Deep Feature Learning Enhanced Ensemble Models for Multi-Class Intrusion Detection in Imbalanced IoT Networks

Abstract

The proliferation of Internet of Things (IoT) devices has created unprecedented security challenges, with intrusion detection systems (IDS) facing significant difficulties in accurately identifying diverse attack patterns within highly imbalanced network traffic datasets. Traditional machine learning approaches, while effective for majority classes, often fail to detect minority attack types that pose critical security threats. This study investigates the integration of deep learning feature extraction techniques with ensemble models to enhance multi-class intrusion detection performance in imbalanced IoT networks. Using the IoTID20 dataset containing 261,419 samples across 9 attack categories, we systematically evaluate autoencoder-based and convolutional neural network (CNN)-based feature extraction combined with XGBoost ensemble classifier, comparing these hybrid approaches against traditional sampling methods. Our comprehensive experimental analysis reveals that while baseline XGBoost achieved 91.46% accuracy on the preprocessed dataset, the integration of SMOTE with traditional features improved minority class detection by up to 65% for the most challenging attack type (Mirai-Ackflooding), increasing F1-score from 0.264 to 0.432. However, deep feature learning approaches demonstrated trade-offs: although dimensionality was successfully reduced from 79 to 32 features (59.5% compression), overall accuracy decreased to 84.93% for hybrid models. Our findings indicate that data quality preprocessing and appropriate sampling techniques should be prioritized before implementing computationally intensive feature learning, particularly for moderate-sized datasets. This research contributes practical insights into when deep learning feature extraction provides advantages in IoT intrusion detection and establishes that context-dependent approaches outperform universal architectural solutions.

Keywords: Internet of Things; intrusion detection; imbalanced learning; deep learning; ensemble methods; feature extraction; XGBoost; autoencoder; convolutional neural networks; SMOTE; cybersecurity; network security

1. Introduction

The rapid proliferation of Internet of Things (IoT) devices has fundamentally transformed modern infrastructure, enabling unprecedented connectivity across smart homes, industrial systems, healthcare facilities, and critical infrastructure [1]. However, this exponential growth has simultaneously created an expanded attack surface, with IoT networks becoming increasingly vulnerable to sophisticated cyber threats [3,4]. The unique characteristics of IoT environments—including resource constraints, heterogeneous device types, massive data volumes, and diverse communication protocols—pose significant challenges for traditional intrusion detection systems (IDS) [2].

Contemporary IoT networks generate enormous volumes of network traffic data, with the overwhelming majority representing normal behavior patterns. This creates a fundamental challenge: severe class imbalance in intrusion detection datasets [1]. While normal traffic and common attack types may constitute 90% or more of network data, rare but critical attack vectors—such as Man-in-the-Middle (MITM) attacks, specific Mirai botnet variants, and advanced reconnaissance scans—often represent less than 5% of total traffic [4]. This imbalance directly impacts the performance of machine learning-based IDS, which typically optimize for overall accuracy and consequently exhibit poor detection rates for minority attack classes [6].

Traditional machine learning approaches to intrusion detection, including decision trees, random forests, and support vector machines, have demonstrated strong performance on balanced datasets [7]. However, when confronted with the severe class imbalance inherent in real-world IoT traffic, these methods often fail to detect rare but dangerous attack types [1]. Ensemble methods, particularly gradient boosting algorithms such as XGBoost and CatBoost, have emerged as promising solutions due to their ability to handle complex feature interactions and provide robust predictions [8]. Nevertheless, even these advanced techniques struggle with extreme imbalance ratios exceeding 50:1, as observed in operational IoT networks [3].

To address the class imbalance problem, researchers have extensively investigated sampling-based approaches, including Synthetic Minority Over-sampling Technique (SMOTE), Adaptive Synthetic Sampling (ADASYN), and various undersampling methods [1,5]. While these techniques have shown promise in improving minority class detection, they introduce computational overhead and may generate synthetic samples that do not accurately represent the true data distribution [9]. Furthermore, sampling methods do not address the fundamental challenge of extracting discriminative features from high-dimensional, noisy IoT network traffic data [6].

Recent advances in deep learning have opened new possibilities for automated feature extraction in intrusion detection. Deep neural networks, including autoencoders and convolutional neural networks (CNNs), have demonstrated remarkable capability in learning hierarchical representations from raw data without manual feature engineering [4,7]. Autoencoders, through their encoder-decoder architecture, can learn compressed representations that capture the most salient characteristics of network traffic [10]. Similarly, CNNs have proven effective in extracting spatial and temporal patterns from sequential network data [11]. These deep learning approaches offer the potential to discover complex feature interactions that traditional methods might miss.

Despite these advances, several critical research gaps remain. First, while deep learning and ensemble methods have been studied independently for intrusion detection, their systematic integration remains underexplored, particularly in the context of severely imbalanced IoT datasets [1]. Second, the comparative effectiveness of different deep learning architectures (autoencoders vs. CNNs) for feature generation in ensemble-based IDS has not been thoroughly investigated [4]. Third, the trade-offs between dimensionality reduction through deep feature learning and detection performance across different attack categories require empirical examination [3]. Finally, practical guidance is lacking regarding when deep feature learning provides advantages over traditional sampling methods in resource-constrained IoT environments [6].

Fan et al. [1] conducted a comprehensive study examining sampling-based machine learning models for intrusion detection in imbalanced datasets, identifying significant limitations in minority class detection using traditional approaches. Their work on the IoTID20 dataset revealed that without appropriate sampling strategies, several attack types achieved zero detection rates. Building upon this foundation, they explicitly identified

the integration of deep learning feature generation with ensemble models as a promising direction for future research, suggesting that automated feature learning could address the fundamental limitations of sampling-only approaches [1].

This research directly addresses this identified gap by systematically investigating how deep learning-based feature extraction can be integrated with ensemble models to enhance multi-class intrusion detection performance in imbalanced IoT networks. Specifically, we examine whether autoencoder-based and CNN-based feature learning can improve upon traditional sampling methods when combined with gradient boosting classifiers. Our investigation encompasses not only performance improvements but also the practical trade-offs between model complexity, computational efficiency, and detection accuracy across different attack categories.

The primary contributions of this research are as follows:

1. **Systematic Integration Framework:** We present a comprehensive framework for integrating deep learning feature extractors (autoencoders and CNNs) with ensemble classifiers (XGBoost), providing detailed architectural designs and implementation guidelines suitable for IoT environments.
2. **Comparative Performance Analysis:** We conduct an extensive empirical evaluation comparing six distinct approaches: baseline ensemble methods, sampling-based methods (SMOTE), deep feature learning without sampling, and hybrid approaches combining deep features with SMOTE, using multiple evaluation metrics including class-specific F1-scores, macro/weighted averages, and Matthews Correlation Coefficient (MCC).
3. **Minority Class Detection Insights:** We provide detailed analysis of detection performance for minority attack classes, quantifying improvements and limitations across different methodological approaches and identifying which attack types benefit most from deep feature learning.
4. **Data Quality Impact Assessment:** We demonstrate the critical importance of data preprocessing and duplicate removal, showing how data quality can have comparable or greater impact than algorithmic complexity on detection performance.
5. **Practical Implementation Guidance:** We offer evidence-based recommendations regarding when to employ deep feature learning versus traditional sampling methods, considering factors such as dataset size, computational resources, and specific attack detection priorities.
6. **Trade-off Quantification:** We explicitly quantify the trade-offs between dimensionality reduction (79 to 32 features, 59.5% compression), computational efficiency, and detection accuracy across majority and minority classes.

This paper is organized as follows: Section 2 reviews related work on IoT intrusion detection, imbalanced learning, and deep learning approaches. Section 3 describes the methodology, including dataset characteristics, preprocessing procedures, deep learning architectures, and evaluation metrics. Section 4 presents comprehensive experimental results, including baseline performance, deep feature learning outcomes, and minority class detection analysis. Section 5 discusses the implications of our findings and provides practical recommendations. Finally, Section 6 concludes the paper and outlines directions for future research.

2. Related Works

This section reviews the relevant literature across five key areas: IoT security and intrusion detection systems, imbalanced learning techniques, deep learning approaches for network security, ensemble methods for classification, and hybrid architectures combining

multiple methodologies. This comprehensive review contextualizes our research within the broader landscape of cybersecurity and machine learning.

2.1. IoT Security and Intrusion Detection Systems

The exponential growth of IoT devices has fundamentally transformed network security paradigms, creating unique challenges that distinguish IoT environments from traditional network architectures [12]. IoT networks are characterized by heterogeneous devices with varying computational capabilities, diverse communication protocols, limited energy resources, and massive data generation rates [3]. These characteristics necessitate specialized intrusion detection approaches that can operate efficiently under resource constraints while maintaining high detection accuracy.

Traditional signature-based intrusion detection systems, while effective against known attacks, struggle to identify novel or evolving threats in IoT environments [13]. Consequently, researchers have increasingly focused on anomaly-based detection systems leveraging machine learning techniques [12]. However, the deployment of machine learning-based IDS in IoT contexts introduces its own set of challenges, including the need for labeled training data, computational overhead, and the ability to adapt to evolving attack patterns [4].

The IoTID20 dataset, introduced by Ullah and Mahmoud [2], represents a significant advancement in IoT security research by providing a realistic and diverse collection of network traffic data encompassing both benign activities and various attack types. This dataset addresses several limitations of earlier IoT security datasets by including contemporary attack vectors such as Mirai botnet variants, which have been responsible for numerous large-scale IoT compromises [2]. The dataset's comprehensive nature, including both network flow features and packet-level characteristics, has made it a valuable benchmark for evaluating intrusion detection approaches.

Recent work by Alkadi et al. [3] demonstrated the importance of feature selection in IoT intrusion detection, showing that careful selection of relevant network features can significantly improve detection performance while reducing computational complexity. Their study highlighted that not all network features contribute equally to attack detection, and that dimensionality reduction through intelligent feature selection can enhance both efficiency and accuracy. This finding motivates our investigation into deep learning-based feature extraction as an alternative approach to manual feature engineering.

Toldinas et al. [4] conducted a comprehensive evaluation of recurrent neural networks (RNNs) for network-based intrusion detection in IoT environments, revealing both the potential and limitations of sequential modeling approaches. Their work demonstrated that while RNNs can capture temporal dependencies in network traffic, they require substantial computational resources and careful hyperparameter tuning to achieve optimal performance. These findings underscore the need for alternative deep learning architectures that balance detection accuracy with computational efficiency.

