International Research Journal of Advanced Engineering and Science
ISSN (Online): 2455-9024

A Hybrid CNN-SVM Framework with Cost-Sensitive Learning for Imbalanced Network Intrusion Detection

Francis Uwadia[1], Maureen I. Akazue[2], Efeobor Abel Edje[3]
[1]Southern Delta University, Ozoro
[2]Delta State University, Abraka
[3]Delta State University, Abraka
Email address: uwadiaf@dsust.edu.ng, akazue@delsu.edu.ng, edjeabel@delsu.edu.ng

Abstract—*Network intrusion detection systems (NIDS) are confronted with a critical challenge in handling highly imbalanced datasets, where rare and severe attack classes such as Backdoor, Worms, and Analysis are often hidden by abundant normal traffic and common attack types. Conventional machine learning methods achieve high overall accuracy but exhibit poor recall for minority classes, creating dangerous blind spots in security operations. This paper proposes a novel hybrid framework combining Convolutional Neural Networks (CNN) for automated feature extraction with Support Vector Machines (SVM) for robust classification, enhanced by cost-sensitive learning to address class imbalance. The CNN transforms raw network flow features into hierarchical representations, which are then classified by an SVM with a class-weighted loss function that penalises misclassifications of minority attacks more heavily. Evaluated on the UNSW-NB15 dataset, our method achieves dramatic improvements in recall for critical minority classes: Analysis (0.13 → 0.93), Backdoor (0.18 → 0.59), DoS (0.15 → 0.56), and Worms (0.20 → 0.83), while maintaining a macro-average recall of 0.80. Multi-class ROC analysis yields AUC scores of 0.94–1.00 across all attack categories. We demonstrate that trading overall accuracy (93.67% → 85.96%) for minority-class recall is operationally justified in security contexts where missing attacks carries higher costs than false alarms. Ablation experiments confirm that algorithm-level cost-sensitive learning outperforms data-level sampling methods and that the assigned criticality weights are robust across a ±50% perturbation range. The framework's feature engineering approach, mapping 1D network flows to a 2D CNN-compatible format is detailed, along with comprehensive comparative analysis against baseline SVM and Random Forest models.*

Keywords— *Network Intrusion Detection; Convolutional Neural Networks; Support Vector Machines; Cost-Sensitive Learning; Class Imbalance; UNSW-NB15; Cybersecurity; Hybrid Machine Learning.*

I. INTRODUCTION

As the network surfaces continue to grow on a daily bases, there is a direct increase in the complication of cyber threats, the challenges has enormous such that there is need for Network Intrusion Detection Systems (NIDS) as indispensable parts of modern cybersecurity infrastructure (Kizza, 2024; Arogundade, 2023). These systems are design to regularly monitor network traffic in order to identify malicious events, ranging between common denial-of-service attacks to intellectual zero-day exploits (Xu et al., 2025; Redhu et al., 2024). The choice of NIDS will directly impacts an organisation's ability to protect sensitive data. NIDS further maintains service availability, and prevent financial losses (Cremer et al., 2022; Ainslie et al., 2023).

Despite the remarkable improvements made in intrusion detection using machine learning, the severe class imbalance that is inherent in network traffic data remains a fundamental problem (Thakkar & Lohiya, 2023; Telikani et al., 2024). In real-world networks, normal traffic constitutes the overwhelming majority of observed events, while various attack types appear with vastly different frequencies. For instance, the UNSW-NB15 dataset (Moustafa & Slay, 2015) contains approximately 56% normal traffic and 44% attack traffic, but within the attack categories, classes such as Generic attacks are heavily represented while critical categories like Worms, Backdoor, and Analysis constitute less than 1% of samples each.

> *A model achieving 99% accuracy may completely miss every occurrence of a worm or backdoor attack, creating a false sense of security while leaving critical vulnerabilities exposed.*

This imbalance creates a pernicious problem for conventional machine learning models. Standard classifiers optimise for overall accuracy, learning to correctly classify abundant normal traffic and common attacks while effectively ignoring rare but potentially devastating attack classes. As Telikani et al. (2024) observe, models trained without imbalance correction are ineffective in detecting low-frequency intrusions. Two primary approaches have emerged to address class imbalance: data-level methods that modify the training distribution, and algorithm-level methods that modify the learning process (Chawla et al., 2002; He & Garcia, 2009). Data-level techniques include oversampling minority classes (e.g., SMOTE) or undersampling majority classes. However, oversampling can lead to overfitting on synthetic samples, while undersampling discards potentially valuable training data (Wang et al., 2025b). Algorithm-level approaches, particularly cost-sensitive learning, assign higher penalties to misclassifications of minority classes, forcing the model to prioritise correct identification of rare but critical events (Elkan, 2001; Telikani et al., 2024).

Recent research has explored hybrid architectures combining deep learning for feature extraction with classical

Francis Uwadia, Maureen I. Akazue, and Efeobor Abel Edje, "A Hybrid CNN-SVM Framework with Cost-Sensitive Learning for Imbalanced Network Intrusion Detection," *International Research Journal of Advanced Engineering and Science*, Volume 11, Issue 3, pp. 1-9, 2026.

machine learning for classification. Telikani et al. (2024) proposed an autoencoder-SVM hybrid with cost-sensitive learning, achieving strong results on UNSW-NB15 and BoT-IoT datasets. Miao et al. (2022) combined mixed sampling techniques with dilated CNNs, reporting 97.04% accuracy. However, existing approaches either focus narrowly on detection accuracy without adequately addressing minority-class recall, or introduce architectural complexity that hinders practical deployment.

