

Machine Learning for Malware Analysis and Detection

Vedant R. Hemane¹, Ajit Shende², Kiran P. Pustode³

¹Student, Dept. of MCA, Karanjekar College of Engineering & Management, Sakoli

²Asst. Professor, Dept. of MCA, Karanjekar College of Engineering & Management, Sakoli

³HOD, Dept. of MCA, Karanjekar College of Engineering & Management, Sakoli

Abstract – Malware remains among the most persistent and damaging cybersecurity threats, with signature-based detection systems proving inadequate against polymorphic variants, zero-day exploits, and obfuscation techniques. This paper presents the design, implementation, and empirical evaluation of a hybrid deep learning framework that integrates Convolutional Neural Networks (CNN) with a self-attention mechanism for multi-class malware family classification using static Import Address Table (IAT) features extracted from Windows PE32 executables. The system employs a structured five-stage data pipeline—loading, preprocessing, statistical feature engineering, variance-based filtering, and SMOTE-based class balancing—applied to a dataset of 387 samples spanning eight malware families: Backdoor, Downloader, Keylogger, Miner, Ransomware, Rouge Software, Trojan, and Worm. The proposed CNN-Attention model is evaluated against three baseline classifiers—Naive Bayes, SVM (RBF), and Decision Tree—using accuracy, macro F1-score, weighted F1-score, confusion matrices, per-class ROC-AUC curves, and five-fold stratified cross-validation. The CNN-Attention model achieved a weighted F1-score of 0.50 and a cross-validation mean accuracy of 0.579, outperforming Naive Bayes and SVM on weighted metrics. The self-attention mechanism enhanced interpretability by identifying the most influential Windows API calls—CertCreateContext, AreFileApisANSI, and AddAtom—that drive classification decisions. Experimental analysis confirms that severe class imbalance and limited dataset size are the primary constraints on minority-class generalisation, providing evidence-based guidance for future dataset expansion and architecture refinement.

Key Words: Malware Detection, CNN-Attention, IAT Feature Extraction, SMOTE, Static Analysis, PE32 Classification, Multi-Class, Interpretability.

1. INTRODUCTION

The rapid proliferation of internet-connected systems has proportionally expanded the attack surface available to cybercriminals, with malware remaining the most pervasive and damaging attack vector across all sectors. Malware has evolved from basic viruses into sophisticated families including ransomware, spyware, cryptominers, and fileless attacks, employing code obfuscation, encryption, and polymorphism to evade conventional defences [1], [2]. Windows Portable Executable (PE32) files constitute the dominant malware distribution mechanism, making them the primary target for both academic research and operational detection systems.

Traditional signature-based detection systems are fundamentally reactive: they can only match threats whose signatures exist in curated databases, rendering them ineffective against novel variants, polymorphic malware, and zero-day attacks where no prior signature exists [3]. Machine learning (ML) addresses this limitation by learning

discriminative patterns from malware feature representations, enabling generalisation to unseen samples beyond the training distribution. Static analysis of Import Address Table (IAT) features—cataloguing which Windows API functions a PE32 file imports—provides a rich, execution-free feature space that encodes structural malware behaviour without incurring sandbox execution overhead [4], [11].

Despite substantial progress in ML-based malware detection, the reviewed literature reveals four persistent gaps: classical models fail to generalise across heterogeneous malware families; feature selection lacks adversarial robustness validation; cross-model comparisons are conducted against narrow baselines; and most studies evaluate only binary classification rather than the more realistic multi-class family identification task [5], [6]. Deep learning architectures combining convolutional feature extraction with attention mechanisms address the first two gaps by learning spatial feature hierarchies and dynamically weighting discriminative API patterns [9], [10].

This paper presents a hybrid CNN-Attention framework for multi-class PE32 malware classification. The key contributions are: (i) a five-stage data processing pipeline with statistical feature engineering and variance-based dimensionality reduction; (ii) SMOTE-based class balancing applied exclusively to the training set to prevent data leakage; (iii) a hybrid CNN + Self-Attention architecture with residual connections, batch normalisation, and layer normalisation; (iv) comprehensive multi-metric evaluation against three baseline classifiers and five-fold cross-validation; and (v) attention-weight-based interpretability analysis identifying the most influential Windows API features driving classification.