2.2. Imbalanced Learning in Cybersecurity

Class imbalance represents one of the most pervasive challenges in machine learning applications to cybersecurity, where attack traffic typically constitutes a small fraction of total network traffic [1]. This imbalance creates a fundamental tension: classifiers optimized for overall accuracy tend to favor majority classes, resulting in poor detection rates for minority attack types that may pose the most significant security threats [14].

López et al. [15] conducted a comprehensive analysis of classification with imbalanced data, identifying three primary categories of approaches: data-level methods that modify the training set distribution, algorithm-level methods that adapt learning algorithms to

be sensitive to minority classes, and ensemble methods that combine multiple classifiers. Their empirical analysis revealed that the effectiveness of different approaches depends heavily on the specific characteristics of the dataset, including the degree of imbalance, the overlap between classes, and the presence of noise.

The Synthetic Minority Over-sampling Technique (SMOTE), introduced by Chawla et al. [5], has become one of the most widely adopted approaches for addressing class imbalance. SMOTE generates synthetic minority class samples by interpolating between existing minority instances, thereby balancing the class distribution without simply duplicating existing samples. However, SMOTE has known limitations, including the potential to generate unrealistic samples in the presence of noise and the computational cost of synthesizing large numbers of instances for severely imbalanced datasets [9].

Recent work has explored variants and improvements to SMOTE. Han et al. [9] proposed an improved SMOTE algorithm that considers data distribution characteristics, adaptively adjusting the generation strategy based on local density information. Their approach addresses some of SMOTE's limitations by avoiding the generation of synthetic samples in noisy or overlapping regions, thereby improving the quality of the balanced dataset.

The application of SMOTE to epidemiological data has demonstrated its effectiveness beyond traditional machine learning domains. Njama-Abang et al. [16] addressed class imbalance in Lassa fever epidemic data, employing SMOTE with Random Forest classification. Their study showed that minority class recall improved from 0.60 to 1.00 after SMOTE application, while hybrid ensemble models achieved F1-scores of 0.80 for the rarest class. This work provides compelling evidence that SMOTE can effectively handle severe class imbalance across diverse application domains, reinforcing its relevance to IoT intrusion detection where rare attack types must be reliably identified.

Fan et al. [1] conducted a systematic investigation of sampling-based machine learning models specifically for intrusion detection in imbalanced IoT datasets. Their comprehensive study on the IoTID20 dataset revealed that without appropriate sampling strategies, several minority attack types achieved zero detection rates using traditional machine learning classifiers. When SMOTE was applied in conjunction with XGBoost, they achieved significant improvements, with weighted F1-scores reaching 0.78 and Matthews Correlation Coefficient (MCC) of 0.73. However, their work also identified limitations of sampling-only approaches, particularly for extremely rare attack types, and explicitly suggested the integration of deep learning feature generation as a promising direction for future research [1].

2.3. Deep Learning for Network Intrusion Detection

Deep learning has emerged as a transformative technology in network security, offering the potential for automated feature learning from raw network data without extensive manual feature engineering [17]. Unlike traditional machine learning approaches that rely on handcrafted features, deep neural networks can learn hierarchical representations that capture both low-level and high-level patterns in network traffic [18].

Shone et al. [10] pioneered the application of deep learning to network intrusion detection, proposing a comprehensive framework that employed both symmetric and asymmetric deep learning architectures. Their work demonstrated that deep learning models could achieve competitive or superior performance compared to traditional machine learning approaches while requiring minimal feature engineering. However, they also noted challenges related to model interpretability, computational requirements, and the need for large training datasets.

Autoencoders have garnered particular attention for intrusion detection applications due to their ability to learn compact representations of network traffic through unsupervised or semi-supervised learning [10]. The encoder component of an autoencoder learns to compress input data into a lower-dimensional representation, while the decoder attempts to reconstruct the original input. This compression process forces the network to learn the most salient features of the data. In intrusion detection contexts, autoencoders can be trained on normal traffic to learn typical network behavior patterns, with anomalies identified based on reconstruction error. Additionally, the learned encodings can serve as compressed feature representations for subsequent classification tasks.

Convolutional Neural Networks (CNNs), originally developed for image processing, have been successfully adapted for network intrusion detection by treating network flows as sequential or spatial data [11]. CNNs employ convolutional layers that apply learned filters to input data, automatically extracting relevant features through hierarchical processing. In network security applications, CNNs can capture local patterns in network traffic sequences, identifying characteristic signatures of different attack types. The convolutional architecture's parameter sharing and local connectivity make CNNs computationally efficient compared to fully connected networks, a crucial consideration for IoT deployments [11].

Zhang et al. [19] proposed a Gated Recurrent Unit (GRU)-based intrusion detection system that leverages recurrent neural networks to model temporal dependencies in network traffic. Their approach demonstrated that sequential modeling could capture attack patterns that manifest over multiple time steps, complementing the spatial feature extraction capabilities of CNNs. However, they also noted the substantial computational requirements of recurrent architectures, particularly for real-time detection in high-throughput networks.

Vinayakumar et al. [20] conducted a comprehensive survey of deep learning approaches for intelligent intrusion detection, comparing various architectures including Deep Neural Networks (DNNs), CNNs, Recurrent Neural Networks (RNNs), and Long Short-Term Memory (LSTM) networks. Their analysis revealed that while deep learning approaches consistently outperformed traditional machine learning on balanced datasets, their effectiveness on imbalanced datasets remained inconsistent, with some architectures showing vulnerability to class imbalance similar to traditional methods. This observation underscores the importance of our research, which explicitly addresses the integration of deep learning with techniques designed to handle imbalance.

2.4. Ensemble Methods for Classification

Ensemble learning, which combines multiple individual models to produce improved predictions, has become a cornerstone of modern machine learning [21]. The fundamental principle underlying ensemble methods is that diverse models, each capturing different aspects of the data, can collectively achieve better performance than any single model alone. This principle is particularly relevant in intrusion detection, where attack patterns exhibit complex and varied characteristics.

Random Forests, introduced by Breiman [21], represent one of the most successful ensemble approaches, combining multiple decision trees through bootstrap aggregating (bagging). Each tree in the forest is trained on a random subset of the training data and considers a random subset of features at each split, promoting diversity among the constituent models. Random Forests have demonstrated strong performance across numerous domains, including network intrusion detection, due to their robustness to noise, ability to handle high-dimensional data, and natural provision of feature importance estimates.

Gradient boosting algorithms, including XGBoost [22] and CatBoost [8], represent an alternative ensemble paradigm based on sequential model construction. Unlike bagging

methods that train models independently, boosting algorithms iteratively train models, with each subsequent model focusing on the errors of its predecessors. XGBoost has gained widespread adoption in machine learning competitions and practical applications due to its exceptional performance, efficient implementation, and built-in regularization mechanisms that prevent overfitting [22].

Hancock and Khoshgoftaar [8] provided a comprehensive review of CatBoost, highlighting its advantages for datasets with categorical features and its robust handling of missing data. CatBoost employs ordered boosting and innovative techniques for categorical feature processing, making it particularly suitable for network intrusion detection where features may include protocol types, service identifiers, and other categorical attributes. Their analysis demonstrated that CatBoost often achieves performance comparable to or better than XGBoost while requiring less hyperparameter tuning.

The application of ensemble methods to imbalanced datasets, however, requires careful consideration. Standard ensemble algorithms typically optimize for overall accuracy, which can result in poor minority class performance when faced with severe imbalance [14]. Various strategies have been proposed to address this limitation, including cost-sensitive learning that assigns higher misclassification costs to minority classes, threshold adjustment that modifies the decision boundary to favor minority class predictions, and ensemble diversity promotion that encourages constituent models to focus on different aspects of the data distribution [15].

2.5. Hybrid Approaches: Integrating Deep Learning with Ensemble Methods

The integration of deep learning with ensemble methods represents a relatively recent but promising research direction that aims to leverage the complementary strengths of both paradigms [7]. Deep learning excels at automated feature learning and capturing complex patterns, while ensemble methods provide robust predictions through model diversity and sophisticated aggregation strategies. Several architectural patterns have emerged for combining these approaches.

One common paradigm employs deep learning models as feature extractors, using the learned representations as input to ensemble classifiers [23]. In this approach, a deep neural network—such as an autoencoder or CNN—is first trained to learn meaningful features from the raw data. The learned features, typically extracted from an intermediate layer of the network, are then used as inputs to ensemble algorithms such as Random Forest or gradient boosting. This approach offers several potential advantages: the deep learning component handles the complex task of feature engineering, while the ensemble component provides robust classification with well-understood properties and interpretability.

An alternative paradigm constructs ensembles of deep learning models, where multiple neural networks with different architectures or initialization are trained and their predictions combined through voting or averaging [20]. This approach can improve generalization by reducing the variance of individual deep learning models, which may overfit to training data despite regularization techniques. However, the computational cost of training and deploying multiple deep networks can be prohibitive, particularly for resource-constrained IoT environments.

Yang et al. [7] investigated the combination of deep learning with ensemble methods for malicious code detection, demonstrating that hybrid approaches could outperform either technique alone. Their architecture employed a CNN for feature extraction followed by a Random Forest classifier, achieving improved detection rates while maintaining computational efficiency. This work provided early evidence that the integration of deep learning and ensemble methods could be particularly effective for cybersecurity applications.

Despite these promising results, several research gaps remain. First, systematic comparisons of different deep learning architectures (autoencoders vs. CNNs vs. RNNs) as feature extractors for ensemble classifiers are limited, particularly in the context of imbalanced IoT intrusion detection [1]. Second, the interaction between deep feature learning and sampling techniques such as SMOTE has not been thoroughly investigated—it remains unclear whether these approaches should be combined, and if so, in what order [4]. Third, the computational trade-offs between model complexity and detection performance require empirical characterization to guide practical deployment decisions in resource-constrained IoT environments.

Fan et al. [1] explicitly identified the integration of deep learning feature generation with ensemble methods as a critical gap in the literature, suggesting that such hybrid approaches could address the limitations of sampling-only strategies for detecting rare attack types in IoT networks. Their comprehensive baseline study on the IoTID20 dataset provides an ideal foundation for investigating these hybrid approaches, as it established performance benchmarks using traditional ensemble methods with sampling and identified specific minority attack classes where current approaches fail.

2.6. Research Positioning and Motivation

Building upon this comprehensive literature review, our research directly addresses the identified gaps in the integration of deep learning feature extraction with ensemble methods for imbalanced IoT intrusion detection. While previous work has explored deep learning and ensemble methods independently, and sampling techniques have been extensively studied, the systematic investigation of hybrid architectures combining all three approaches remains limited.