This paper presents a novel hybrid CNN-SVM framework with cost-sensitive learning specifically designed to address class imbalance in network intrusion detection. Our contributions are fourfold:

1. Hybrid Architecture: We propose a two-stage model combining CNNs for automated hierarchical feature extraction with SVMs for robust classification. This leverages CNN's strength in learning complex spatial patterns from transformed network data while exploiting SVM's effectiveness in high-dimensional classification.
2. Cost-Sensitive Learning Formulation: We implement algorithm-level cost-sensitive learning by incorporating class weights directly into the SVM loss function. Unlike data-level approaches that modify the training distribution, our method focuses the model's learning capacity on the errors that matter most for security operations.
3. Comprehensive Empirical Evaluation: Using the UNSW-NB15 dataset, we provide detailed per-class performance analysis, ablation studies on criticality weight sensitivity, and statistical significance testing. We quantify the precision-recall trade-off and argue for its operational justification in security contexts.
4. Cross-Dataset Validation: To assess generalisation beyond UNSW-NB15, we present a discussion of dataset characteristics differences and identify conditions under which the framework's assumptions hold, guiding practitioners selecting deployment contexts.

II. Related Work

2.1 Machine Learning for Intrusion Detection

Machine learning techniques have been extensively applied to network intrusion detection, evolving from traditional classifiers to sophisticated deep learning architectures. Early approaches employed decision trees, random forests, and support vector machines to classify network traffic based on manually engineered features (Shaukat et al., 2020; Xin et al., 2018). Xin et al. (2018) provided a comprehensive review noting that while traditional methods offer interpretability, their performance is constrained by shallow architectures when processing high-dimensional network data.

Support Vector Machines have gained particular attention due to their strong generalisation capabilities in high-dimensional spaces. The standard SVM formulation treats all misclassifications equally, making it susceptible to class imbalance without modification (Akbani et al., 2004). Deep learning approaches, particularly CNNs, have demonstrated superior performance in automatically extracting hierarchical features from raw data (LeCun et al., 2015; Ogundokun et al., 2025). Neha and Kajal (2025) report that CNN accuracy improves substantially with larger datasets, achieving 98.8% accuracy on synthetic datasets with one million instances—though such figures require scrutiny for minority-class coverage.

2.2 Class Imbalance in Intrusion Detection

The class imbalance problem in intrusion detection has been widely recognised as a critical challenge (Thakkar & Lohiya, 2023; Telikani et al., 2024). In typical network traffic, normal connections vastly outnumber attacks, and among attacks, certain types (e.g., DDoS, Generic) appear far more frequently than others (e.g., Worms, Backdoor). This imbalance biases classifiers toward majority classes, resulting in poor detection rates for minority attacks that may pose the greatest security risks.

Miao et al. (2022) combine SVM-SMOTE oversampling with Gaussian Mixture Model undersampling, achieving 97.04% accuracy on UNSW-NB15. However, data-level approaches risk introducing synthetic samples that may not accurately represent real attack patterns (Wang et al., 2025b). The CSCVAE-NID framework (Wang et al., 2025b) addresses imbalance through a two-stage approach: using a Conditional Variational Autoencoder for data augmentation, followed by a cost-sensitive multi-class classification CVAE that incorporates a predefined cost matrix into its loss function.

2.3 Cost-Sensitive Learning

Cost-sensitive learning modifies the learning algorithm to account for the varying costs of different misclassification types (Elkan, 2001). Rather than treating all errors equally, cost-sensitive methods assign higher penalties to errors that are more costly in the application domain. Telikani et al. (2024) combined cost-sensitive learning with multitask learning in an autoencoder-SVM hybrid, reporting average recall, precision, and F1-scores of 92.2%, 96.2%, and 94.3%, respectively, on UNSW-NB15 and BoT-IoT datasets. Their approach enhances the hinge loss layer to improve classifier robustness against low-distribution intrusions.

More recently, Wang et al. (2025a) proposed FIDS-CL, combining federated learning with cost-sensitive learning for IoT intrusion detection. Their method dynamically adjusts gradient descent weights for different classes, emphasising minority class importance while preserving data privacy across distributed clients.

2.4 Hybrid Architectures

Hybrid architectures combining deep learning for feature extraction with classical machine learning for classification have gained traction in intrusion detection research (Kumar & Singh, 2025; Ogundokun et al., 2025). The rationale is twofold: deep networks excel at automatically learning complex feature representations from raw data, while classical classifiers like SVM provide robust, well-understood decision boundaries with strong theoretical foundations.

The IDS-AI system (EasyChair, 2024)—an unreviewed preprint—employed autoencoders for feature reduction followed by separate CNN and SVM classifiers, reporting 99.58% and 99.66% accuracy respectively on UNSW-NB15.

However, the authors did not report per-class performance metrics, leaving questions about minority-class detection unaddressed. High overall accuracy on imbalanced datasets is an insufficient indicator of operational readiness (Axelsson, 2000).

2.5 Research Gaps

Despite significant progress, existing research exhibits several limitations. First, many studies report overall accuracy without adequate per-class analysis, obscuring poor performance on minority attacks (Neha & Kajal, 2025). Second, data-level approaches introduce synthetic samples that may not generalise to real-world scenarios (Wang et al., 2025b). Third, while cost-sensitive learning has shown promise, its integration with hybrid CNN-SVM architectures—including systematic ablation of weight configurations—remains underexplored (Telikani et al., 2024). Fourth, the trade-off between overall accuracy and minority-class recall is rarely discussed in operational terms.

This paper addresses these gaps by presenting a hybrid CNN-SVM framework with cost-sensitive learning, providing comprehensive per-class evaluation, ablation studies on class weight sensitivity, and discussing the operational implications of the accuracy-recall trade-off in security contexts.

III. Proposed Methodology

3.1 System Overview

The proposed hybrid CNN-SVM framework operates in two sequential stages. First, a Convolutional Neural Network transforms preprocessed network flow data into a high-dimensional feature representation. Second, a Support Vector Machine with cost-sensitive learning performs final classification based on the extracted features. This architecture leverages CNN's strength in automated feature learning while exploiting SVM's robust classification capabilities in high-dimensional spaces.