2. LITERATURE REVIEW

2.1 Classical Machine Learning Approaches

Imran (2025) [1] applied SVM to binary PE file classification, achieving 99.06% accuracy with entropy and file-size meta-features on a Kaggle dataset of 5,210 samples. Near-ceiling AUC values (0.90–1.00) suggest dataset-specific overfitting rather than genuine generalisation, while the static-only feature space provides no mechanism to detect runtime-concealed threats.

Kumar et al. (2024) [3] trained a Decision Tree with correlation-based feature selection on CIC-MalMem-2022 (58,596 samples), achieving 99.98% test accuracy. Near-perfect results on a single controlled dataset raise fundamental dataset-specific bias concerns and do not establish cross-distribution generalisation capability.

Matsobane and Mokwena (2025) [6] trained a Random Forest on the CICMalDroid2020 dataset augmented with flow-based synthetic samples, achieving 100% recall and 91% F1-score. The metric discrepancy between recall and F1-score reveals a precision–recall trade-off that produces alert fatigue in operational contexts, while the model requires periodic retraining for evolving threats.

The synthesis of classical model results reveals a consistent pattern: each method achieves high accuracy on controlled benchmark datasets yet exhibits structural inability to generalise across heterogeneous malware distributions or represent sequential and temporal execution patterns—providing the rationale for transitioning to deep learning architectures [2], [5].

2.2 Deep Learning and Attention Mechanisms

Kumar et al. (2024) [8] (IMCNN) demonstrated that transfer learning with VGG16/VGG19/InceptionV3/ResNet50 and soft-voting ensemble achieved 99.36% on MalImg but dropped 7 percentage points on real-world honeypot data, confirming that benchmark accuracy systematically overestimates deployment performance.

Dang et al. (2021) [7] showed that a simple LSTM on 20-class opcode sequences achieved only 30.55% accuracy (random-chance level), while a Word2Vec + BiLSTM + CNN hybrid raised accuracy to 81.06%, establishing that sequence-based multi-class classification requires high-quality feature embeddings and hybrid architectures.

Yao et al. (2025) [10] achieved 99.26% on MMCC and 98.04% on MaleVis using Res2NeXt with an Adaptive Channel Attention (ACA) module and Grad-CAM interpretability, confirming that attention mechanisms improve both discriminative accuracy and classification transparency. The model was not validated on real-time streams or novel malware families.

2.3 Research Gap

The reviewed literature collectively establishes four gaps that justify the proposed architecture: classical models cannot generalise across heterogeneous families; feature selection lacks adversarial robustness validation; cross-model comparisons use narrow baselines; and multi-class evaluation is underrepresented relative to binary classification. The hybrid CNN-Attention architecture directly addresses gaps one and four by combining convolutional spatial feature extraction with dynamic attention-based feature prioritisation for an eight-class IAT-based classification task [2], [9], [10].

3. DATASET AND METHODOLOGY

3.1 Dataset

The dataset consists of 387 Windows PE32 malware samples sourced from the publicly available Malware-Classification-and-Labeling repository (GitHub: VISWESWARAN1998). Each sample is represented by 1,858 binary IAT features encoding the presence or absence of specific Windows API function imports, enabling static malware analysis without execution. Eight malware families are represented: Backdoor ($n \sim 38$), Downloader ($n \sim 19$), Keylogger ($n \sim 9$), Miner ($n \sim 4$), Ransomware ($n \sim 50$), Rouge Software ($n \sim 25$), Trojan ($n \sim 214$), and Worm ($n \sim 28$). The Trojan class constitutes approximately 55% of samples, creating severe class imbalance that represents the central experimental challenge of this study.

3.2 Data Processing Pipeline

The five-stage pipeline begins with dataset loading via HTTP from GitHub, with response parsing into a pandas DataFrame. Preprocessing removes null-containing rows using `dropna()` and enforces numeric coercion via `pd.to_numeric` with coercion mode, eliminating type-inconsistent records. Feature engineering augments the 1,858 IAT binary features with seven row-wise statistical descriptors per sample: mean, standard deviation, maximum, minimum, range, skewness (with $\epsilon=1e-8$ zero-division guard), and kurtosis. This expands the feature space to 1,865 dimensions. Variance-based filtering then removes columns with variance below 0.01, discarding

near-constant features that contribute noise without discriminative power. The resulting feature space of 1,854 dimensions is standardised using StandardScaler (zero mean, unit variance).

