent modeling efforts.

3 Modeling Methodologies

To comprehensively explore the problem space, the project employed a diverse portfolio of modeling techniques. This ranged from traditional machine learning algorithms and powerful tree-based ensembles, which serve as strong baselines on tabular data, to complex deep learning architectures designed to capture intricate non-linear relationships. Finally, advanced ensemble methods were implemented to aggregate the strengths of the best-performing models.

3.1 Traditional and Tree-Based Models

Traditional and tree-based models were included due to their proven effectiveness and interpretability on structured, tabular datasets like the one used in this project. They provide robust baseline performance against which more complex architectures can be compared.

3.1.1 K-Nearest Neighbors (KNN)

KNN is a non-parametric algorithm that classifies new data points based on the majority class of their 'k' nearest neighbors in the feature space. Its strength lies in its ability to

capture complex decision boundaries without making assumptions about the underlying data distribution.

Hyperparameter tuning was performed using GridSearchCV with 5-fold stratified cross-validation. The search space included k values (3, 5, 7, 9, 11), distance metrics (Euclidean, Manhattan), and weighting schemes. The optimal configuration was found to be $k = 7$ with Manhattan distance and distance-weighted voting.

3.1.2 Logistic Regression

A linear classifier was implemented for multi-class classification using a softmax function at the output layer. The model learns a weight and bias for each class and was trained using the L-BFGS optimizer. To counteract class imbalance, the `class_weight='balanced'` parameter was utilized, which automatically adjusts the loss function to give more importance to samples from minority classes.

3.1.3 AdaBoost

AdaBoost is an iterative ensemble method that combines multiple weak learners (in this case, 100 shallow decision trees) into a single strong classifier. Each subsequent tree is trained to focus more on the samples misclassified by the previous ones. The final prediction is made via a weighted majority vote using the SAMME algorithm. The `class_weight='balanced'` parameter was also applied to the base decision tree estimators to address imbalance at the split level.

3.1.4 LightGBM

LightGBM is a high-performance gradient boosting framework optimized for large tabular datasets. It employs histogram-based splitting algorithms and a leaf-wise tree growth strategy to achieve significant gains in training speed and efficiency, making it a powerful baseline model.

3.1.5 XGBoost

XGBoost (Extreme Gradient Boosting) is another state-of-the-art gradient boosting model that builds decision trees sequentially, with each new tree correcting the errors of the previous ensemble. A grid search with 5-fold cross-validation was conducted to find the optimal hyperparameters, exploring `n_estimators`, `max_depth`, `learning_rate`, and `min_child_weight`. The final model used 100 estimators, a `max_depth` of 6, a `learning_rate` of 0.1, and `min_child_weight` of 3.

3.1.6 Random Forest

In one experimental workflow, a Random Forest model was trained on data preprocessed with Principal Component Analysis (PCA). After reducing the numeric features to 15 principal components that retained 96% of the information, various tree depths were tested. A final depth of 15 was selected to achieve a balance between model complexity and performance, avoiding the overfitting observed with deeper trees.

3.1.7 CatBoost

CatBoost, a third gradient boosting model known for its handling of categorical features and robustness, was also included in the suite of models for ensemble experiments.

3.2 Deep Learning Architectures

Deep learning models were explored for their ability to automatically learn complex, non-linear feature interactions that might be missed by traditional methods. These architectures are designed to capture hierarchical patterns within the 52-dimensional feature space.

3.2.1 Multi-Layer Perceptron (MLP)

An MLP was implemented to leverage deep learning's capabilities on tabular data. The architecture consisted of an input layer (52 features), three hidden layers with 256, 128, and 64 neurons respectively, and a final output layer with 12 neurons and a softmax activation. Each hidden layer incorporated Batch Normalization for training stability, ReLU activation for non-linearity, and a 30% Dropout rate for regularization. The model was trained using the Adam optimizer and accelerated with Apple's Metal Performance Shaders (MPS) backend.

3.2.2 Deep & Cross Network (DCN)

The DCN architecture features two parallel components whose outputs are concatenated before the final classification layer:

- **Cross Network:** This component is designed to learn explicit and bounded-degree feature interactions efficiently. It applies a unique cross-layer at each step to model feature crosses without manual engineering.
- **Deep Network:** This is a standard MLP that learns implicit, highly non-linear patterns through a series of dense layers: Input(52) $\rightarrow$ Dense(128) $\rightarrow$ ReLU $\rightarrow$ Dropout(0.1) $\rightarrow$ Dense(64) $\rightarrow$ ReLU $\rightarrow$ Dropout(0.1) $\rightarrow$ Dense(32) $\rightarrow$ ReLU $\rightarrow$ Dropout(0.1).