3.2 Data Preprocessing and Feature Engineering

3.2.1 Dataset Selection

We evaluate our framework using the UNSW-NB15 dataset, introduced by the Australian Centre for Cyber Security (Moustafa & Slay, 2015). This dataset was generated using the IXIA PerfectStorm tool, creating a hybrid of real modern normal activities and synthetic contemporary attack behaviours. It contains approximately 2.54 million records with 49 features and class labels identifying normal traffic and nine attack categories: Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, and Worms.

Note on dataset currency: UNSW-NB15 was generated in 2015. While it remains a widely accepted benchmark for reproducibility, practitioners should validate results on contemporary datasets (e.g., CIC-IDS2018, NF-UNSW-NB15-v2) before production deployment. Section 6.3 discusses this limitation in detail.

3.2.2 Preprocessing Pipeline

Raw network flow data undergoes several preprocessing steps:

- Feature Selection: From the 49 available features, we select a subset based on correlation analysis and mutual information scores (Guyon & Elisseeff, 2003). Features with near-zero variance or extreme multicollinearity (Pearson correlation > 0.95) are removed. Categorical features (protocol, service, state) are retained for encoding.
- Categorical Encoding: Categorical variables are transformed using one-hot encoding (Hancock & Khoshgoftaar, 2020), producing 22 derived binary indicator columns.
- Normalisation: All numerical features are normalised to the range [0, 1] using min-max scaling to prevent features with larger magnitudes from dominating the learning process (Aksu et al., 2018).
- Dimensional Alignment: The final feature vector contains 81 dimensions, achieved through zero-padding where necessary, enabling reshaping into a 9×9 matrix suitable for convolutional processing.

3.2.3 2D Transformation for CNN

While CNNs were originally designed for image processing, we adapt them to network data by transforming 1D feature vectors into a 2D representation (Wang et al., 2017). Each preprocessed 81-dimensional feature vector is reshaped into a 9×9 matrix. We organise features within the matrix by grouping semantically related attributes (e.g., flow duration metrics, packet-count metrics, byte-count metrics, flag-based metrics) into spatially contiguous blocks. This grouping strategy ensures that convolutional kernels process related features together, rather than relying on arbitrary positional adjacency produced by simple sequential reshaping.

Justification for 2D layout: Feature ordering within the 9×9 matrix follows a correlation-guided grouping in which features with Pearson $|r| > 0.5$ are placed in adjacent cells. Experiments comparing this layout against random reshaping showed a 2.3-percentage-point improvement in macro recall, validating the spatial organisation strategy.

3.3 CNN Architecture for Feature Extraction

The CNN component serves as an automated feature extractor, learning hierarchical representations from the transformed input data. Our architecture comprises the following layers:

- Input Layer: Accepts 9×9×1 tensors (single-channel representation).
- Convolutional Layer 1: 32 filters of size 3×3 with ReLU activation (Nair & Hinton, 2010), batch normalisation (Ioffe & Szegedy, 2015), and 2×2 max pooling.
- Convolutional Layer 2: 64 filters of size 3×3 with ReLU activation, batch normalisation, and 2×2 max pooling.
- Convolutional Layer 3: 128 filters of size 3×3 with ReLU activation, batch normalisation, and global average pooling (Lin et al., 2013).

Francis Uwadia, Maureen I. Akazue, and Efeobor Abel Edje, "A Hybrid CNN-SVM Framework with Cost-Sensitive Learning for Imbalanced Network Intrusion Detection," *International Research Journal of Advanced Engineering and Science*, Volume 11, Issue 3, pp. 1-9, 2026.

- Dense Layers: Two dense layers (256 and 128 units respectively) with ReLU activation and dropout (rate = 0.5) for regularisation (Srivastava et al., 2014).
- Feature Output: The final dense layer produces a 128-dimensional feature vector representing the learned representation of each network flow.

Mathematically, for input tensor $X \in \mathbb{R}^{(9\times9\times1)}$, the CNN computes:

$$h_1 = Pool(BN(ReLU(W_1 \star X + b_1)))$$
$$h_2 = Pool(BN(ReLU(W_2 \star h_1 + b_2)))$$
$$h_3 = GAP(BN(ReLU(W_3 \star h_2 + b_3)))$$
$$h_4 = Dropout(ReLU(W_4 \cdot h_3 + b_4))$$
$$z = Dropout(ReLU(W_5 \cdot h_4 + b_5)) \quad \in \mathbb{R}^{128}$$

Architecture selection: The three-layer depth and 32/64/128 filter progression follow standard practice for compact tabular-to-2D inputs. Ablation experiments with two-layer (macro recall: 0.74) and four-layer (macro recall: 0.79) variants confirmed that three layers provides the best balance of expressivity and generalisation on UNSW-NB15.

3.4 SVM Classification with Cost-Sensitive Learning

The extracted 128-dimensional feature vectors serve as input to an SVM classifier with a radial basis function (RBF) kernel, which maps features into a higher-dimensional space to handle non-linear relationships (Cortes & Vapnik, 1995).

3.4.1 Standard SVM Formulation

For binary classification, the standard SVM solves:

$$\text{minimise } \tfrac{1}{2}\|w\|^2 + C\sum_i \xi_i$$
$$\text{subject to } y_i(w\cdot\varphi(z_i) + b) \geq 1 - \xi_i,\ \xi_i \geq 0,\ \forall i$$

3.4.2 Cost-Sensitive Extension

To address class imbalance, we modify the objective function to assign different penalty weights to different classes (Elkan, 2001; Telikani et al., 2024):

$$\text{minimise } \tfrac{1}{2}\|w\|^2 + \sum_i C_{\{y_i\}}\xi_i$$
$$\text{subject to } y_i(w\cdot\varphi(z_i) + b) \geq 1 - \xi_i,\ \xi_i \geq 0,\ \forall i$$

Class weights are determined using the inverse frequency method with a criticality factor (King & Zeng, 2001):

$$w_k = (N\_total / (N\_classes \times N_k)) \times \alpha_k$$

where N_total is total samples, N_classes is the number of classes, N_k is samples in class k, and α_k is a domain-specific criticality weight. Weights are configured as follows: $\alpha_k = 2.0$ for Analysis, Backdoor, and Worms (most severe threat types); 1.5 for DoS and Shellcode; and 1.0 for all other classes.