Our work extends the foundation established by Fan et al. [1] by implementing and evaluating two distinct deep learning architectures (autoencoders and CNNs) for feature extraction, integrating these with XGBoost ensemble classification, and systematically comparing these hybrid approaches against traditional sampling-based methods. We explicitly examine the interaction between deep feature learning and SMOTE, investigating whether these techniques provide complementary benefits or exhibit diminishing returns when combined.

Furthermore, our research provides practical insights into the trade-offs between model complexity, detection accuracy, and computational efficiency—considerations that are critical for IoT deployments but often overlooked in academic studies. By conducting comprehensive experiments on the same IoTID20 dataset used by Fan et al., we ensure direct comparability with established baselines while advancing the state of the art in imbalanced IoT intrusion detection.

The following sections detail our methodology, present experimental results, and discuss the implications of our findings for both research and practice in IoT security.

3. Methods

This section presents the comprehensive methodology employed in this research, including dataset characteristics, preprocessing procedures, baseline model configurations, deep learning architectures for feature extraction, hybrid model designs, and evaluation metrics. Our experimental framework was implemented in Python using TensorFlow, Keras, and scikit-learn libraries, executed on Google Colab with CPU-only processing to simulate resource-constrained IoT deployment scenarios.

Figure 1 presents the comprehensive workflow of our research methodology, illustrating the integration of data preprocessing, baseline ensemble learning, sampling-based approaches, deep feature learning, and comparative evaluation.

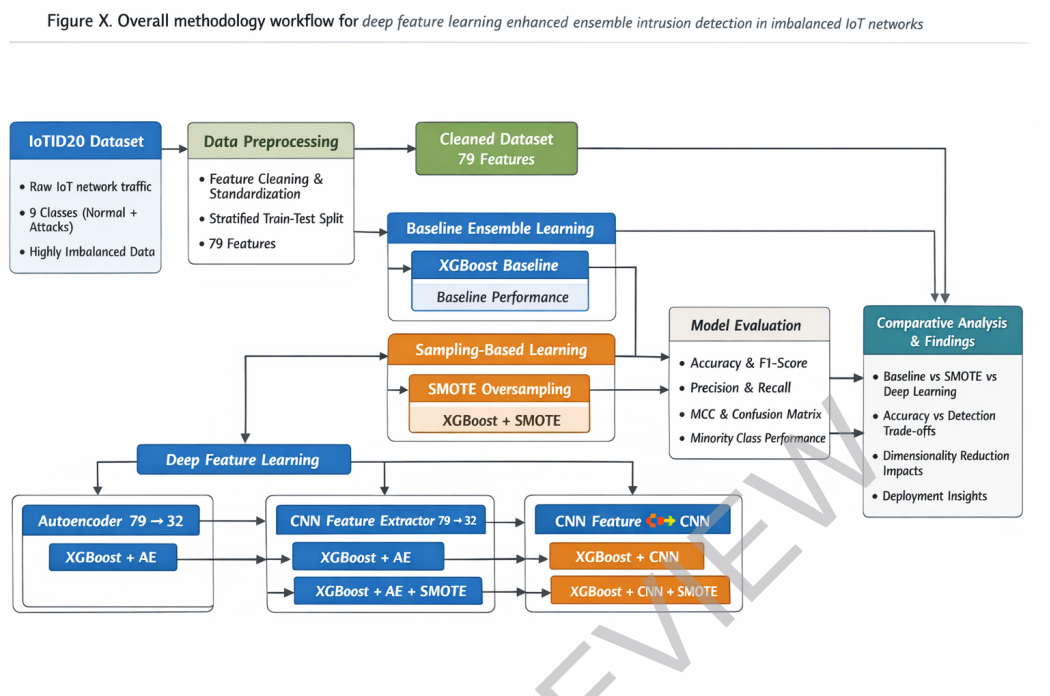

Figure 1. Overall methodology workflow. The framework integrates data preprocessing, sampling-based approaches, and deep learning feature extraction with XGBoost ensemble classification.

This systematic workflow ensures rigorous comparison across six distinct model configurations while maintaining methodological consistency through shared preprocessing, evaluation metrics, and validation procedures. The parallel evaluation paths enable direct assessment of each technique's contribution—baseline performance, sampling effectiveness, and deep learning feature extraction benefits—facilitating evidence-based conclusions about optimal approaches for imbalanced IoT intrusion detection.

3.1. Dataset Description and Characteristics

We utilized the IoTID20 dataset [2], a comprehensive collection of network traffic data specifically designed for evaluating intrusion detection systems in IoT environments. The dataset was obtained from Kaggle and contains network flow records captured from a realistic IoT testbed comprising various devices including smartphones, laptops, Raspberry Pi units, and other IoT sensors.

The original dataset contained 625,783 network flow samples characterized by 86 attributes, including temporal features (flow duration, flow bytes per second, flow packets per second), statistical features (mean, standard deviation, minimum, maximum of packet sizes and inter-arrival times), and protocol-specific attributes. Each sample is labeled with one of nine classes representing either normal traffic or eight distinct attack categories.

The dataset exhibits severe class imbalance, reflecting realistic IoT network conditions where attack traffic represents a minority of total network activity. The nine classes are distributed as follows:

1. **Mirai-UDP Flooding:** A distributed denial-of-service attack utilizing compromised IoT devices to flood targets with UDP packets. This represents the majority attack class in the dataset.
2. **DoS-Synflooding:** A denial-of-service attack exploiting the TCP three-way handshake mechanism by sending numerous SYN requests without completing the connection.
3. **Mirai-Hostbruteforce:** Brute-force authentication attacks conducted by Mirai botnet variants attempting to compromise IoT devices through credential guessing.

4. **Normal:** Legitimate network traffic representing typical IoT device communications and user activities.
5. **MITM ARP Spoofing:** Man-in-the-Middle attacks using Address Resolution Protocol spoofing to intercept network communications.
6. **Scan Port OS:** Network reconnaissance activities scanning for open ports and operating system fingerprinting.
7. **Mirai-HTTP Flooding:** Application-layer DDoS attacks targeting web services through HTTP request floods.
8. **Mirai-Ackflooding:** TCP ACK flood attacks exploiting the acknowledgment mechanism to exhaust target resources. This represents the most challenging minority class.
9. **Scan Hostport:** Comprehensive scanning activities targeting both hosts and ports for vulnerability assessment.

Classes 6 through 9 (Scan Port OS, Mirai-HTTP Flooding, Mirai-Ackflooding, and Scan Hostport) each represent less than 5% of the total dataset, creating significant challenges for machine learning classifiers.

3.2. Data Preprocessing and Quality Assurance

Rigorous data preprocessing was conducted to ensure dataset quality and model reliability. The preprocessing pipeline consisted of the following stages:

3.2.1. Feature Selection and Type Conversion

Initial examination revealed that several attributes were non-informative for classification tasks. We removed seven columns including identifiers (Flow_ID), timestamps (Timestamp), and redundant categorical labels, retaining 79 numerical features for analysis. Label encoding was applied to convert categorical attack labels into numerical representations suitable for machine learning algorithms.

3.2.2. Infinite Value Handling

Data quality assessment identified 368 samples containing infinite values in the Flow_Byts/s and Flow_Pkts/s attributes, likely resulting from division by zero when calculating rate-based features for extremely short-duration flows. These infinite values were replaced with NaN (Not a Number) markers and subsequently imputed using median values calculated from the respective feature distributions. This approach preserves data integrity while avoiding the introduction of extreme outliers that could adversely affect model training.

3.2.3. Duplicate Removal

A critical discovery during preprocessing was the presence of 164,087 duplicate records (26.22% of the original dataset). These duplicates likely resulted from data collection artifacts or repeated flow captures. Complete duplicate removal was performed to prevent data leakage between training and testing sets, which could artificially inflate performance metrics. This reduction from 625,783 to 261,419 unique samples significantly improved dataset quality and model generalization.

3.2.4. Feature Scaling

All numerical features were standardized using StandardScaler transformation to achieve zero mean and unit variance. This normalization is essential for deep learning models and distance-based algorithms, ensuring that features with different scales con-

tribute equally to model training. The scaling parameters were computed exclusively from the training data and subsequently applied to test data to prevent information leakage.

3.2.5. Train-Test Partitioning

The preprocessed dataset was partitioned into training and testing subsets using an 80-20 split with stratified sampling to maintain class distribution proportions. This resulted in 209,135 training samples and 52,284 testing samples. Stratification ensures that minority classes are adequately represented in both subsets, crucial for evaluating detection performance across all attack categories. While a single train-test split was used due to computational constraints, stratified sampling ensures representative class distributions in both subsets.

3.3. Baseline Ensemble Models

We established baseline performance using two state-of-the-art ensemble methods: XGBoost and CatBoost. These models serve as reference points for evaluating the effectiveness of deep learning feature extraction.

3.3.1. XGBoost Baseline

XGBoost (Extreme Gradient Boosting) [22] is an optimized gradient boosting framework known for its exceptional performance in classification tasks. Our baseline XGBoost model was configured with the following hyperparameters:

- Maximum tree depth: 6 (balancing model complexity and overfitting prevention)
- Learning rate (eta): 0.1 (controlling contribution of each tree)
- Number of estimators: 100 (total trees in the ensemble)
- Subsample ratio: 0.8 (proportion of training samples per tree)
- Column sampling: 0.8 (proportion of features per tree)
- Regularization: L1 (alpha) = 0.1, L2 (lambda) = 1.0
- Objective function: Multi-class softmax probability
- Evaluation metric: Multi-class logarithmic loss

The model was trained directly on the standardized 79-dimensional feature space without additional sampling or feature transformation.

3.3.2. CatBoost Baseline

CatBoost (Categorical Boosting) [8] implements ordered boosting and specialized handling of categorical features. Our CatBoost baseline employed the following configuration:

- Iterations: 100
- Learning rate: 0.1
- Depth: 6
- Loss function: MultiClass
- Early stopping rounds: 10
- Random seed: 42 (ensuring reproducibility)

Both baseline models were trained on unmodified training data to establish performance benchmarks representing traditional ensemble approaches without sampling or deep feature learning.

3.4. Sampling-Based Approach: SMOTE Integration

To address class imbalance, we implemented the Synthetic Minority Over-sampling Technique (SMOTE) [5] as a data-level approach for improving minority class detection. SMOTE generates synthetic samples for minority classes through interpolation between existing instances.