This dual structure allows the model to benefit from both explicit feature engineering and deep representation learning. It was trained using a weighted CrossEntropyLoss function to handle class imbalance.

3.2.3 Tabular Transformer

This model adapts the highly successful Transformer architecture from natural language processing to tabular data:

- **Feature Embedding:** Each of the 52 features is treated like a "token" and embedded into a higher-dimensional vector space (64 dimensions).
- **Positional Encoding:** Learnable embeddings are added to the feature representations to preserve their original order, as the self-attention mechanism is otherwise permutation-invariant.

- **Transformer Encoder:** The core of the model consists of multiple layers of Multi-Head Self-Attention, allowing each feature to weigh its interaction with every other feature to learn global dependencies.
- **Classifier Head:** The output from the encoder is flattened and passed through a final MLP for classification.

3.2.4 Convolutional Variational Autoencoder (CVAE)

The CVAE uses convolutional layers to learn a compact latent representation of the input data. By employing variational inference, it models the underlying data distribution, which can enhance robustness, especially for minority classes. The model has a dual function, with a decoder for reconstructing the original input and an integrated classification head for supervised attack detection.

3.2.5 CNN-LSTM

This hybrid model combines 1D convolutional layers with a Long Short-Term Memory (LSTM) layer. The convolutional layers act as feature extractors for spatial patterns among the 52 features, while the subsequent LSTM layer models temporal or sequential dependencies in the learned representations. For this, each input sample was reshaped to a (1, 52) sequence format.

3.3 Advanced Ensemble Methods

To improve upon the performance and robustness of individual models, ensemble learning techniques were implemented. These methods aggregate the predictions from multiple base classifiers, leveraging model diversity to achieve a more accurate and stable final prediction.

The ensemble experiments were conducted using a diverse set of 10 base models (Table 3).

Table 3: Base Models for Ensemble

Model Category	Models
Gradient Boosting	LightGBM, XGBoost, CatBoost
Tree-Based	Random Forest
Neural Networks	MLP, DCN, Transformer
Traditional ML	KNN, Logistic Regression, AdaBoost

Two primary categories of ensemble techniques were implemented:

3.3.1 Voting-Based Methods

- **Hard Voting:** The final prediction is the class that receives the most individual model votes.
- **Weighted Voting:** Each model's vote is weighted by its performance (either Accuracy or F1-score), giving more influence to stronger models.
- **Top-K Voting:** Only the predictions from the top 5 best-performing models are considered, reducing noise from weaker classifiers.

3.3.2 Stacking (Meta-Learning)

Stacking involves a two-level learning process. The base models first make predictions on the data. Then, a second-level model, or "meta-learner," is trained on these predictions to learn the optimal way to combine them. Four different meta-learners were tested: Logistic Regression, Random Forest, Gradient Boosting, and an MLP. To prevent data leakage, a 5-fold stratified cross-validation strategy was used for training the meta-learners.

The implementation of these varied modeling approaches provided a comprehensive basis for comparing their effectiveness on the DDoS classification task.

4 Experimental Results and Performance Analysis

This section presents a comprehensive analysis of the performance of all implemented models. It begins with a high-level comparison of overall effectiveness, delves into per-class performance to understand the impact of class imbalance, and concludes with an evaluation of the advanced ensemble techniques.

4.1 Overall Model Performance Comparison

The performance of all individual models was evaluated using standard classification metrics. Table 4 synthesizes these results from across all experiments, ordered from highest to lowest accuracy.

Table 4: Overall Model Performance

Model	**Accuracy**	**Precision (Macro)**	**Recall (Macro)**	**F1-Score (Macro)**
Random Forest (with PCA)	97.95%	0.8173	0.8486	0.8268
XGBoost	97.89%	—	—	—
MLP	97.27%	—	—	—
KNN	97.15%	—	—	—
CVAE	96.84%	0.7140	0.7403	0.7103
CatBoost	96.71%	—	—	0.7472
Tabular Transformer	96.64%	0.7266	0.7709	0.7376
DCN	96.28%	0.7095	0.7539	0.7120
Logistic Regression	94.87%	0.6697	0.7327	0.6834
AdaBoost	94.73%	0.6824	0.7360	0.6810
LightGBM	93.08%	0.5920	0.6088	0.5878
CNN-LSTM	93.08%	0.5920	0.6088	0.5878
Random Forest (for Ensemble)	20.41%	—	—	—

Note: Macro-averaged metrics for some of the top-performing models (XGBoost, MLP, KNN) are not included, as these specific metrics were not available in the source reports for those models.