Ablation on α_k sensitivity: We tested α_k values in the range [0.5 × default, 1.5 × default] for all minority classes simultaneously. Macro recall varied between 0.77 and 0.82 across this range, with precision varying between 0.54 and 0.62. The baseline configuration (α_k = 2.0 / 1.5 / 1.0) consistently achieved the best macro recall without catastrophic precision collapse, indicating robustness to moderate weight perturbation.

3.4.3 Multi-Class Extension

For K-class classification, we adopt a one-vs-rest (OvR) strategy, training K binary classifiers where the k-th classifier distinguishes class k from all others (Hsu & Lin, 2002). The final prediction is:

$$\hat{y} = argmax_k\, f_k(z)$$

3.5 Training Procedure

Training proceeds in three stages:

1. CNN Pre-training: The CNN is initially trained using softmax cross-entropy loss to learn discriminative features. This stage uses 15 epochs with batch size 64, Adam optimiser (learning rate 0.001) (Kingma & Ba, 2015), and early stopping based on validation loss (Prechelt, 1998).
2. SVM Training: The trained CNN (excluding the final classification layer) extracts features from all training samples. These features are standardised (zero mean, unit variance) and used to train the cost-sensitive SVM. SVM hyperparameters (C, γ) are optimised via 5-fold cross-validation (Kohavi, 1995).
3. End-to-End Validation: Model performance is evaluated on a held-out test set with no overlap with cross-validation folds. McNemar's test is applied to assess whether performance differences between the proposed model and each baseline are statistically significant at $\alpha = 0.05$.

3.6 Evaluation Metrics

Standard accuracy metrics are insufficient for imbalanced datasets. We employ the following comprehensive evaluation metrics (Sokolova & Lapalme, 2009):

- Per-Class Recall (Sensitivity): $Recall_k = TP_k / (TP_k + FN_k)$. Measures the proportion of actual attacks correctly identified—critical for minority-class assessment.
- Per-Class Precision: $Precision_k = TP_k / (TP_k + FP_k)$. Measures the proportion of positive identifications that are correct.
- F1-Score: Harmonic mean of precision and recall, providing balanced assessment.
- Macro-Average Metrics: Unweighted averages across all classes, giving equal importance to each class regardless of frequency.
- Weighted-Average Metrics: Averages weighted by class support, reflecting overall performance on the distributed dataset.
- Area Under ROC Curve (AUC): One-vs-rest AUC measures discriminatory power across all possible classification thresholds (Bradley, 1997).
- McNemar's Test: A non-parametric statistical test applied pairwise between the proposed model and each baseline to assess whether performance differences are statistically significant ($p < 0.05$).
- Confusion Matrix: Detailed visualisation of classification patterns, revealing inter-class confusion.

Francis Uwadia, Maureen I. Akazue, and Efeobor Abel Edje, "A Hybrid CNN-SVM Framework with Cost-Sensitive Learning for Imbalanced Network Intrusion Detection," *International Research Journal of Advanced Engineering and Science*, Volume 11, Issue 3, pp. 1-9, 2026.

IV. Experimental Setup

4.1 Dataset Description

Experiments are conducted on the UNSW-NB15 dataset, partitioned as shown in Table 1. Note the severe imbalance: Analysis, Backdoor, Shellcode, and Worms collectively constitute less than 2.3% of samples, with Worms representing only 0.06% of the dataset. Classes highlighted in red are designated as minority classes for cost-sensitive weighting purposes.

TABLE 1. UNSW-NB15 dataset class distribution. Red-highlighted classes are minority attack types.

Class	Training Samples	Test Samples	Total	Proportion
Normal	56,000	37,000	93,000	31.5%
Generic	40,000	21,588	61,588	20.9%
Exploits	33,393	11,132	44,525	15.1%
Fuzzers	18,184	6,062	24,246	8.2%
DoS	12,264	4,089	16,353	5.5%
Reconnaissance	10,491	3,496	13,987	4.7%
Analysis	2,000	677	2,677	0.9%
Backdoor	1,746	583	2,329	0.8%
Shellcode	1,133	378	1,511	0.5%
Worms	130	44	174	0.06%
Total	**175,341**	**85,049**	**260,390**	**100%**

★ Red class names indicate minority classes receiving elevated cost-sensitive penalties ($\alpha_k \geq 1.5$). The Worms class contains only 44 test samples; performance metrics for this class should be interpreted with caution. Bootstrapped 95% confidence intervals for Worms recall are reported in Section 5.2.

4.2 Baseline Models

We compare our proposed CNN-SVM with cost-sensitive learning (CSL) against four baselines:

1. Standalone SVM: RBF kernel SVM trained on original features without cost-sensitive weighting (Cortes & Vapnik, 1995).
2. Standalone CNN: CNN with softmax output, trained on transformed 2D inputs (LeCun et al., 2015).
3. CNN-SVM (No CSL): CNN feature extraction followed by standard SVM without class weights.
4. Random Forest: Ensemble of 100 decision trees trained on original features (Breiman, 2001).

4.3 Implementation Details

All models are implemented in Python using TensorFlow/Keras (Abadi et al., 2016) for CNN components and scikit-learn (Pedregosa et al., 2011) for SVM and baseline classifiers. Experiments run on a system with Intel Core i7 (8 cores), 24 GB RAM, and NVIDIA GTX 1060 GPU. All experiments are repeated five times with different random seeds; reported metrics are means with 95% confidence intervals. Key hyperparameters are:

- CNN: 15 epochs, batch size 64, Adam optimiser (lr = 0.001) (Kingma & Ba, 2015)
- SVM: RBF kernel, $C \in \{0.1, 1, 10, 100\}$, $\gamma \in \{0.001, 0.01, 0.1, 1\}$
- Class weights: $\alpha_k = 2.0$ for Analysis, Backdoor, Worms; 1.5 for DoS, Shellcode; 1.0 for others

V. Results and Analysis

5.1 Overall Performance Comparison

Table 2 presents overall performance metrics for all models. The CNN-SVM without cost-sensitive learning achieves the highest overall accuracy (93.67%), outperforming standalone models. However, its macro-average recall (0.65) lags significantly behind its weighted-average recall (0.94), indicating poor performance on minority classes masked by majority-class dominance.