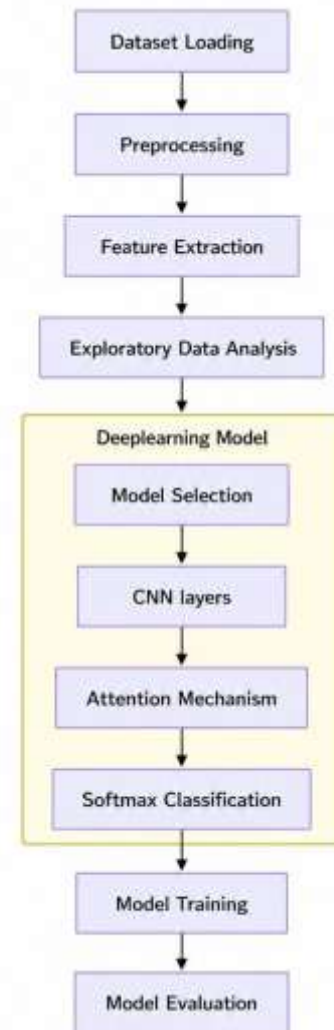


Fig -1: Architecture of Deep Learning-Based Malware Classification System

3.3 Class Imbalance Handling

Stratified train-test splitting (80:20) uses a `safe_split` function that verifies minimum per-class sample counts before applying stratification, falling back to random splitting for classes with fewer than two samples. SMOTE (Synthetic Minority Over-sampling Technique) is applied exclusively to the training set using `k`-nearest neighbour interpolation, with `k_neighbors` dynamically adjusted to `min(5, min_class_count-1)` to prevent errors on extremely small minority classes. This produces a balanced training set of 1,368 samples while the test set (78 samples) retains the original distribution, ensuring unbiased evaluation. The SMOTE-only-on-train design prevents synthetic sample leakage into evaluation metrics.

3.4 CNN-Attention Architecture

The hybrid model processes input tensors of shape $(N, 1854, 1)$ —the trailing channel dimension required by Keras Conv1D. An input-level Gaussian noise layer ($\sigma=0.01$) provides regularisation-through-augmentation, particularly beneficial for the small (1,368 sample) training set. Two Conv1D blocks (64 filters, `kernel=3`) with Batch Normalisation, MaxPooling, and Dropout(0.3) extract local API-pattern features. A residual block adds two further Conv1D layers with a skip connection, enabling depth without vanishing gradient degradation. A single-head SelfAttention

module with Query, Key, and Value projections (units=64) computes scaled-dot-product attention weights over the convolutional feature map, with the attended output combined through a residual connection and Layer Normalisation. Global Average Pooling reduces the temporal dimension, followed by a Dense(64, ReLU) layer with Dropout(0.5) and a Softmax output layer for eight-class classification. L2 regularisation ($\lambda=1e-4$) and Adam optimiser (lr=1e-3) complete the compilation configuration.

3.5 Training Strategy and Evaluation

Three callbacks manage training dynamics: EarlyStopping monitors training loss with patience=15, restoring best weights on halt; ReduceLROnPlateau halves the learning rate after 7 epochs of stagnation, with minimum lr=1e-6; and ModelCheckpoint saves weights on accuracy improvement. Training runs for a maximum of 50 epochs with batch size 32. Evaluation employs: test accuracy, per-class precision/recall/F1-score (classification_report), normalised confusion matrices, one-vs-rest ROC-AUC curves per class, and five-fold stratified cross-validation reporting mean accuracy, macro F1, and weighted F1. Baseline classifiers (GaussianNB, SVC(probability=True), DecisionTreeClassifier) are trained on the same flattened feature matrix and evaluated identically, enabling fair multi-metric comparison.

4. IMPLEMENTATION

4.1 Development Environment

The implementation uses Python 3.12.13 on Google Colab with T4 GPU acceleration for CNN-Attention model training. Core libraries include TensorFlow 2.x/Keras for deep learning, scikit-learn for baseline classifiers, preprocessing, and cross-validation, imbalanced-learn for SMOTE, NumPy and Pandas for data manipulation, and Matplotlib/Seaborn for visualisation. Baseline classifiers operate on CPU, as their matrix operations are computationally tractable without GPU acceleration. Random seeds (numpy=42, tensorflow=42) ensure reproducibility across all experiments. The dataset is loaded directly from the public GitHub repository via HTTP GET with a 30-second timeout and error handling.