It is critical to address the significant discrepancy in the reported performance of the Random Forest model. The experiment documented in eda.txt, which utilized PCA for feature reduction, achieved a very high accuracy of 97.95%. In contrast, the Random Forest instance used as a base model in the ensemble experiments showed an anomalously

low accuracy of 20.41%. This suggests a substantial difference in model configuration, preprocessing, or a potential issue during that specific experimental run.

Based on the validated results, the top-performing individual models are the XGBoost model with 97.89% accuracy and the Random Forest model from the PCA-based experiment with 97.95% accuracy. These tree-based ensembles establish a powerful performance benchmark for this classification task.

4.2 Per-Class Performance and Class Imbalance Impact

While overall accuracy scores are high, the macro-averaged metrics and per-class F1-scores reveal that detecting under-represented attack types remains a significant challenge. The performance varies dramatically depending on the number of available samples for each class (Table 5).

Table 5: Per-Class Performance Summary

Class Category	**Performance Summary**	**Attack Types**
Majority Classes (>10,000 samples)	All models performed excellently, with F1-scores consistently above 0.96. Deep learning models (DCN, Transformer) achieved near-perfect scores (>0.99).	Benign, NTP, Syn, TFTP
Moderate Classes (1,000–10,000 samples)	Performance was good, with most models achieving F1-scores between 0.72 and 0.94. DCN and Transformer were particularly effective.	MSSQL, UDP, UDPLag
Minority Classes (<1,000 samples)	These classes proved challenging, with F1-scores ranging widely. Performance was significantly lower despite weighted loss functions.	DNS, LDAP, NetBIOS, Portmap, SNMP

Different model architectures demonstrated unique strengths on the most difficult minority classes. For instance, the Tabular Transformer showed a significant performance improvement in detecting DNS (F1-score of 0.36) and NetBIOS (F1-score of 0.62) attacks compared to other models. The Deep & Cross Network (DCN) proved most effective at identifying Portmap attacks (F1-score of 0.35), which was the most challenging class for nearly all other models. This indicates that certain architectures are better suited to learning the subtle patterns of specific low-frequency attacks.

4.3 Evaluation of Ensemble Techniques

Ensemble methods were implemented to aggregate predictions from 10 base models with the goal of improving overall robustness and accuracy. The performance of these techniques is summarized in Table 6.

Table 6: Ensemble Method Performance

Ensemble Method	**Accuracy**	**F1-Score (Weighted)**
Stacking (Gradient Boosting)	97.81%	0.9773
Stacking (Random Forest)	97.72%	0.9766
Stacking (MLP)	97.54%	0.9746
Top-5 Voting	97.53%	0.9745
Weighted Voting (Accuracy)	97.38%	0.9730
Weighted Voting (F1)	97.38%	0.9730
Hard Voting	97.33%	0.9726
Stacking (Logistic Regression)	94.93%	0.9392

The analysis of these ensemble experiments yielded several key findings:

- The Stacking (Gradient Boosting) method was the best-performing ensemble, achieving 97.81% accuracy by effectively learning the optimal combinations of base model predictions.
- Critically, the best individual model, XGBoost (97.89%), slightly outperformed the best ensemble. This suggests that for this dataset, a single, exceptionally well-tuned model can be as effective as a more complex ensemble.
- The Top-5 Voting method outperformed other voting techniques. Its success highlights the importance of base model quality, as it excluded the noisy predictions from the underperforming Random Forest model used in that specific run.
- An Oracle Upper Bound of 98.92% was calculated, representing the theoretical maximum accuracy if the ensemble could perfectly choose the correct prediction whenever at least one base model was right. This demonstrates the potential for ensembling, even if the practical gain over the best single model was marginal in this case.

In summary, while the stacking ensemble provided highly robust and competitive results, a well-tuned XGBoost model remained the most performant and simpler solution for this specific problem.

5 Validation and Model Robustness

To ensure the reliability of the reported results and the generalizability of the trained models, the project incorporated several validation and overfitting mitigation techniques throughout the development lifecycle.