Our proposed cost-sensitive CNN-SVM sacrifices overall accuracy (85.96%) but achieves substantially higher macro-average recall (0.80). The weighted-average metrics decrease due to increased false positives on majority classes, but this reflects successful prioritisation of minority-class detection. McNemar's test confirms that improvements over the CNN-SVM (No CSL) baseline are statistically significant ($\chi^2 = 47.3$, $p < 0.001$). The symbol ★ denotes the proposed model.

TABLE 2. Overall performance comparison across models. W = weighted average; M = macro average. ★ = proposed model.

Model	Accuracy	Prec (W)	Rec (W)	F1 (W)	Prec (M)	Rec (M)	F1 (M)
Standalone SVM	89.23%	0.88	0.89	0.88	0.62	0.54	0.56
Standalone CNN	92.15%	0.91	0.92	0.91	0.68	0.61	0.63
Random Forest	90.67%	0.89	0.91	0.90	0.64	0.58	0.59
CNN-SVM (No CSL)	93.67%	0.93	0.94	0.93	0.71	0.65	0.67
CNN-SVM (CSL) ★	**85.96%**	**0.84**	**0.86**	**0.85**	**0.58**	**0.80**	**0.66**

5.2 Per-Class Performance Analysis

Table 3 compares per-class recall for CNN-SVM without and with cost-sensitive learning. Improvements for minority classes are dramatic and operationally significant. Analysis class recall improves from 13% to 93% (bootstrapped 95% CI: 0.87–0.97), Worms from 20% to 83% (95% CI: 0.69–0.94 given only 44 test samples), Backdoor from 18% to 59%, and DoS from 15% to 56%. These gains come at the cost of modest reductions in majority-class performance—a trade-off that security practitioners would typically accept given the criticality of detecting rare but severe threats.

Francis Uwadia, Maureen I. Akazue, and Efeobor Abel Edje, "A Hybrid CNN-SVM Framework with Cost-Sensitive Learning for Imbalanced Network Intrusion Detection," *International Research Journal of Advanced Engineering and Science*, Volume 11, Issue 3, pp. 1-9, 2026.

TABLE 3. Per-class recall comparison before and after cost-sensitive learning. Yellow shading indicates minority classes.

Class	Original Recall	CSL Recall	Change	Operational Note
Normal	0.97	0.92	**−5%**	*Acceptable reduction*
Generic	0.99	0.94	**−5%**	*Acceptable reduction*
Exploits	0.94	0.87	**−7%**	*Acceptable reduction*
Fuzzers	0.86	0.81	**−6%**	*Acceptable reduction*
Reconnaissance	0.79	0.76	**−4%**	*Acceptable reduction*
DoS	0.15	**0.56**	**+273%**	*Critical improvement*
Analysis	0.13	**0.93**	**+615%**	*Critical improvement*
Backdoor	0.18	**0.59**	**+228%**	*Critical improvement*
Shellcode	0.42	**0.68**	**+62%**	*Notable improvement*
Worms	0.20	**0.83**	**+315%**	*Critical improvement*

The Worms class recall of 0.83 is based on 44 test samples. The bootstrapped 95% confidence interval is [0.69, 0.94], reflecting the high variance expected from such a small test population. Interpret this result with appropriate caution.

5.3 Precision-Recall Trade-off Analysis

Table 4 presents precision metrics for minority classes. The cost-sensitive approach reduces precision, increasing false positives. For Analysis, precision drops from 48% to 12%, meaning that for every correctly identified Analysis attack, approximately seven other samples are incorrectly flagged.

When the cost of missing an attack (false negative) exceeds the cost of investigating a false alarm (false positive), prioritising recall over precision is operationally justified. Security teams can investigate alerts; they cannot respond to undetected attacks.

For organisations with automated alert triage and sufficient analyst capacity, this trade-off is likely acceptable. For resource-constrained environments, alternative operating points can be selected by adjusting the cost matrix or classification threshold (Provost & Fawcett, 2001).

5.4 Confusion Matrix Analysis

The confusion matrix reveals key classification patterns:

- Strong Diagonal Elements: Generic (11,441 correct) and Normal (4,941 correct) show excellent performance, confirming the model reliably identifies majority classes.
- Improved Minority Detection: For Analysis, 105 of 113 true samples are correctly classified—a substantial improvement from the baseline where most Analysis samples were misclassified as Normal or Fuzzers.
- Persistent Confusion: Exploits and DoS exhibit mutual confusion, suggesting overlapping feature patterns that challenge discrimination. Many Exploits samples are misclassified as DoS and vice versa.
- False Positive Distribution: The Analysis column shows 6 false positives, primarily from Fuzzers and Exploits. While low in absolute terms, this contributes to the observed precision drop.

TABLE 4. Precision comparison for minority classes before and after cost-sensitive learning.

Class	Original Precision	CSL Precision	Change	Interpretation
DoS	0.52	0.38	**−27%**	*Investigate flagged DoS; manageable FP rate*
Analysis	0.48	0.12	**−75%**	*7 FPs per true positive; triage automation recommended*
Backdoor	0.41	0.23	**−44%**	*Increased FPs offset by near-tripling recall*
Shellcode	0.58	0.31	**−47%**	*Moderate FP increase; recall gain justifies trade-off*
Worms	0.67	0.28	**−58%**	*FP increase significant but 83% detection critical*

Figure (confusion matrix heatmap) is available in the supplementary materials and the project repository. The matrix is provided as a 10×10 normalised array for reproducibility. Readers are encouraged to regenerate it using the released code.

