Technical Report on Using a CNN for Malware Classification

Jeffrey Jeyachandren, Jia Hua Li, Amber Goel

Abstract

This report presents the implementation and evaluation of a Convolutional Neural Network (CNN) for malware classification, as pioneered by Mclaughlin et al. (2017). The analysis leverages opcode sequence data transformed into n-gram representations to explore the efficacy of deep learning techniques in identifying the APT groups responsible for malicious software. The report outlines the data preparation steps, model architecture, training process, and results. Comparisons to traditional machine learning classifiers highlight the advantages of CNNs in capturing complex patterns within opcode-derived data, achieving an accuracy rate of ~90%. Source code: https://github.com/JeffinWithYa/apt_analysis_and_samples

Implementation Methodology

To prepare the malware opcode data for training and testing the classifier, the following pre-processing steps were performed. The processing of the opcodes follows the same methodology previously used for traditional machine learning classifiers and aligns with the approach for constructing n-grams described in the reference paper by McLaughlin et al. (2017).

These steps are the similar to the steps used to train the traditional machine learning classifiers (SVM, KNN, Decision-Tree), but are repeated here for clarity:

1. Data Collection and Organization

- The opcode data was stored in directories organized by Advanced Persistent Threat (APT) groups. Each directory represented a specific APT group and contained text files with opcode sequences extracted from malware samples associated with that group.

2. Reading and Structuring the Data

A script was implemented to navigate through the directory structure, read each opcode file, and extract the opcode sequences:

- Opcodes within each file were separated by newline characters. These were combined into comma-separated strings to ensure a uniform representation.
- The extracted opcode sequences were organized in a DataFrame, paired with their corresponding APT group labels for classification.
- Example:

	Opcodes	APT
0	JMP, JMP, JMP, JMP, JMP, JMP, JMP, JMP, POP, P...	Evilnum
1	PUSH, MOV, MOV, CALL, PUSH, PUSH, PUSH, PUSH, ...	APT19_opcodes
2	MOV, PUSH, MOV, CALL, TEST, PUSH, CALL, ADD, M...	APT19_opcodes
3	PUSH, PUSH, PUSH, MOV, CMP, JE, CMP, JNE, PUSH...	APT19_opcodes
4	PUSH, PUSH, PUSH, MOV, CMP, JE, CMP, JNE, PUSH...	APT19_opcodes

4. Feature Engineering

- To convert opcode sequences into a numerical format suitable for machine learning, n-gram features were extracted:
    - **1-gram Features**:
        - Individual opcodes were identified as unique tokens across the dataset.
        - A frequency matrix was generated, where each row corresponded to a malware sample, and columns represented the unique opcodes. The matrix values indicated the frequency of each opcode within a sample.
    - **2-gram Features**:
        - Consecutive opcode pairs (bigrams) were extracted from the opcode sequences.
        - A frequency matrix was created for bigrams, with rows representing samples and columns representing unique bigrams, capturing the frequency of each bigram in the dataset.
    - **3-gram and Combined Features:**
        - The 3-gram features extend the n-gram analysis by considering sequences of three consecutive opcodes, capturing more complex patterns in the malware's behavior. Each unique triplet of opcodes (e.g., "PUSH-MOV-CALL") becomes a feature column, with the frequency matrix recording how often each triplet appears in each sample. This approach helps identify longer operational patterns that might be characteristic of specific APT groups.
        - The combined feature set concatenates the 1-gram, 2-gram, and 3-gram matrices, creating a comprehensive representation that preserves information at multiple sequence lengths. From our notebook results, the combined features achieved higher accuracy (showing improved classification performance) compared to individual n-gram features, suggesting that the different sequence lengths capture complementary information about the malware samples. However, this comes at the cost of increased feature dimensionality - while a 1-gram might have hundreds of features, the combined representation can have thousands, requiring more computational resources for training. The confusion matrices from our balanced dataset show that this richer feature representation helps the model better distinguish between different APT groups, particularly for groups with similar individual opcode frequencies but different sequential patterns.