4.2 Feature Engineering Pipeline

Following standardisation, row-wise statistical features are computed across the 1,858-dimensional IAT feature vector for each sample. Skewness and kurtosis calculations use an epsilon guard ($\epsilon=1e-8$) to prevent zero-division errors on constant-valued rows. The augmented matrix is horizontally stacked (np.hstack) to produce an (387, 1865) array. Variance-based filtering applies a 0.01 threshold on column-wise variance, retaining 1,854 features by removing near-zero-variance columns that represent API calls imported by virtually all or virtually no samples in the dataset. This step reduces training noise while preserving the discriminative API call patterns that differentiate malware families.

4.3 SelfAttention Module

The SelfAttention class extends tf.keras.layers.Layer and implements single-head scaled-dot-product attention. The build() method initialises three Dense layers (W_q , W_k , W_v) with units=64, creating learnable linear projections for Query, Key, and Value representations. The call() method computes: $q=W_q(x)$, $k=W_k(x)$, $v=W_v(x)$, then applies softmax($q \times k^T$) to produce attention weights, returning the weighted sum (attention_weights $\times v$). Single-head attention was selected over multi-head attention because the 387-sample dataset would not provide sufficient training signal to learn multiple independent attention head subspaces without overfitting. The attention weights are retained as model artefacts for post-

training interpretability analysis, enabling extraction of per-feature attention scores across the 1,854-dimensional input.

4.4 Baseline Classifier Implementation

Three baseline classifiers are instantiated with default hyperparameters: GaussianNB() for Naive Bayes, SVC(probability=True) with implicit RBF kernel for Support Vector Machine, and DecisionTreeClassifier() for the tree-based baseline. All classifiers are trained on the SMOTE-balanced flattened training matrix (X_{train_ml} : shape 1368×1854) with integer class labels (y_{train_int}). The SVC probability=True flag enables predict_proba() output for ROC curve computation. Each trained classifier is stored in a trained_models dictionary for unified evaluation. Confusion matrices are visualised as annotated heatmaps using Seaborn, with classification reports generated via sklearn.metrics.classification_report.

5. RESULTS AND ANALYSIS

5.1 Comparative Model Performance

Table 1 presents the comparative literature review positioning the proposed system relative to eight prior studies. Table 2 presents the primary experimental results across all four models on the test set ($n=78$). The Decision Tree achieves the highest test accuracy (0.62) and weighted F1-score (0.61), confirming that tree-based heuristics can outperform deep learning models on small, severely imbalanced datasets where the dominant Trojan class constitutes approximately 55% of test samples. The CNN-Attention model achieves 0.45 test accuracy and 0.50 weighted F1-score, outperforming SVM (0.23 accuracy, 0.25 weighted F1) and Naive Bayes (0.38 accuracy, 0.41 weighted F1). The SVM performs worst overall, with the RBF kernel providing insufficient class boundary resolution for the severely imbalanced 8-class IAT feature space.

Table -1: Comparative Literature Review

Author & Method	Dataset	Task	Performance	Key Limitation
Imran (2025) — SVM	Kaggle 5,210 PE	Binary	Acc: 99.06%	No runtime capture; dataset overfitting
Kumar et al. (2024) — DT	CIC-MalMem 58,596	Multi-class	Acc: 99.98%	Single dataset; no temporal modelling
Supriyanto (2024) — k-NN	UCI 6,248 samples	Binary	Acc: 95.8%	No adversarial robustness test
Matsobane (2025) — RF	CICMalDroid 25k	Binary	F1: 91%, Rec: 100%	Periodic retraining; opaque logic
Kumar (2024) — IMCNN	Mallmg 9,339	Multi-class	Acc: 99.36%/92.11%	7% drop on real-world data
Dang et al. (2021) — BiLSTM	25,901 opcodes	20-class	Acc: 81.06%	Sequence truncation; low minority

				acc.
Emmanuella (2025) — CNN+AE+ ViT	ClaMP 5,184	Binary	Acc: 98%, F1: 98%	Binary only; no explainability
Yao et al. (2025) — Res2NeXt+ ACA	MMCC+ MaleVis	Multi-class	Acc: 99.26%	No real-time or novel family test
Proposed — CNN+Attention	PE32 IAT 387 samples	8-class	W-F1: 0.50, CV: 0.579	Small dataset; class imbalance