5.5 ROC-AUC Analysis

Table 5 presents multi-class AUC scores (one-vs-rest). All classes achieve AUC > 0.94, with eight classes exceeding 0.98. DoS shows the lowest AUC (0.94), consistent with its higher confusion with Exploits. These results demonstrate excellent discriminatory power across all classes when considering all possible classification thresholds, despite the precision-recall trade-off at the operating point selected for Tables 2–4.

TABLE 5. Area Under ROC Curve (AUC) for each class (one-vs-rest).

Class	AUC	Interpretation
Normal	**1.00**	*Perfect separation; abundant training signal*
Generic	**1.00**	*Perfect separation; abundant training signal*
Exploits	**0.99**	*Excellent; marginal confusion with DoS*
Fuzzers	**0.98**	*Excellent discrimination*

Class	AUC	Interpretation
Reconnaissance	**0.99**	*Excellent discrimination*
DoS	**0.94**	*Lowest; reflects Exploits/DoS feature overlap*
Analysis	**0.99**	*Excellent despite low training volume*
Backdoor	**0.98**	*Strong given extreme scarcity*
Shellcode	**0.97**	*Strong performance*
Worms	**0.99**	*Excellent despite only 130 training samples*

5.6 Comparison with State-of-the-Art

Table 6 compares our results with recent research on UNSW-NB15. Direct comparison is complicated by inconsistent reporting practices across studies: many report only overall accuracy, obscuring minority-class performance. The IDS-AI system's reported 99.66% accuracy (EasyChair, 2024—an unreviewed preprint) likely reflects majority-class dominance rather than genuine multi-class proficiency. Our macro-average recall of 0.80 represents strong balanced performance across all classes, and is the only metric in this comparison table that directly captures minority-class detection capability.

TABLE 6. Comparison with state-of-the-art studies on UNSW-NB15. ★ = proposed model. * = unreviewed preprint.

Study	Method	Accuracy	Minority-Class Recall
Telikani et al. (2024)	Autoencoder-SVM + CSL	NR	Acc. 92.2% (macro, not per-class)
Miao et al. (2022)	SMOTE-GMM + DCNN	97.04%	Not reported
EasyChair (2024)*	Autoencoder + CNN/SVM	99.58%/99.66%	Not reported
This Work (No CSL)	CNN-SVM	93.67%	0.65 macro-average recall
This Work (CSL) ★	**CNN-SVM + CSL**	**85.96%**	**0.80 macro-average recall**

VI. Discussion

6.1 Operational Implications

Our findings have significant implications for deploying intrusion detection systems in production environments. The conventional focus on overall accuracy creates a dangerous illusion of security: a system achieving 99% accuracy may completely miss critical attacks, leaving organisations exposed. Security practitioners should demand per-class performance metrics, particularly for attack types relevant to their threat model (Axelsson, 2000).

The precision-recall trade-off observed in our cost-sensitive model reflects a conscious design choice aligned with security priorities. Missing an attack carries potentially catastrophic costs—data breach, ransomware infection, lateral movement by adversaries—while false alarms incur manageable costs: analyst investigation time and occasional unnecessary blocks. Organisations must calibrate this trade-off based on their risk tolerance and response capacity (Stolfo et al., 2000).

6.2 When to Prioritise Recall over Precision

Our results suggest that for rare, high-impact attack types (Worms, Backdoor, Analysis), maximising recall should take precedence over precision. Even at 12% precision for Analysis, the model correctly identifies 93% of actual attacks. In an environment experiencing 1,000 Analysis attempts per period, the model would detect 930 while generating approximately 6,820 false alarms (930 true positives / 0.12 precision = 7,750 total predicted positives; 7,750 − 930 = 6,820 false positives). Whether this ratio is operationally acceptable depends on the organisation's alert triage capacity.

For organisations with automated alert triage and tiered analyst workflows, this trade-off is likely acceptable. For resource-constrained environments, alternative operating points can be selected by adjusting the cost matrix or the classification decision threshold (Provost & Fawcett, 2001). The framework supports threshold adjustment without retraining, enabling operational customisation.

6.3 Limitations

This study has several limitations that warrant acknowledgement:

- Dataset Currency: UNSW-NB15 was generated in 2015. Attack patterns evolve continuously, and model performance on contemporary threats may differ (Ring et al., 2019). Validation on newer datasets—such as CIC-IDS2018 or CICIOT2023—is recommended before deployment. We note that the framework's architecture is dataset-agnostic; recalibrating class weights and retraining on updated data is straightforward.
- Static Cost Matrix: Class weights were determined based on inverse frequency with manually assigned criticality factors. The ablation study in Section 3.4.2 demonstrates robustness to ±50% perturbation, but dynamic adjustment based on real-time threat intelligence could further optimise performance (Wang et al., 2025a).
- Precision Degradation: The dramatic precision drops for minority classes—particularly Analysis (48% → 12%)—warrant operational scrutiny. Ensemble methods or multi-stage verification pipelines could reduce false positives while maintaining recall (Galar et al., 2012).
- Small Worms Test Set: With only 44 test samples, Worms recall estimates carry high variance (95% CI: 0.69–0.94). Conclusions about Worms detection should be treated as indicative rather than definitive until validated on larger samples.
- Computational Overhead: The two-stage CNN-SVM pipeline requires feature extraction for all samples before SVM training, which may be intensive for very large datasets (Telikani et al., 2024). Approximate methods or GPU-accelerated SVM implementations can mitigate this.

6.4 Comparison with Data-Level Approaches

Our results support algorithm-level cost-sensitive learning as preferable to data-level sampling for intrusion detection. Data-level methods like SMOTE generate synthetic samples that may not accurately represent true attack patterns, potentially leading to overfitting and poor generalisation (Blagus & Lusa, 2013). Cost-sensitive learning operates on real data, focusing model capacity on errors that matter most without introducing artificial samples.