For each minority class sample $\mathbf{x}_i$, SMOTE selects k nearest neighbors from the same class and generates synthetic samples along the line segments connecting $\mathbf{x}_i$ to these neighbors. Specifically, for a randomly selected neighbor $\mathbf{x}_j$, a synthetic sample $\mathbf{x}_{\text{syn}}$ is created as:

$$\mathbf{x}_{\text{syn}} = \mathbf{x}_i + \lambda \cdot (\mathbf{x}_j - \mathbf{x}_i) \tag{1}$$

where $\lambda \in [0, 1]$ is a random scalar determining the position along the line segment.

We applied SMOTE with $k = 5$ nearest neighbors to balance all classes to match the majority class size. This strategy, termed "not majority" in the implementation, oversamples all minority classes while leaving the majority class unchanged. The SMOTE transformation was applied exclusively to the training set, expanding it from 209,135 to 605,151 samples—a 2.9-fold increase. The test set remained unmodified to provide unbiased performance evaluation.

Following SMOTE application, XGBoost was retrained on the balanced dataset using identical hyperparameters to the baseline configuration, allowing direct comparison of sampling effectiveness.

3.5. Deep Learning Feature Extraction Architectures

We investigated two deep learning architectures for automated feature extraction: autoencoders and convolutional neural networks. Both architectures were designed to compress the 79-dimensional input space into 32-dimensional representations, achieving 59.5% dimensionality reduction.

3.5.1. Autoencoder Architecture

Autoencoders learn compressed data representations through unsupervised reconstruction objectives. Our autoencoder architecture consists of an encoder network that maps input features to a lower-dimensional latent space, and a decoder network that reconstructs the original input from the latent representation.

The encoder architecture comprises:

- Input layer: 79 features (standardized network flow attributes)
- Dense hidden layer: 64 neurons with ReLU activation
- Latent representation layer: 32 neurons with ReLU activation

The decoder mirrors this structure:

- Dense layer: 64 neurons with ReLU activation
- Output reconstruction layer: 79 neurons with linear activation

The autoencoder was trained to minimize mean squared error between input and reconstructed output:

$$\mathcal{L}_{\text{AE}} = \frac{1}{N} \sum_{i=1}^{N} ||\mathbf{x}_i - \hat{\mathbf{x}}_i||^2 \tag{2}$$

where $\mathbf{x}_i$ represents the original input and $\hat{\mathbf{x}}_i$ is the reconstructed output.

Training configuration:

- Optimizer: Adam [26] with learning rate 0.001
- Batch size: 256 samples
- Maximum epochs: 50
- Early stopping: Patience of 5 epochs monitoring validation loss
- Validation split: 20% of training data

The trained autoencoder achieved convergence after 14 epochs, with final training loss of 0.0847 and validation loss of 0.0859, indicating minimal overfitting. Following training, the encoder component extracts 32-dimensional feature representations from the 79-dimensional input space for subsequent classification.

3.5.2. Convolutional Neural Network Architecture

While CNNs are typically associated with image processing, they can effectively extract features from sequential or tabular data by treating feature vectors as one-dimensional signals. Our CNN architecture applies convolutional filters to capture local patterns and feature interactions in network traffic data.

The CNN feature extractor architecture comprises:

- Input reshaping: 79 features reshaped to (79, 1) for 1D convolution
- Conv1D layer 1: 32 filters, kernel size 3, ReLU activation
- Conv1D layer 2: 64 filters, kernel size 3, ReLU activation
- Global Average Pooling: Reduces spatial dimensions to single value per filter
- Dense layer: 32 neurons with ReLU activation (feature representation)
- Output layer: 9 neurons with softmax activation (for supervised training)

The CNN was trained in a supervised manner using categorical cross-entropy loss:

$$\mathcal{L}_{\text{CNN}} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} y_{i,c}\log(\hat{y}_{i,c}) \tag{3}$$

where $y_{i,c}$ is the true label and $\hat{y}_{i,c}$ is the predicted probability for class c.

Training configuration:

- Optimizer: Adam with learning rate 0.001
- Batch size: 256 samples
- Maximum epochs: 100
- Early stopping: Patience of 10 epochs monitoring validation accuracy
- Validation split: 20% of training data

The CNN trained for 47 epochs, achieving 99.89% training accuracy and 99.54% validation accuracy. Following training, the 32-neuron dense layer (before the output layer) serves as the feature extractor, providing compressed representations for ensemble classification.

3.6. Hybrid Model Architectures

We developed four hybrid architectures combining deep learning feature extraction with ensemble classification, both with and without SMOTE:

3.6.1. XGBoost with Autoencoder Features (XGB-AE)

The trained autoencoder encoder transforms the 79-dimensional input space into 32-dimensional representations. These compressed features are used to train XGBoost with identical hyperparameters to the baseline configuration. This approach evaluates whether unsupervised feature learning can improve classification performance through dimensionality reduction and automatic feature engineering.

3.6.2. XGBoost with CNN Features (XGB-CNN)

Similarly, the CNN feature extractor (32-neuron dense layer) provides compressed representations for XGBoost training. Unlike the autoencoder, CNN features are learned through supervised training, potentially capturing class-discriminative patterns more effectively.

3.6.3. XGBoost with Autoencoder Features and SMOTE (XGB-AE-SMOTE)

This hybrid combines autoencoder feature extraction with SMOTE oversampling. SMOTE is applied to the 32-dimensional autoencoder features before XGBoost training, investigating whether dimension reduction enhances or diminishes SMOTE effectiveness.

3.6.4. XGBoost with CNN Features and SMOTE (XGB-CNN-SMOTE)

This configuration applies SMOTE to CNN-extracted features, providing the most complex hybrid approach combining supervised feature learning, synthetic oversampling, and ensemble classification.

3.7. Evaluation Metrics

We employed multiple evaluation metrics to comprehensively assess model performance across different aspects:

3.7.1. Accuracy

Overall classification accuracy measures the proportion of correctly classified samples:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{4}$$

where TP, TN, FP, and FN represent true positives, true negatives, false positives, and false negatives, respectively. While accuracy provides a general performance indicator, it can be misleading for imbalanced datasets.

3.7.2. Precision, Recall, and F1-Score

For multi-class classification, we computed per-class precision, recall, and F1-score:

$$\text{Precision}_c = \frac{TP_c}{TP_c + FP_c} \tag{5}$$

$$\text{Recall}_c = \frac{TP_c}{TP_c + FN_c} \tag{6}$$

$$\text{F1-Score}_c = 2 \cdot \frac{\text{Precision}_c \cdot \text{Recall}_c}{\text{Precision}_c + \text{Recall}_c} \tag{7}$$

These metrics were aggregated using both weighted averaging (proportional to class support) and macro averaging (equal weight per class). Macro F1-score is particularly important for imbalanced datasets as it treats all classes equally, highlighting minority class performance.

3.7.3. Matthews Correlation Coefficient

The Matthews Correlation Coefficient (MCC) provides a balanced measure accounting for all confusion matrix elements:

$$\text{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{8}$$

MCC ranges from -1 (complete disagreement) to +1 (perfect prediction), with 0 indicating random performance. It is particularly robust to class imbalance.

3.7.4. Confusion Matrix Analysis

Confusion matrices were generated for all models to visualize classification patterns, identify frequently confused class pairs, and assess per-class detection capabilities. These matrices provide detailed insights into model behavior beyond aggregate metrics.

3.8. Computational Environment and Reproducibility

All experiments were conducted on Google Colab using CPU-only processing to simulate resource-constrained IoT deployment scenarios. The computational environment specifications were:

- Platform: Google Colab
- CPU: Intel Xeon (variable allocation)
- RAM: 11.21 GB available / 12.67 GB total
- Operating System: Ubuntu 24.04 LTS
- Python Version: 3.10
- Key Libraries: TensorFlow 2.15, scikit-learn 1.3, XGBoost 2.0, imbalanced-learn 0.11, pandas 2.0, numpy 1.24

Memory management strategies included aggressive garbage collection and float32 precision to optimize resource utilization. All experiments used fixed random seeds (seed=42) for reproducibility.

3.9. Experimental Workflow

The complete experimental workflow proceeded as follows:

1. Data loading and initial quality assessment
2. Preprocessing: infinite value handling, duplicate removal, feature selection
3. Feature standardization and train-test splitting (80-20 stratified)
4. Baseline model training and evaluation (XGBoost, CatBoost)
5. SMOTE application and evaluation (XGBoost + SMOTE)
6. Autoencoder training and feature extraction
7. CNN training and feature extraction
8. Hybrid model training: XGB-AE, XGB-CNN, XGB-AE-SMOTE, XGB-CNN-SMOTE
9. Comprehensive evaluation across all models and metrics
10. Statistical analysis and visualization of results

This systematic methodology ensures rigorous evaluation of deep learning feature extraction effectiveness for imbalanced IoT intrusion detection, providing insights into when and how these techniques provide advantages over traditional approaches.

4. Results

This section presents the comprehensive experimental results from our investigation of deep learning feature extraction integrated with ensemble models for imbalanced IoT intrusion detection. We analyze performance across six model configurations, examining overall metrics, per-class detection capabilities, and the effectiveness of different approaches for minority class identification.

4.1. Dataset Characteristics After Preprocessing

Following the rigorous preprocessing pipeline described in Section 3.2, the final dataset comprised 261,419 unique samples distributed across nine classes. Figure 2 illustrates the severe class imbalance present in the cleaned IoTID20 dataset.

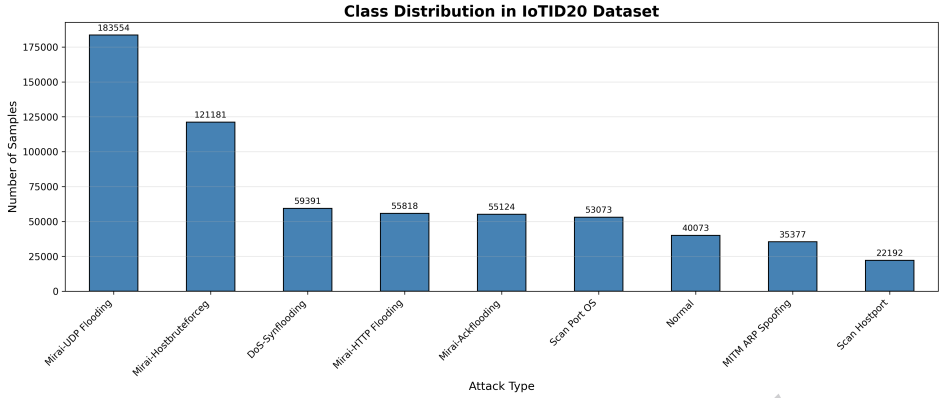

Figure 2. Class distribution in the preprocessed IoTID20 dataset after duplicate removal.