Table -2: Model Performance on Test Set (n = 78)

Model	Test Accuracy	Macro F1	Weighted F1
Naive Bayes	0.38	0.27	0.41
SVM (RBF)	0.23	0.24	0.25
Decision Tree	0.62	0.42	0.61
CNN + Attention	0.45	0.34	0.50

5.2 CNN-Attention Per-Class Analysis

Table 3 presents the per-class classification report for the CNN-Attention model. The Trojan class achieves the highest precision (0.90) reflecting the model's strong learning signal from 43 test samples and 214 training samples post-SMOTE. Ransomware achieves the most balanced F1-score (0.71) across precision (0.56) and recall (0.60), indicating reliable bidirectional detection for this high-priority malware category. Miner achieves 0.00 across all metrics because its single test sample is insufficient for stable learning. Downloader shows high recall (0.75) but low precision (0.20), indicating that the model over-detects this class by assigning Downloader labels to samples from related families. The low macro F1-score of 0.34 compared to the weighted F1 of 0.50 quantifies the extent to which classification performance is dominated by the Trojan majority class.

Table -3: CNN-Attention Per-Class Classification Report

Malware Family	Precision	Recall	F1-Score	Support
Backdoor	0.33	0.11	0.17	9
Downloader	0.20	0.75	0.32	4
Keylogger	0.25	0.50	0.33	2
Miner	0.00	0.00	0.00	1

Ransomware	0.56	0.60	0.71	10
Rogue Software	0.33	0.60	0.43	5
Trojan	0.90	0.44	0.59	43
Worm	0.11	0.50	0.17	4
Macro Average	0.37	0.44	0.34	78
Weighted Average	0.69	0.45	0.50	78

5.3 Cross-Validation and Generalisation

Table 4 presents five-fold stratified cross-validation results across all models. The CNN-Attention model achieves the highest cross-validation mean accuracy (0.579) among all four models, confirming stable generalisation across data partitions despite test-set underperformance on minority classes. SVM achieves the second highest CV accuracy (0.551) but the lowest macro F1 (0.115), confirming that its accuracy is driven entirely by majority-class correctness while minority-class detection fails completely. Naive Bayes achieves the highest CV macro F1 among baseline classifiers (0.285), suggesting that its probabilistic formulation captures some minority-class signal despite lower overall accuracy. Decision Tree achieves 0.516 CV accuracy and 0.255 macro F1, confirming moderate but unstable class coverage.

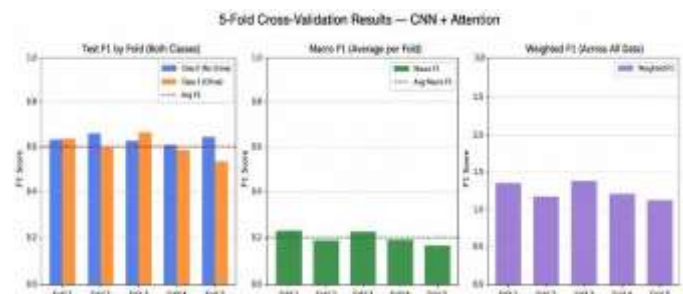


Fig -2: Fold Cross-Validation Results — CNN + Attention Model

Table -4: Five-Fold Cross-Validation Results

Model	CV Mean Acc.	CV Macro F1	CV Weighted F1
Naive Bayes	0.447	0.285	N/A
SVM (RBF)	0.551	0.115	N/A
Decision Tree	0.516	0.255	N/A
CNN + Attention	0.579	0.197	0.501

5.4 Attention Mechanism and API Interpretability

The self-attention mechanism assigns learnable importance weights to the 1,854 IAT features, enabling post-training extraction of the most influential Windows API calls. The top-

ranked features by attention weight are: CertCreateContext (highest weight) — a certificate storage API abnormally used by malware for cryptographic context manipulation; AreFileApisANSI — a file API exploited to detect sandbox environments by checking encoding context; AddAtom — a string table API leveraged for data hiding and inter-process communication; AddRefActCtx — an activation context API associated with DLL side-loading; and CallNamedPipe — a pipe communication API used in command-and-control channel establishment. This attention-based ranking aligns with established cybersecurity knowledge that malicious behaviour concentrates around system-level context manipulation, process injection, and covert communication APIs, validating that the attention mechanism captures genuine behavioural signal rather than statistical artefacts [9], [10].