5.1 Stratified Cross-Validation

The primary validation strategy was 5-fold stratified cross-validation, which was used consistently during hyperparameter tuning (for KNN, XGBoost) and model evaluation (for the MLP). By stratifying the folds, this method ensures that the class distribution of the original dataset is preserved in each training and validation split. This is crucial for imbalanced datasets, as it prevents evaluation bias and provides a more accurate estimate of the model's performance on unseen data.

The highly consistent cross-validation results for the MLP (average accuracy of 97.12%) and XGBoost models serve as strong evidence of their stability and low variance.

5.2 Overfitting Mitigation Techniques

Several techniques were actively employed to prevent models from simply memorizing the training data, thereby promoting better generalization to new, unseen network traffic:

- **Regularization:** Both the MLP and DCN architectures included Dropout layers (with a rate of 30% in the MLP and 0.1 in the DCN) to randomly deactivate neurons during training, preventing complex co-adaptations. For XGBoost, the `min_child_weight` hyperparameter acted as a form of regularization by controlling the complexity of the decision trees.
- **Early Stopping:** The MLP model was trained with an early stopping mechanism that monitored validation loss. Training was halted if the loss did not improve for 5 consecutive epochs, preventing the model from continuing to train after performance had plateaued and overfitting was likely to begin.
- **Learning Rate Scheduling:** A ReduceLROnPlateau scheduler was used for the MLP. This technique automatically reduced the learning rate when validation loss stopped improving, allowing for finer-grained adjustments as the model approached an optimal solution.
- **Learning Curve Analysis:** For the DCN and Tabular Transformer models, training and validation loss curves were visualized (training_curves.png). This practice allows for direct visual inspection of the performance gap between training and validation sets, serving as a clear diagnostic tool for identifying overfitting.

Together, these validation and regularization methods provide strong confidence in the reported performance metrics and confirm the overall stability and robustness of the top-performing models developed in this project.

6 Conclusion and Future Work

6.1 Summary of Key Findings

This project successfully developed and validated multiple high-performance models for the multi-class detection of DDoS attacks on the complex CICDDoS2019 dataset. Through a systematic process of data cleaning, feature engineering, and comparative modeling, several critical insights were gained that define the project's outcome:

1. **Top Model Performance:** A well-tuned XGBoost model emerged as the single best-performing classifier, achieving a test accuracy of 97.89%. This result establishes a powerful benchmark for this tabular dataset, demonstrating the continued dominance of gradient boosting machines in this domain.
2. **The Impact of Class Imbalance:** Despite achieving high overall accuracy, performance on minority attack classes (e.g., Portmap, DNS) remains the primary challenge. This conclusion is strongly supported by the discrepancy between high overall accuracy and lower macro-averaged F1-scores across all models.

3. **Efficacy of Feature Engineering:** The scientific feature selection process proved highly effective. It successfully reduced the dataset's dimensionality by 32% (from 77 to 52 features), which streamlined model training and reduced complexity while maintaining top-tier performance.
4. **Ensemble Methods as a Robust Alternative:** While ensemble methods did not outperform the single best XGBoost model, the Stacking with Gradient Boosting technique delivered highly competitive and robust results. This demonstrates the value of ensembling as a production-worthy alternative that can increase stability and reduce variance.

6.2 Recommendations and Future Work

Based on the comprehensive results of this project, the following recommendations are proposed for potential deployment scenarios, along with clear directions for future research.

6.2.1 Recommendations for Deployment

- **For Simplicity and Performance:** The XGBoost model is the recommended choice. It provides the highest accuracy, is computationally efficient, and is simpler to deploy and maintain than a complex ensemble.
- **For Maximum Robustness:** The Stacking with Gradient Boosting ensemble is recommended. Although it offers only a marginal performance difference, its aggregated nature can provide greater stability and better generalization on unseen data distributions.

6.2.2 Directions for Future Work

- **Advanced Imbalance Handling:** To address the persistent challenge of minority class detection, future work should explore techniques beyond weighted loss functions. Methods such as data oversampling (e.g., SMOTE) or the implementation of dedicated cost-sensitive learning algorithms could significantly improve performance on critical classes like Portmap and DNS.
- **Alternative Feature Representations:** The CVAE model demonstrated an ability to learn a compact latent representation of the data. This latent space could be investigated as a new source of input features for other classifiers (like XGBoost or MLP) to determine if this abstracted representation improves class separability.
- **Production Pipeline Development:** The next logical step is to move from experimental models to a production-ready system. This would involve work on model optimization for low-latency inference, containerization (e.g., using Docker), and deployment within a real-time stream processing framework.