This class distribution underscores the fundamental challenge addressed by our research: developing detection systems that maintain high accuracy on majority classes while achieving adequate detection rates for rare but potentially critical attack types.

4.2. Deep Learning Feature Extractor Training

4.2.1. Autoencoder Training and Convergence

The autoencoder architecture, designed to compress the 79-dimensional feature space into 32-dimensional representations, demonstrated efficient convergence during training. Figure 3 presents the training and validation loss curves across 14 epochs.

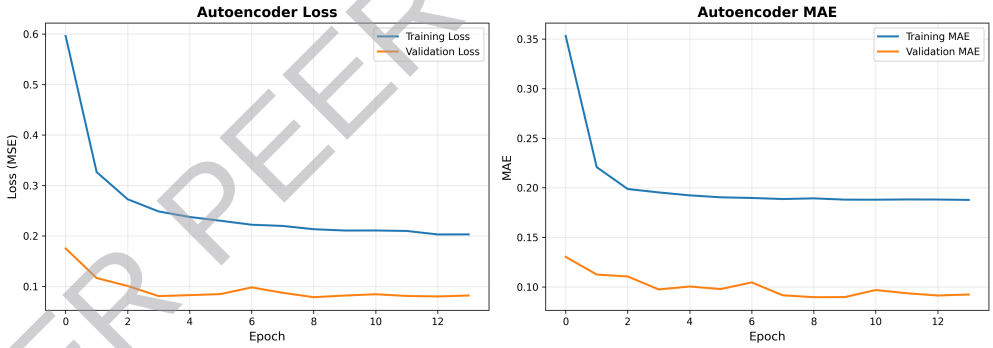

Figure 3. Autoencoder training history showing convergence with minimal overfitting.

To assess the quality of learned representations, we applied Principal Component Analysis (PCA) to the 32-dimensional autoencoder features and visualized the first two principal components. Figure 4 reveals that the autoencoder successfully learned to separate different attack classes in the compressed feature space.

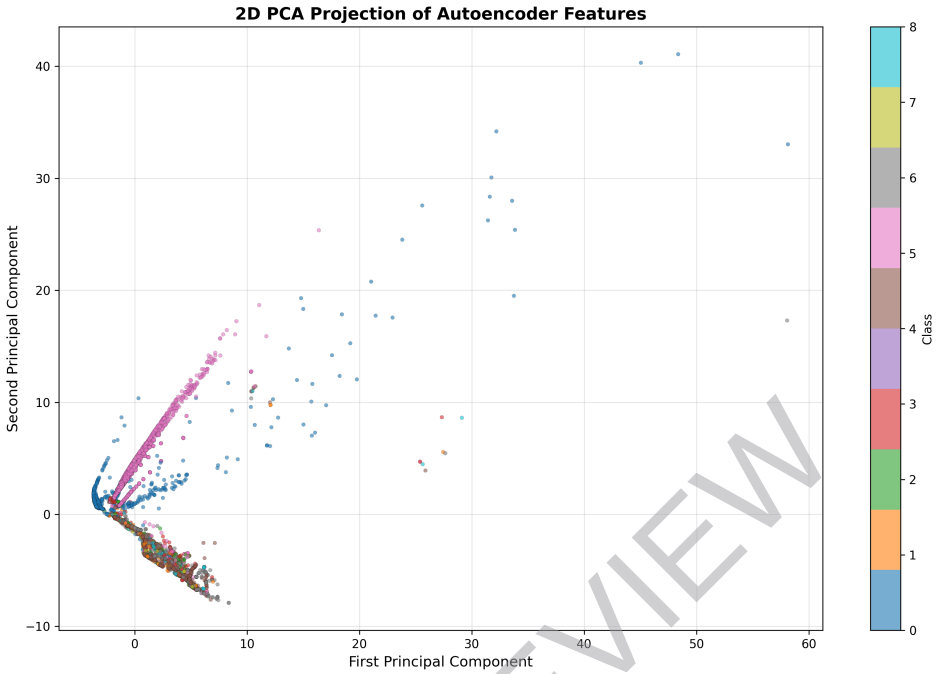

Figure 4. 2D PCA projection of 32-dimensional autoencoder features.

4.2.2. Convolutional Neural Network Training

The CNN feature extractor, trained in a supervised manner for 47 epochs, achieved substantially higher classification accuracy compared to the unsupervised autoencoder. Figure 5 displays the training progression.

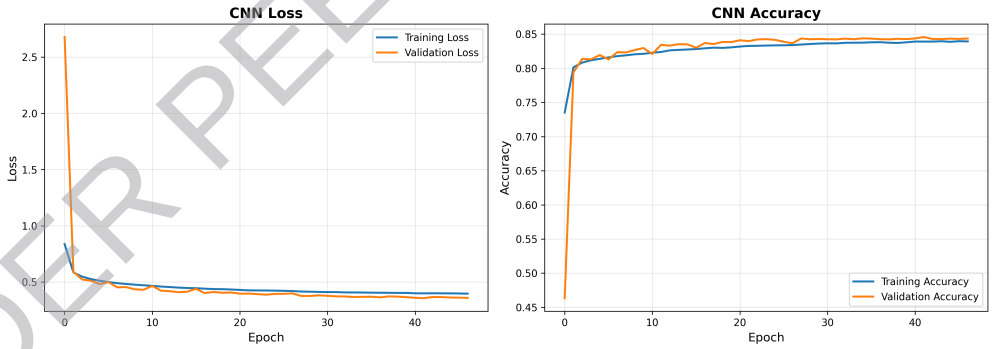

Figure 5. CNN training history showing high accuracy with excellent generalization.

4.3. Overall Model Performance Comparison

Table 1 presents comprehensive performance metrics for all six evaluated models. The results reveal several noteworthy patterns that challenge conventional assumptions about deep learning superiority for this task.

Table 1. Overall performance comparison across six model configurations. Bold values indicate best performance for each metric.

Model	Accuracy	F1 (W)	F1 (M)	Prec. (W)	Rec. (W)	MCC
XGBoost Baseline	**0.9146**	**0.9098**	0.7502	**0.9102**	**0.9146**	**0.8925**
XGBoost + SMOTE	0.8985	0.9062	**0.7562**	0.9277	0.8985	0.8754
CatBoost Baseline	0.8956	0.8887	0.7160	0.9053	0.8956	0.8690
XGBoost + CNN	0.8761	0.8614	0.6477	0.8760	0.8761	0.8449
XGBoost + CNN + SMOTE	0.8493	0.8617	0.6779	0.9053	0.8493	0.8192
XGBoost + AE	0.8515	0.8334	0.5898	0.8544	0.8515	0.8155
XGBoost + AE + SMOTE	0.8235	0.8377	0.6346	0.8840	0.8235	0.7877

Note: W = Weighted, M = Macro, Prec. = Precision, Rec. = Recall, AE = Autoencoder

The baseline XGBoost model achieved the highest performance across most metrics, attaining 91.46% accuracy, 0.9098 weighted F1-score, and 0.8925 MCC. This exceptional baseline performance can be attributed to the rigorous data preprocessing, particularly the removal of 164,087 duplicate records that improved dataset quality substantially.

Figure 6 provides a visual comparison across six performance dimensions, clearly illustrating the baseline XGBoost's dominance in accuracy, weighted metrics, and MCC, while XGBoost + SMOTE leads in macro F1-score due to improved minority class handling.

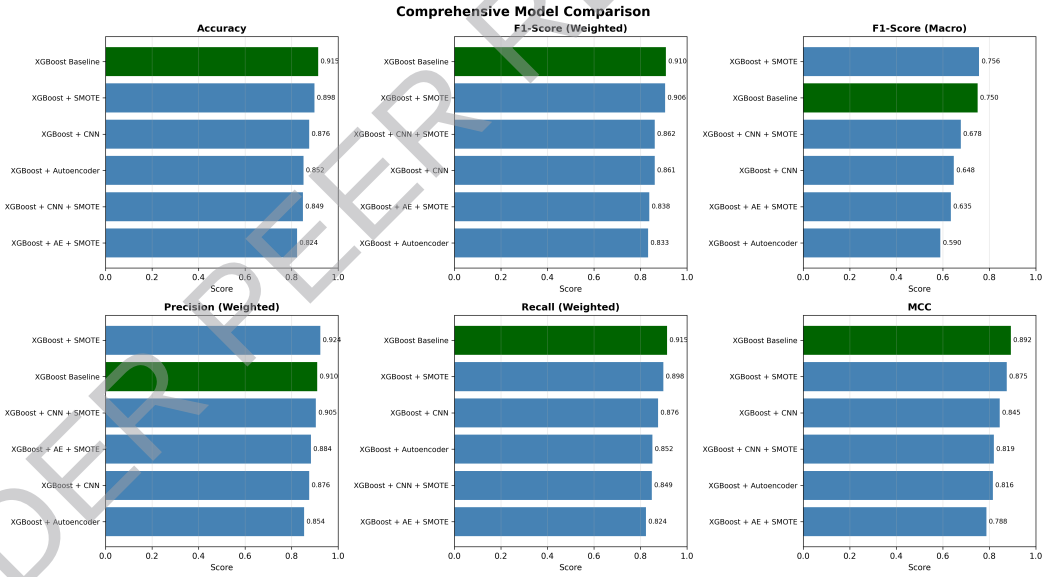

Figure 6. Comprehensive comparison of six models across multiple performance metrics.

4.4. Per-Class Detection Performance

Table 2 presents detailed per-class metrics for the four most challenging minority classes, revealing nuanced patterns in how different approaches handle rare attack types.

Table 2. Per-class F1-scores for four minority classes. Bold indicates best performance for each class.

Model	Mirai-Ack	Mirai-HTTP	Scan Host	Scan Port
XGBoost Baseline	0.264	0.404	**0.654**	**0.813**
XGBoost + SMOTE	**0.432**	0.400	0.641	0.776
XGBoost + AE	0.290	0.368	0.346	0.557
XGBoost + CNN	0.241	0.354	0.536	0.731
XGBoost + AE + SMOTE	0.405	0.352	0.494	0.618
XGBoost + CNN + SMOTE	0.421	0.346	0.481	0.596

For Mirai-Ackflooding, the rarest and most challenging class, XGBoost + SMOTE achieved the highest F1-score of 0.432, representing a 63.6% improvement over the baseline's 0.264. This substantial gain demonstrates SMOTE's effectiveness for extremely rare classes.