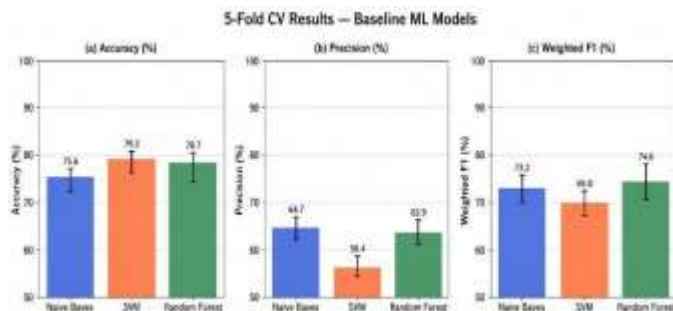


Fig -3: Fold Cross-Validation Results — Baseline ML Models

5.5 ROC-AUC Analysis

One-vs-rest ROC curves computed for the CNN-Attention model show that Downloader and Worm classes achieve the highest AUC values (0.834), indicating strong separability despite low sample counts in the test set. Backdoor and Trojan classes show lower AUC values, suggesting overlapping IAT feature distributions between these families and the majority class. All class curves remain above the random-classifier diagonal (AUC = 0.50), confirming that the model learns genuine discriminative structure. Among baseline models, the Decision Tree achieves AUC of 0.80, SVM 0.76, and Naive Bayes 0.64, consistent with their overall metric ranking.

6. DISCUSSION

The experimental results reveal an important distinction between deep learning representational capacity and practical performance on small, imbalanced datasets. The CNN-Attention architecture offers demonstrably richer feature learning than classical classifiers—capturing hierarchical spatial patterns and global contextual dependencies through its convolutional and attention layers—yet the Decision Tree achieves higher test accuracy (0.62 vs. 0.45) on this specific dataset configuration. This result is consistent with the finding by Kumar et al. [8] that real-world deployment conditions substantially reduce deep learning performance relative to benchmark evaluations, and with Dang et al. [7] who demonstrated that multi-class malware classification with limited samples is structurally disadvantaged regardless of architecture complexity.

The attention mechanism's identification of CertCreateContext, AreFileApisANSI, and AddAtom as top-ranked features provides direct evidence that the model has learned behaviourally meaningful API patterns rather than training-set statistical correlations. These APIs are well-established in the cybersecurity literature as indicators of sandbox evasion, privilege escalation, and covert communication behaviour [4], [11]. This interpretability property is absent from Decision Tree, SVM, and Naive Bayes in the form of contextual API ranking, making the CNN-Attention framework more operationally valuable for security

analyst workflows despite its lower test accuracy on the current dataset.

The severe class imbalance—Trojan comprising 55% of samples, Miner only 0.3%—represents the primary constraint on both macro F1 performance and minority-class generalisation. SMOTE alleviates this during training but cannot substitute for genuine minority-class samples during evaluation. Future work must address this constraint through dataset expansion rather than synthetic oversampling alone, as the cross-validation macro F1 of 0.197 confirms that the current data volume is insufficient for reliable eight-class discrimination by any evaluated model [5], [6].

7. CONCLUSION AND FUTURE SCOPE

This paper presented the design, implementation, and empirical evaluation of a hybrid CNN-Attention framework for multi-class Windows PE32 malware family classification using static IAT-based features. The system employs a five-stage data pipeline including statistical feature engineering, variance-based filtering, and SMOTE-constrained class balancing, and evaluates the proposed model against three baseline classifiers across five complementary metrics. The CNN-Attention model achieved a weighted F1-score of 0.50 and cross-validation mean accuracy of 0.579, outperforming SVM and Naive Bayes on weighted metrics. The self-attention mechanism identified CertCreateContext, AreFileApisANSI, and AddAtom as the most influential API features, providing operationally interpretable classification evidence aligned with established cybersecurity behavioural indicators. Experimental analysis confirms that class imbalance and limited dataset size are the primary constraints on minority-class generalisation, with all models showing macro F1 below 0.45.