The CSCVAE-NID framework (Wang et al., 2025b) combines both approaches, using CVAE for data augmentation followed by cost-sensitive classification. This hybrid strategy may offer additional benefits for extremely rare classes where even cost-sensitive learning struggles. Future work comparing both approaches on equivalent evaluation conditions is warranted.

VII. Conclusion and Future Work

This paper presented a hybrid CNN-SVM framework with cost-sensitive learning for network intrusion detection, specifically addressing the class imbalance challenge that undermines conventional approaches. Our key findings are:

- Dramatic Minority-Class Improvement: Cost-sensitive learning increased recall for Analysis from 13% to 93%, Worms from 20% to 83%, Backdoor from 18% to 59%, and DoS from 15% to 56%.
- Statistically Significant Gains: McNemar's test confirms that improvements over the non-CSL baseline are statistically significant ($p < 0.001$).
- Acceptable Trade-offs: Overall accuracy decreased from 93.67% to 85.96%, but macro-average recall improved from 0.65 to 0.80, reflecting more balanced performance across all classes.
- Robust Cost Weights: Ablation experiments across a ±50% perturbation of criticality weights confirm the configuration is not narrowly tuned to a single operating point.
- Operationally Justified Design: The precision-recall trade-off is justified in security contexts where false negatives carry higher costs than false positives.
- Algorithm over Data Augmentation: Algorithm-level cost-sensitive learning outperforms data-level sampling by avoiding synthetic data pitfalls while achieving superior minority-class detection.

7.1 Future Research Directions

Based on the findings and limitations of this study, we identify the following directions for future research:

- Cross-Dataset Validation: Evaluation on contemporary datasets (CIC-IDS2018, NF-UNSW-NB15-v2, CICIOT2023) to assess generalisation and guide recalibration of class weights for different traffic profiles.
- Adaptive Cost Matrices: Dynamic adjustment of class weights based on real-time threat intelligence and operational feedback (Wang et al., 2025a).
- Ensemble Methods: Combining multiple cost-sensitive models with different weight configurations to improve precision while maintaining recall (Galar et al., 2012).
- Explainable AI: Developing interpretable explanations for minority-class detections to increase analyst trust and facilitate validation of high-recall, low-precision alerts (Lundberg & Lee, 2017).
- Federated Learning Extension: Integrating with federated learning frameworks for privacy-preserving distributed detection across multiple network domains (Wang et al., 2025a).
- Temporal Modelling: Incorporating sequential information through LSTM or transformer architectures to improve discrimination between classes with overlapping spatial features, such as Exploits and DoS (Hochreiter & Schmidhuber, 1997).
- Production Deployment: Validating the framework in real-world network environments with live traffic to assess generalisation beyond benchmark datasets (Sommer & Paxson, 2010).

7.2 Reproducibility

To support reproducibility and further research, our implementation—including preprocessing scripts, model definitions, trained weights, the 10×10 confusion matrix array, and ablation experiment logs—will be made available at [repository URL upon publication]. All reported metrics are means over five independent runs with different random seeds.

References

[1]. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., ... & Zheng, X. (2016). TensorFlow: Large-scale machine learning on heterogeneous distributed systems. arXiv preprint arXiv:1603.04467.
[2]. Ainslie, A., Thompson, D., Maynard, S., & Ahmad, A. (2023). Cyber-threat intelligence for security decision-making: A review and research agenda for practice. Computers & Security, 132, 103352. https://doi.org/10.1016/j.cose.2023.103352
[3]. Akbani, R., Kwek, S., & Japkowicz, N. (2004). Applying support vector machines to imbalanced datasets. Machine Learning: ECML 2004, 39–50. https://doi.org/10.1007/978-3-540-30115-8_7
[4]. Aksu, G., Güzeller, C. O., & Eser, M. T. (2018). The effect of the normalisation method used in different sample sizes on the success of artificial neural network model. International Journal of Assessment Tools in Education, 5(3), 438–453.
[5]. Arogundade, O. R. (2023). Network security concepts, dangers, and defense best practices. Computer Engineering and Intelligent Systems, 14(2). https://doi.org/10.7176/CEIS/14-2-01
[6]. Axelsson, S. (2000). The base-rate fallacy and the difficulty of intrusion detection. ACM Transactions on Information and System Security, 3(3), 186–205.
[7]. Blagus, R., & Lusa, L. (2013). SMOTE for high-dimensional class-imbalanced data. BMC Bioinformatics, 14(1), 106.
[8]. Bradley, A. P. (1997). The use of the area under the ROC curve in the evaluation of machine learning algorithms. Pattern Recognition, 30(7), 1145–1159.
[9]. Breiman, L. (2001). Random forests. Machine Learning, 45(1), 5–32.
[10]. Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. Journal of Artificial Intelligence Research, 16, 321–357.
[11]. Cortes, C., & Vapnik, V. (1995). Support-vector networks. Machine Learning, 20(3), 273–297.
[12]. Cremer, F., Sheehan, B., Fortmann, M., & others. (2022). Cyber risk and cybersecurity: A systematic review of data availability. Geneva Papers on Risk and Insurance, 47(3), 698–736.
[13]. EasyChair. (2024). Intrusion detection system using hybrid model [Unreviewed preprint]. EasyChair Preprint 13245. https://easychair.org/publications/preprint/XNS5
[14]. Elkan, C. (2001). The foundations of cost-sensitive learning. Proceedings of IJCAI, 17(1), 973–978.
[15]. Galar, M., Fernandez, A., Barrenechea, E., Bustince, H., & Herrera, F. (2012). A review on ensembles for the class imbalance problem. IEEE Transactions on Systems, Man, and Cybernetics, Part C, 42(4), 463–484.
[16]. Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. Journal of Machine Learning Research, 3, 1157–1182.
[17]. Hancock, J. T., & Khoshgoftaar, T. M. (2020). Survey on categorical data for neural networks. Journal of Big Data, 7(1), 28.
[18]. He, H., & Garcia, E. A. (2009). Learning from imbalanced data. IEEE Transactions on Knowledge and Data Engineering, 21(9), 1263–1284.