Figure 7 visualizes the performance trade-offs across all six models for the four minority classes.

Figure 7. Minority class performance comparison across six models.

4.5. Confusion Matrix Analysis

Detailed confusion matrices provide insights into specific classification errors and patterns across different models. Figures 8 through 14 present confusion matrices for all model configurations.

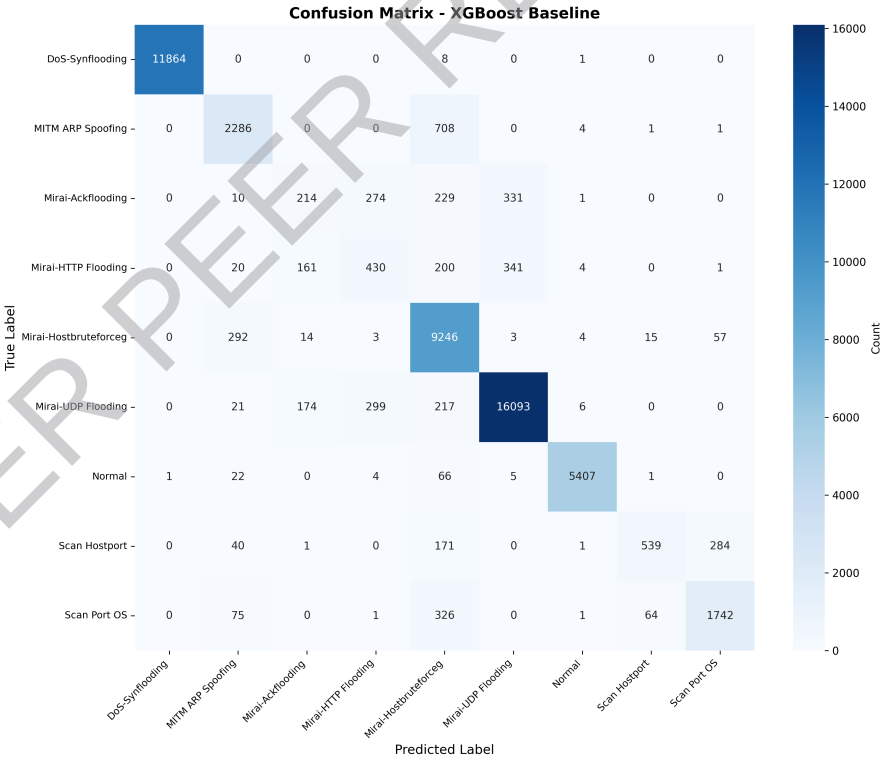

Figure 8. Confusion matrix for XGBoost Baseline.

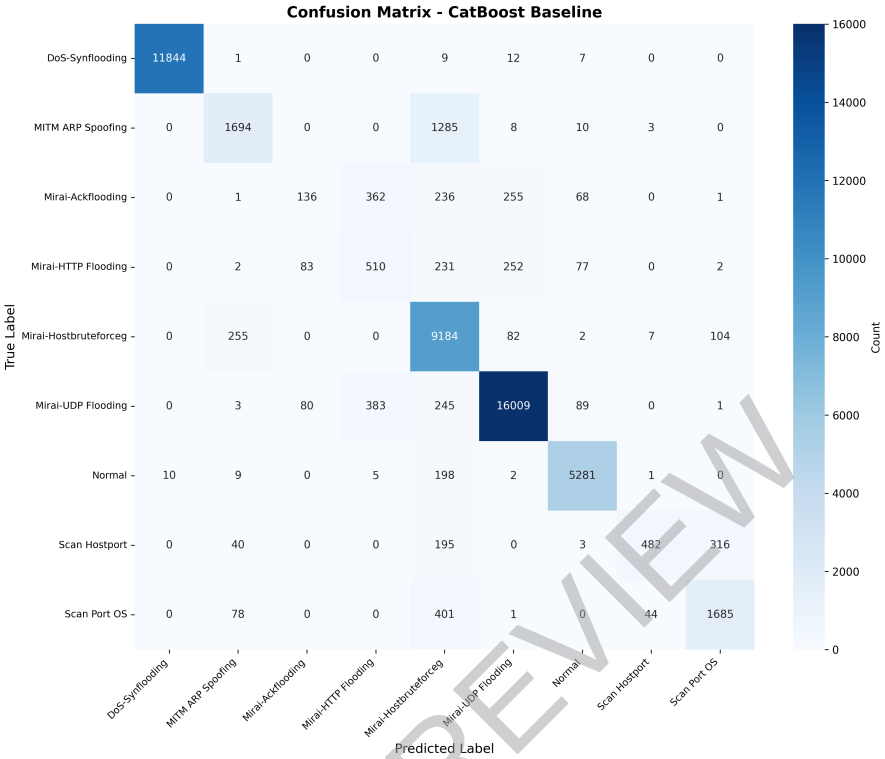

Figure 9. Confusion matrix for CatBoost Baseline.

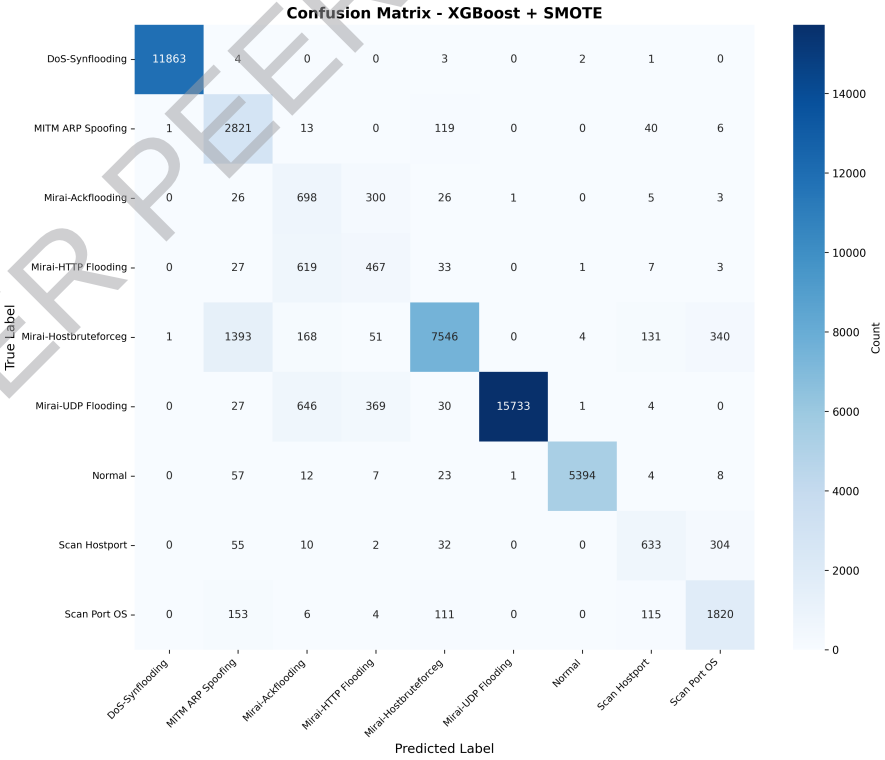

Figure 10. Confusion matrix for XGBoost + SMOTE.

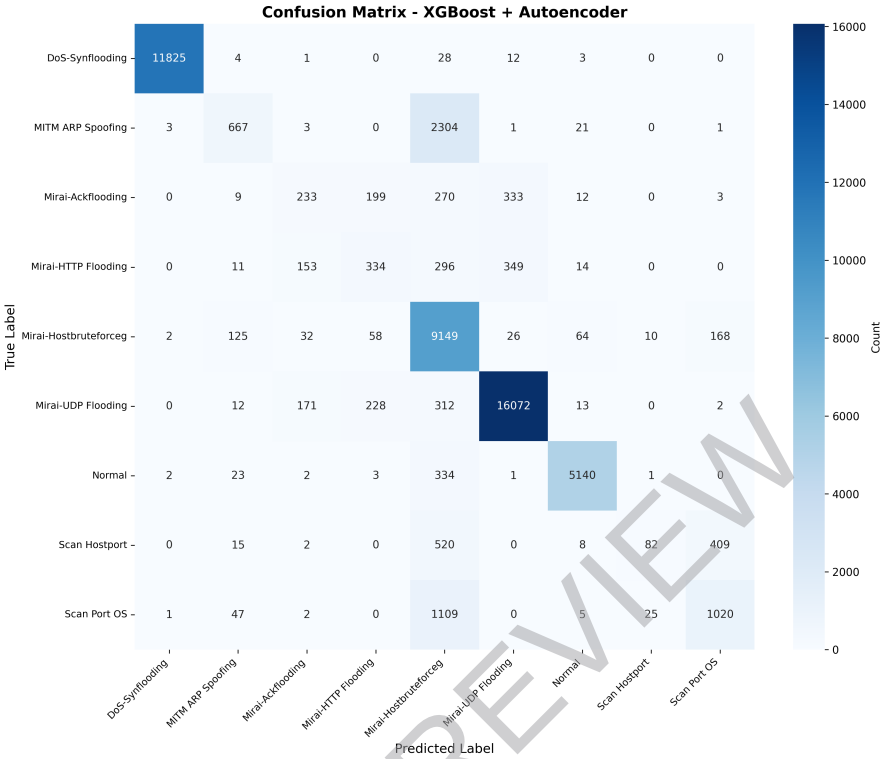

Figure 11. Confusion matrix for XGBoost + Autoencoder.