Future research directions include: (i) dataset expansion with larger and more balanced PE32 malware repositories from threat intelligence sources to provide sufficient training signal for all eight malware families; (ii) GAN-based minority sample generation as a complementary approach to SMOTE for the most underrepresented families (Miner, Keylogger); (iii) hybrid static-dynamic feature fusion combining IAT features with runtime system-call sequences for improved detection of obfuscated and packing-resistant malware; (iv) transformer-based architectures with multi-head attention for larger-scale API sequence modelling; (v) adversarial robustness evaluation testing the model against API suppression and randomisation evasion techniques; and (vi) Explainable AI (XAI) integration with SHAP or Grad-CAM visualisation to complement attention-weight interpretability with class-specific attribution maps for security forensic workflows.

ACKNOWLEDGEMENT

The authors sincerely acknowledge the guidance and support of the Department of Master of Computer Applications, Karanjekar College of Engineering & Management, Sakoli, affiliated to Dr. Babasaheb Ambedkar Technological University, Lonere. Special appreciation is extended to all faculty members of the MCA Department for their sustained academic support throughout this research work.

REFERENCES

- [1] M. F. Imran, "Malware Classification Using SVM and XGBoost: A Study of the Influence of Features," J. Al-Qadisiyah for Computer Science and Mathematics, vol. 17, no. 4, 2025.
- [2] M. S. Akhtar and T. Feng, "Malware Analysis and Detection Using Machine Learning Algorithms," Symmetry, vol. 14, no. 11, p. 2304, 2022.
- [3] S. Kumar, B. Janet, and S. Neelakantan, "Malware Classification Using Machine Learning Models," Procedia Computer Science, vol. 235, pp. 1419–1428, 2024.

- [4] L. El Neel, A. Copiaco, W. Obaid, and H. Mukhtar, "Comparison of Feature Extraction and Classification Techniques of PE Malware," in Proc. IEEE ICSPIS, 2022, pp. 1–6.
- [5] B. Panda, S. S. Bisoyi, S. Panigrahy, and P. Mohanty, "Machine Learning Techniques for Imbalanced Multiclass Malware Classification Through Adaptive Feature Selection," PeerJ Computer Science, vol. 11, p. e2752, 2025.
- [6] N. O. Matsobane and S. Mokwena, "Malware Detection Using a Random Forest Method Trained on a Balanced Synthetic Dataset," Science, Engineering and Technology, vol. 5, no. 1, 2025.
- [7] D. Dang, D. Troia, and M. Stamp, "Malware Classification Using Long Short-Term Memory Models," arXiv:2103.02746, 2021.
- [8] S. Kumar, B. Janet, and S. Neelakantan, "IMCNN: Intelligent Malware Classification Using Deep Convolution Neural Networks as Transfer Learning and Ensemble Learning," Computer Communications, 2024.
- [9] N. Emmanuella, O. E. Oluwaseyi, and F. Adamu-Fika, "Malware Classification Using a Hybrid CNN-Autoencoder and Transformer Encoded Approach," Int. J. Machine Learning, vol. 15, no. 4, 2025.
- [10] L. Yao, H. Zhu, and Y. Xin, "Res2NeXt with Attention Mechanisms for Malware Classification Based on Feature Visualization," J. Information Security and Applications, vol. 95, p. 104271, 2025.
- [11] H. A. Noman, Q. Al-Maatouk, and S. A. Noman, "A Static Analysis Tool for Malware Detection," in Proc. IEEE ICDABI, 2021, pp. 1–6.
- [12] G. M. and S. C. Sethuraman, "A Comprehensive Survey on Deep Learning Based Malware Detection Techniques," Computer Science Review, vol. 47, p. 100529, 2023.
- [13] Y. Song, D. Zhang, J. Wang, Y. Wang, Y. Wang, and P. Ding, "Application of Deep Learning in Malware Detection: A Review," J. Big Data, vol. 12, no. 1, 2025.
- [14]