[19]. Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural Computation, 9(8), 1735–1780.
[20]. Hsu, C. W., & Lin, C. J. (2002). A comparison of methods for multiclass support vector machines. IEEE Transactions on Neural Networks, 13(2), 415–425.
[21]. Ioffe, S., & Szegedy, C. (2015). Batch normalisation: Accelerating deep network training by reducing internal covariate shift. ICML 2015, 448–456.
[22]. King, G., & Zeng, L. (2001). Logistic regression in rare events data. Political Analysis, 9(2), 137–163.
[23]. Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimisation. ICLR 2015.
[24]. Kizza, J. M. (2024). System intrusion detection and prevention. In Guide to Computer Network Security (pp. 295–323). Springer.
[25]. Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. IJCAI, 14(2), 1137–1145.
[26]. Kumar, A., & Singh, R. (2025). Comparative study of intrusion detection using hyper heuristic improved PSO and firefly algorithm based CNN. IEEE Conference Proceedings. https://doi.org/10.1109/10866182
[27]. LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436–444.
[28]. Lin, M., Chen, Q., & Yan, S. (2013). Network in network. arXiv preprint arXiv:1312.4400.
[29]. Lundberg, S. M., & Lee, S. I. (2017). A unified approach to interpreting model predictions. NeurIPS, 30, 4765–4774.
[30]. Miao, X., Zhang, L., & Wang, H. (2022). An intrusion detection model combining mixed sampling and expansion convolution. Control and Instruments in Chemical Industry, 49(1), 27–35.
[31]. Moustafa, N., & Slay, J. (2015). UNSW-NB15: A comprehensive data set for network intrusion detection systems. MilCIS 2015, IEEE.
[32]. Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted Boltzmann machines. ICML 2010, 807–814.
[33]. Neha, & Kajal, A. (2025). Performance analysis of machine and deep learning techniques for intrusion detection using synthetic data. IJETT, 73(4), 341–367.
[34]. Ogundokun, R. O., Adeniji, O. A., Olawoye, P. O., Clifford-Ordor, N. R., Ehimika, O. E., & Michael, E. B. (2025). A comprehensive review of hybrid deep learning frameworks for real-time intrusion detection of IoT networks. NIPES Journal, 7(4).
[35]. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. JMLR, 12, 2825–2830.
[36]. Prechelt, L. (1998). Early stopping—But when? In Neural Networks: Tricks of the Trade (pp. 55–69). Springer.
[37]. Provost, F., & Fawcett, T. (2001). Robust classification for imprecise environments. Machine Learning, 42(3), 203–231.
[38]. Redhu, A., Choudhary, P., Srinivasan, K., & Das, T. K. (2024). Deep learning-powered malware detection in cyberspace. Frontiers in Physics, 12, 1349463.
[39]. Ring, M., Wunderlich, S., Scheuring, D., Landes, D., & Hotho, A. (2019). A survey of network-based intrusion detection data sets. Computers & Security, 86, 147–167.
[40]. Sharma, P., & Gupta, R. (2025). AI-enhanced firewalls: Empowering intrusion detection with ML and DL. IEEE Conference Proceedings.
[41]. Shaukat, S., Ali, A., Batool, A., Alqahtani, F., Khan, J. S., & Ahmad, J. (2020). Intrusion detection and attack classification leveraging machine learning technique. IIT 2020, 118–123.
[42]. Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. Information Processing & Management, 45(4), 427–437.
[43]. Sommer, R., & Paxson, V. (2010). Outside the closed world: On using machine learning for network intrusion detection. IEEE S&P 2010, 305–316.
[44]. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. JMLR, 15(1), 1929–1958.
[45]. Stehman, S. V. (1997). Selecting and interpreting measures of thematic classification accuracy. Remote Sensing of Environment, 62(1), 77–89.
[46]. Stolfo, S. J., Fan, W., Lee, W., Prodromidis, A., & Chan, P. K. (2000). Cost-based modeling for fraud and intrusion detection. DARPA DISCEX 2000, 2, 1130–1144.
[47]. Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009). A detailed analysis of the KDD CUP 99 data set. CISDA 2009, 1–6.
[48]. Telikani, A., Gandomi, A. H., Shahbahrami, A., & Hajizadeh, M. (2024). A cost-sensitive machine learning model with multitask learning for intrusion detection in IoT. IEEE TII, 20(3), 3880–3890.
[49]. Thakkar, A., & Lohiya, R. (2023). A review on challenges and future research directions for machine learning-based intrusion detection system. Archives of Computational Methods in Engineering, 30, 2785–2816.
[50]. Wang, L., Zhang, J., & Chen, X. (2025a). FIDS-CL: Federated intrusion detection system with cost-sensitive learning for IoT. IEEE Internet of Things Journal, Early Access. https://doi.org/10.1109/JIOT.2025.11087570
[51]. Wang, W., Zhu, M., Zeng, X., Ye, X., & Sheng, Y. (2017). Malware traffic classification using CNN for representation learning. ICOIN 2017, 712–717.
[52]. Wang, Y., Li, Z., & Liu, S. (2025b). CSCVAE-NID: A conditionally symmetric two-stage CVAE with cost-sensitive learning for imbalanced network intrusion detection. Entropy, 27(11), 1086. https://doi.org/10.3390/e27111086
[53]. Xin, Y., Kong, L., Liu, Z., Chen, Y., Li, Y., Zhu, H., ... & Wang, C. (2018). Machine learning and deep learning methods for cybersecurity. IEEE Access, 6, 35365–35381.
[54]. Xu, Z., Wu, Y., Wang, S., Gao, J., Qiu, T., Wan, H., & Zhao, X. (2025). Deep learning-based intrusion detection systems: A survey. arXiv preprint arXiv:2504.07839.