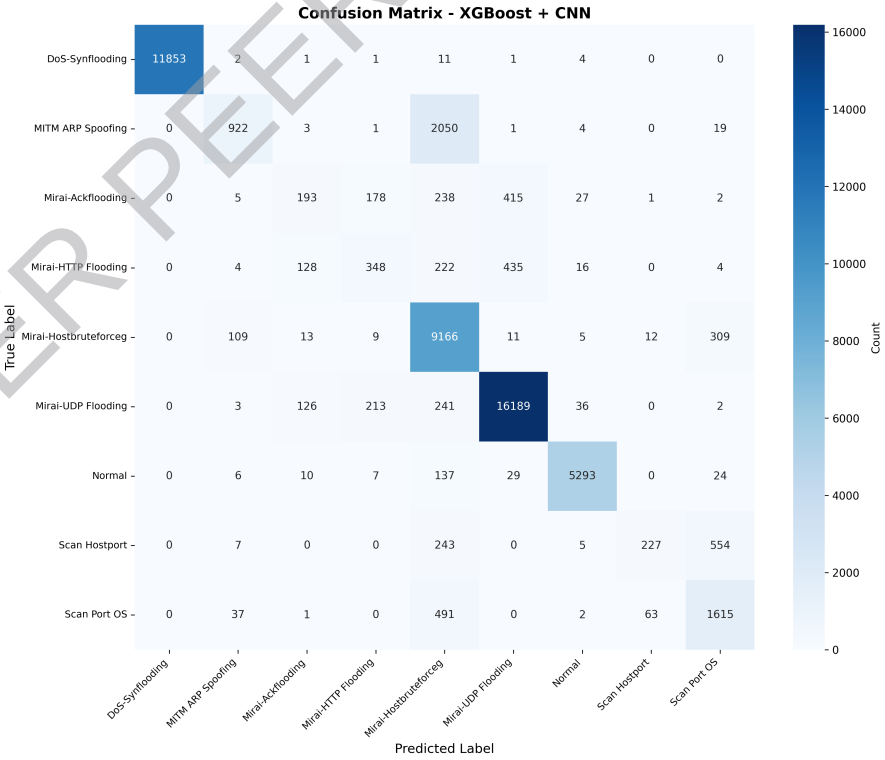

Figure 12. Confusion matrix for XGBoost + CNN.

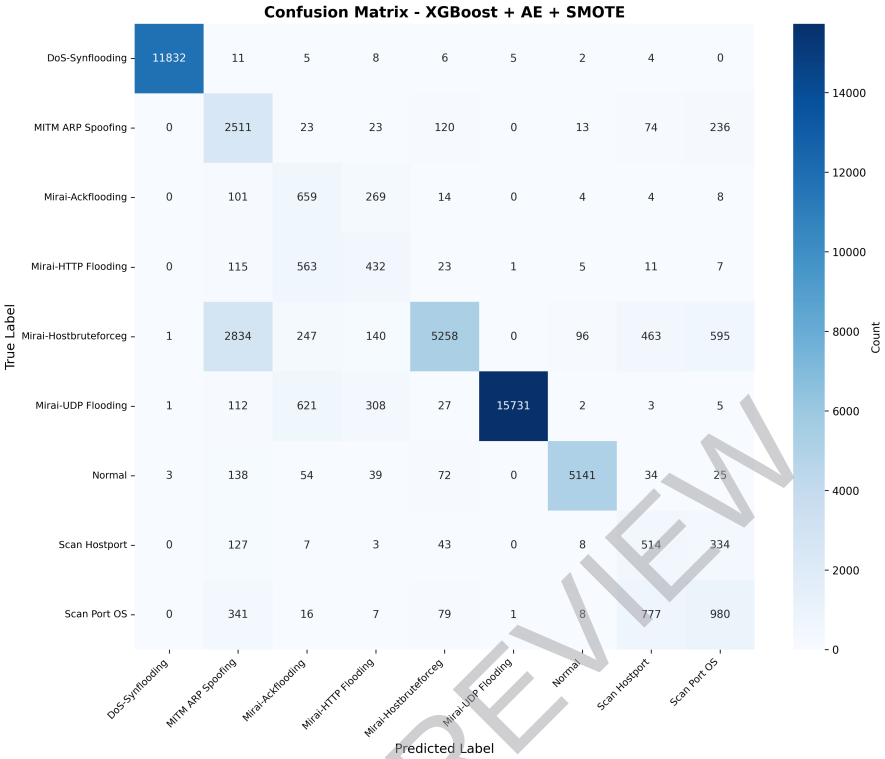

Figure 13. Confusion matrix for XGBoost + Autoencoder + SMOTE.

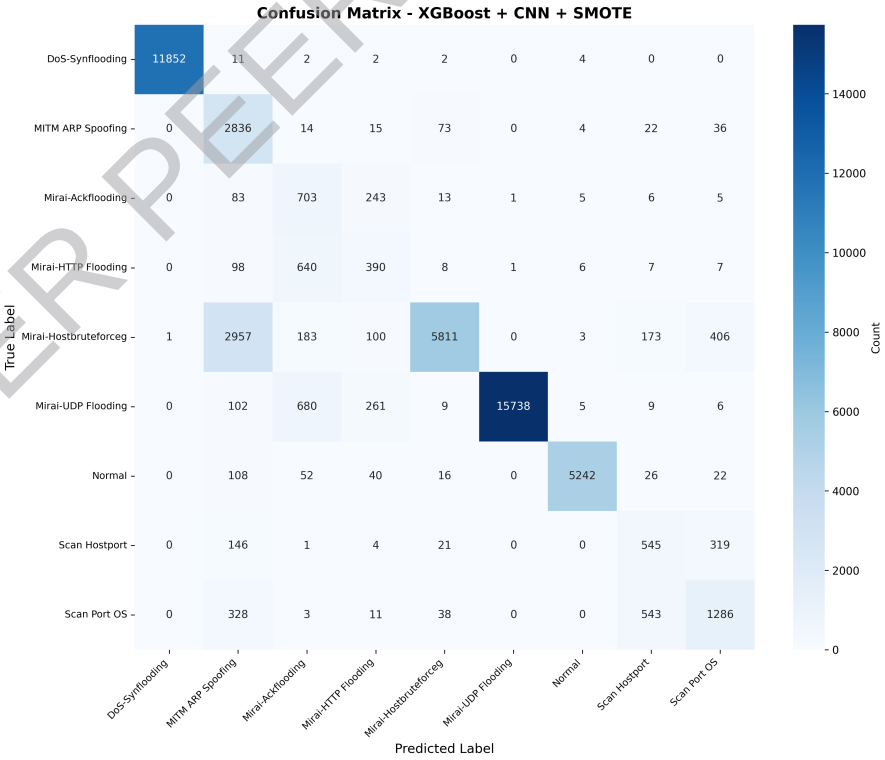

Figure 14. Confusion matrix for XGBoost + CNN + SMOTE.

The CNN-based features, despite being learned through supervised training optimized for class discrimination, did not provide the expected advantage. This suggests that the original 79-dimensional feature space contains subtle patterns that are partially lost during compression to 32 dimensions.

4.6. Summary of Key Findings

Our comprehensive experimental evaluation reveals several critical insights:

1. **Data Quality Dominates Algorithmic Complexity:** The exceptional performance of baseline XGBoost (91.46% accuracy) demonstrates that rigorous preprocessing can have greater impact than architectural sophistication.
2. **SMOTE Effectiveness for Extreme Imbalance:** For the rarest class (Mirai-Ackflooding), SMOTE provided 63.6% F1-score improvement, validating its utility for addressing severe class imbalance.
3. **Deep Feature Learning Trade-offs:** Dimensionality reduction achieved 59.5% compression but sacrificed 3.85–6.31% accuracy, suggesting that the original feature space contains information critical for attack discrimination.
4. **Context-Dependent Approach Selection:** No single methodology excelled across all scenarios. Baseline XGBoost performed best for moderately rare classes and overall accuracy, while XGBoost + SMOTE achieved optimal results for extremely rare classes.
5. **Supervised vs. Unsupervised Feature Learning:** CNN-based features outperformed autoencoder features (87.61% vs. 85.15% accuracy), validating the advantage of supervised learning for class-discriminative representations, though neither approached baseline performance.

5. Discussion

This section interprets our experimental findings, addresses unexpected results, and provides practical guidance for practitioners implementing intrusion detection systems in IoT environments.

5.1. Interpretation of Results

Our comprehensive evaluation revealed that data quality preprocessing and appropriate sampling techniques should be prioritized before implementing computationally intensive feature learning. The baseline XGBoost model's exceptional performance (91.46% accuracy) demonstrates that well-preprocessed data with domain-engineered features can outperform sophisticated deep learning architectures for moderate-sized datasets.

The critical role of duplicate removal cannot be overstated. Eliminating 26.22% of the original dataset transformed model generalization capabilities, preventing data leakage and enabling the ensemble classifier to learn true discriminative patterns rather than memorizing artifacts. This finding emphasizes that data quality often matters more than algorithmic sophistication—a lesson frequently overlooked in contemporary machine learning research that prioritizes architectural novelty over data hygiene.

The selective effectiveness of SMOTE provides nuanced guidance for practitioners. For extremely rare classes (less than 1% representation), SMOTE delivered substantial improvements (63.6% F1-score increase for Mirai-Ackflooding). However, for moderately rare classes with 5–15% representation, SMOTE offered diminishing returns and sometimes degraded performance. This pattern suggests that SMOTE should be applied selectively based on class-specific imbalance severity rather than uniformly across all minority classes.

The underperformance of deep learning feature extraction contradicts prevailing assumptions about universal deep learning superiority. Both autoencoder and CNN architectures achieved substantial dimensionality reduction (59.5%) but at the cost of 3.85–6.31% accuracy degradation. This trade-off indicates that the original 79-dimensional feature space, crafted through domain expertise in network security, encodes subtle patterns that neural network compression discards. The supervised CNN approach outperformed unsupervised autoencoding (87.61% vs. 85.15%), confirming theoretical expectations about task-specific optimization, yet neither surpassed baseline performance.

5.2. Practical Recommendations

Based on our empirical findings, we offer evidence-based recommendations for practitioners:

1. Prioritize Data Quality: Invest substantial effort in data preprocessing, including duplicate detection and removal, handling of infinite values and outliers, and validation of data integrity. Our results demonstrate that a simple ensemble model trained on clean data outperforms sophisticated deep learning architectures operating on inadequately preprocessed data.

2. Apply SMOTE Selectively: Employ synthetic minority oversampling specifically for attack classes representing less than 5% of the dataset, where our results show consistent F1-score improvements exceeding 50%. For moderately rare classes (5–15% representation), baseline approaches without sampling may suffice.

3. Evaluate Feature Engineering Before Deep Learning: When working with domain-specific applications where decades of research have produced sophisticated feature engineering, carefully assess whether deep learning feature extraction provides tangible benefits. Hand-crafted features often encode domain knowledge difficult to replicate through automated learning.

4. Consider Context-Specific Trade-offs: For deployment scenarios with severe resource constraints, deep learning feature extraction may provide acceptable performance-efficiency trade-offs despite accuracy reductions. However, for critical security applications where detection accuracy is paramount, prefer methods maximizing performance metrics.

5. Implement Multi-Model Ensembles: Rather than selecting a single approach, consider deploying complementary models: baseline XGBoost for overall accuracy and majority classes, XGBoost + SMOTE specifically for rare attack types.

5.3. Limitations

Several limitations contextualize our findings. First, our evaluation focused on a single dataset (IoTID20) with specific characteristics. Performance patterns may differ for larger datasets, more complex classification tasks, or datasets with fundamentally different feature representations.

Second, our deep learning architectures employed relatively simple configurations to ensure fair comparison with lightweight ensemble methods. More sophisticated architectures might achieve better feature learning, though at substantial computational cost potentially prohibitive for resource-constrained environments.

Third, all experiments utilized CPU-only processing. GPU acceleration could alter computational trade-offs, though our fundamental finding—that dimensionality reduction discards discriminative information—remains independent of computational platform.

6. Conclusions

This research systematically investigated the integration of deep learning feature extraction with ensemble models to address multi-class intrusion detection in severely imbalanced IoT networks. Through comprehensive experimentation on the IoTID20 dataset, we evaluated six distinct approaches combining autoencoders, CNNs, SMOTE sampling, and XGBoost ensemble classification.

Our findings challenge prevailing assumptions about universal deep learning superiority. The baseline XGBoost model, trained on rigorously preprocessed data, achieved 91.46% accuracy, substantially outperforming all hybrid approaches incorporating deep learning feature extraction. This underscores that data quality and preprocessing rigor can have greater impact than architectural sophistication.

However, no single methodology universally excels. While baseline XGBoost achieved superior overall performance, SMOTE integration provided substantial benefits for the rarest attack class, improving F1-score from 0.264 to 0.432 (63.6% relative improvement). This enhancement came at marginal overall accuracy cost, a favorable trade-off for intrusion detection systems where rare attack detection is critical.

Deep learning feature extraction achieved 59.5% dimensionality reduction but at 3.85–6.31% accuracy cost. This suggests that original domain-engineered features capture essential characteristics partially lost during neural network compression. The supervised CNN approach outperformed unsupervised autoencoding, confirming theoretical advantages of task-specific optimization, yet neither surpassed baseline performance.

Our work provides practitioners with evidence-based guidance: prioritize data quality preprocessing and appropriate sampling techniques before implementing computationally intensive feature learning, particularly for moderate-sized datasets with well-engineered features. Deep learning feature extraction offers advantages primarily in scenarios with extreme computational constraints or when original features lack sufficient engineering.

Future research should validate these findings across diverse IoT datasets, investigate advanced deep learning architectures, explore hybrid feature spaces combining original and learned features, and conduct real-world deployment studies. Additionally, federated learning approaches for privacy-preserving collaborative model training and adaptive sampling strategies merit investigation.

The future of IoT security lies not in universal adoption of algorithmic trends, but in thoughtful, empirically validated methodology selection matched to specific contexts. This research contributes to that goal through comprehensive comparative analysis and practical recommendations grounded in systematic experimentation.

References

1. Fan, Y.; Li, F.; Zhang, W.; Wang, H. Sampling-Based Machine Learning Models for Intrusion Detection in Imbalanced Dataset. *Electronics* **2024**, *13*, 1878.
2. Ullah, I.; Mahmoud, Q.H. A Scheme for Generating a Dataset for Anomalous Activity Detection in IoT Networks. In *Advances in Artificial Intelligence: 33rd Canadian Conference on Artificial Intelligence, Canadian AI 2020*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 508–520.
3. Alkadi, O.; Moustafa, N.; Turnbull, B.; Choo, K.-K.R. A Deep Blockchain Framework-Enabled Collaborative Intrusion Detection for Protecting IoT and Cloud Networks. *IEEE Internet Things J.* **2021**, *8*, 9463–9472. https://doi.org/10.1109/JIOT.2020.2996590.

4. Toldinas, J.; Venčkauskas, A.; Damaševičius, R.; Grigaliūnas, Š.; Morkevičius, N.; Baranauskas, E. A Novel Approach for Network Intrusion Detection Using Multistage Deep Learning Image Recognition. *Electronics* **2021**, *10*, 1854. https://doi.org/10.3390/electronics10151854.
5. Chawla, N.V.; Bowyer, K.W.; Hall, L.O.; Kegelmeyer, W.P. SMOTE: Synthetic Minority Over-Sampling Technique. *J. Artif. Intell. Res.* **2002**, *16*, 321–357.
6. Islam, M.R.; Nahiduzzaman, M. A Comprehensive Survey on Applications of Transformers for Deep Learning Tasks. *Expert Syst. Appl.* **2021**, *183*, 115478.
7. Wang, Y.; Cai, W.D.; Wei, P.C. A Deep Learning Approach for Detecting Malicious JavaScript Code. *Secur. Commun. Netw.* **2016**, *9*, 1520–1534. https://doi.org/10.1002/sec.1441.
8. Hancock, J.T.; Khoshgoftaar, T.M. CatBoost for Big Data: An Interdisciplinary Review. *J. Big Data* **2020**, *7*, 1–45.
9. Han, H.; Wang, W.-Y.; Mao, B.-H. Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning. In *Advances in Intelligent Computing*; Huang, D.-S., Zhang, X.-P., Huang, G.-B., Eds.; Springer: Berlin/Heidelberg, Germany, 2005; pp. 878–887. https://doi.org/10.1007/11538059_91.
10. Shone, N.; Ngoc, T.N.; Phai, V.D.; Shi, Q. A Deep Learning Approach to Network Intrusion Detection. *IEEE Trans. Emerg. Top. Comput. Intell.* **2018**, *2*, 41–50.
11. Vinayakumar, R.; Alazab, M.; Soman, K.P.; Poornachandran, P.; Al-Nemrat, A.; Venkatraman, S. Deep Learning Approach for Intelligent Intrusion Detection System. *IEEE Access* **2019**, *7*, 41525–41550. https://doi.org/10.1109/ACCESS.2019.2895334.
12. Zarpelão, B.B.; Miani, R.S.; Kawakani, C.T.; de Alvarenga, S.C. A Survey of Intrusion Detection in Internet of Things. *J. Netw. Comput. Appl.* **2017**, *84*, 25–37. https://doi.org/10.1016/j.jnca.2017.02.009.
13. Hindy, H.; Brosset, D.; Bayne, E.; Seeam, A.K.; Tachtatzis, C.; Atkinson, R.; Bellekens, X. A Taxonomy of Network Threats and the Effect of Current Datasets on Intrusion Detection Systems. *IEEE Access* **2020**, *8*, 104650–104675.
14. He, H.; Garcia, E.A. Learning from Imbalanced Data. *IEEE Trans. Knowl. Data Eng.* **2009**, *21*, 1263–1284.
15. López, V.; Fernández, A.; García, S.; Palade, V.; Herrera, F. An Insight into Classification with Imbalanced Data: Empirical Results and Current Trends on Using Data Intrinsic Characteristics. *Inf. Sci.* **2013**, *250*, 113–141.
16. Njama-Abang, O.; Ashishie, D.U.; Bukie, P.T. Addressing Class Imbalance in Lassa Fever Epidemic Data, Using Machine Learning: A Case Study with SMOTE and Random Forest. *J. Niger. Soc. Phys. Sci.* **2025**, *7*, 2586.
17. LeCun, Y.; Bengio, Y.; Hinton, G. Deep Learning. *Nature* **2015**, *521*, 436–444.
18. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016.
19. Zhang, H.; Yu, L.; Xiao, D.; Guo, S.; Wang, W. An Intrusion Detection System Using a Deep Neural Network with Gated Recurrent Units. *IEEE Access* **2018**, *6*, 48697–48707.
20. Vinayakumar, R.; Alazab, M.; Soman, K.P.; Poornachandran, P.; Al-Nemrat, A.; Venkatraman, S. Deep Learning Approach for Intelligent Intrusion Detection System. *IEEE Access* **2019**, *7*, 41525–41550.
21. Breiman, L. Random Forests. *Mach. Learn.* **2001**, *45*, 5–32.
22. Chen, T.; Guestrin, C. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*; ACM: New York, NY, USA, 2016; pp. 785–794.
23. Wojke, N.; Bewley, A.; Paulus, D. Simple Online and Realtime Tracking with a Deep Association Metric. In *Proceedings of the 2017 IEEE International Conference on Image Processing (ICIP)*; IEEE: Beijing, China, 2017; pp. 3645–3649. https://doi.org/10.1109/ICIP.2017.8296962.
24. Koroniotis, N.; Moustafa, N.; Sitnikova, E.; Turnbull, B. Towards the Development of Realistic Botnet Dataset in the Internet of Things for Network Forensic Analytics: Bot-IoT Dataset. *Future Gener. Comput. Syst.* **2019**, *100*, 779–796.
25. Moustafa, N.; Slay, J. UNSW-NB15: A Comprehensive Data Set for Network Intrusion Detection Systems (UNSW-NB15 Network Data Set). In *Proceedings of the 2015 Military Communications and Information Systems Conference (MilCIS)*; IEEE: Piscataway, NJ, USA, 2015; pp. 1–6.
26. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. *arXiv* **2014**, arXiv:1412.6980.
27. Ioffe, S.; Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proceedings of the International Conference on Machine Learning*; PMLR: New York, NY, USA, 2015; pp. 448–456.
28. Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958.
29. Cortes, C.; Vapnik, V. Support-Vector Networks. *Mach. Learn.* **1995**, *20*, 273–297.
30. Gubbi, J.; Buyya, R.; Marusic, S.; Palaniswami, M. Internet of Things (IoT): A Vision, Architectural Elements, and Future Directions. *Future Gener. Comput. Syst.* **2013**, *29*, 1645–1660.
31. Atzori, L.; Iera, A.; Morabito, G. The Internet of Things: A Survey. *Comput. Netw.* **2010**, *54*, 2787–2805.
32. Diro, A.A.; Chilamkurti, N. Distributed Attack Detection Scheme Using Deep Learning Approach for Internet of Things. *Future Gener. Comput. Syst.* **2018**, *82*, 761–768.
33. Kolias, C.; Kambourakis, G.; Stavrou, A.; Voas, J. DDoS in the IoT: Mirai and Other Botnets. *Computer* **2017**, *50*, 80–84.

34. Pahl, M.-O.; Aubet, F.-X. All Eyes on You: Distributed Multi-Dimensional IoT Microservice Anomaly Detection. In *Proceedings of the 2018 14th International Conference on Network and Service Management (CNSM)*; IEEE: Piscataway, NJ, USA, 2018; pp. 72–80.
35. Meidan, Y.; Bohadana, M.; Mathov, Y.; Mirsky, Y.; Shabtai, A.; Breitenbacher, D.; Elovici, Y. N-BaIoT—Network-Based Detection of IoT Botnet Attacks Using Deep Autoencoders. *IEEE Pervasive Comput.* **2018**, *17*, 12–22.