Hybrid Oversampling with ROS and SMOTE

The dataset exhibited significant class imbalances, with certain APT groups being notably underrepresented. This imbalance posed a risk of the classifier favoring the majority class, thereby reducing its ability to accurately detect minority classes. Such oversight could lead to missing less common but critical malware types (Kamalov et al., 2023). To address this challenge, a hybrid oversampling strategy was employed, starting with Random Oversampling

and followed by the Synthetic Minority Oversampling Technique (SMOTE). This led to a balanced distribution of classes in the data set, as seen in **Figure 6**.

Random Oversampling (ROS)

Random Oversampling was used as the initial step to tackle extreme class imbalances. This technique works by duplicating samples from minority classes, ensuring that all classes have at least a baseline number of representatives in the training data.

Advantages of ROS:

- Increases the classifier's exposure to minority class samples, ensuring balanced representation during training.
- Simplifies the learning process by equalizing the class distribution (Kamalov et al., 2023).

Limitations of ROS:

- May lead to overfitting, as repeated samples can cause the model to memorize rather than generalize.
- Expands the dataset size, increasing training time and computational demands.

ROS was applied first because it addresses situations where certain classes have very few or even a single sample. Without this initial step, SMOTE cannot function effectively, as it requires multiple samples to interpolate and generate synthetic data.

Synthetic Minority Oversampling Technique (SMOTE)

Once ROS established a baseline for minority class representation, SMOTE was applied to further enhance the dataset. SMOTE generates synthetic samples by interpolating between existing samples within the same class, creating a more diverse and realistic representation of the minority classes (Alkhawaldeh et al., 2023).

SMOTE Parameters:

- `random_state=42`: Ensures reproducibility of the synthetic data generation process.

Similarities and Differences with Reference Paper (Mclaughlin, et al. 2017)

Our approach shares several methodological similarities with the reference paper by McLaughlin et al., while introducing key differences to adapt the methodology to our specific classification task and dataset. Both studies trained the CNN model for 10 epochs with a learning rate of 1×10^{-2}, leveraging an embedding layer to process raw opcode sequences. This approach preserves the sequential nature of the opcodes, as opposed to frequency-based representations, which can lose order-dependent information. However, despite maintaining this similarity, our experiments with raw opcode sequences yielded poor results even after

addressing class imbalances through oversampling. To improve performance and enable a fair comparison with traditional classifiers, we employed frequency-based n-gram features (1-gram, 2-gram, 3-gram, and combined) alongside the CNN, which significantly improved classification accuracy.

Several differences set our approach apart. While the original paper utilized NVIDIA GPUs for training, our experiments were conducted on an M1 Mac. Additionally, the reference paper worked with malware datasets sourced confidentially from McAfee Labs, focusing on binary malware classification (malware vs. benign). In contrast, we aimed to classify malware samples into APT groups, adapting the methodology and dataset to accommodate this multi-class classification task. Our training data was labeled to reflect APT groups, enabling a direct evaluation of CNN performance alongside traditional classifiers for this specific use case. Finally, while the reference study implemented their model using Torch, a low-level library from 2017, we used Keras, a high-level library that simplifies implementation and abstracts much of the complexity, making the training process more accessible and streamlined. These adaptations ensure the methodology is aligned with both the unique goals and constraints of our research.

Results

Classifier	Pre-Balance Accuracy	Pre-Balance Precision	Pre-Balance Recall	Pre-Balance F1-Score	Post-Balance Accuracy	Post-Balance Precision	Post-Balance Recall	Post-Balance F1-Score
SVM	0.3023	0.0400	0.0800	0.0500	**0.7361**	**0.8595**	**0.7361**	**0.7188**
KNN-3	0.4651	0.200	0.2600	0.2100	**0.8958**	**0.8723**	**0.8958**	**0.8690**
KNN-5	0.2791	0.1200	0.1500	0.1200	**0.8958**	**0.8746**	**0.8958**	**0.8695**
Decision Tree	0.3953	0.1400	0.1800	0.1600	**0.5000**	**0.4142**	**0.5000**	**0.4307**
CNN opcode frequency	0.5814	0.36	0.32	0.31	**0.9028**	**0.8700**	**0.9100**	**0.8800**
CNN raw opcodes	-	-	-	-	**0.0233**	**0.0000**	**0.0600**	**0.0000**

Figure 1: Classifier Results for Combined Data (1-gram + 2-gram or 1+2+3-gram for CNN)

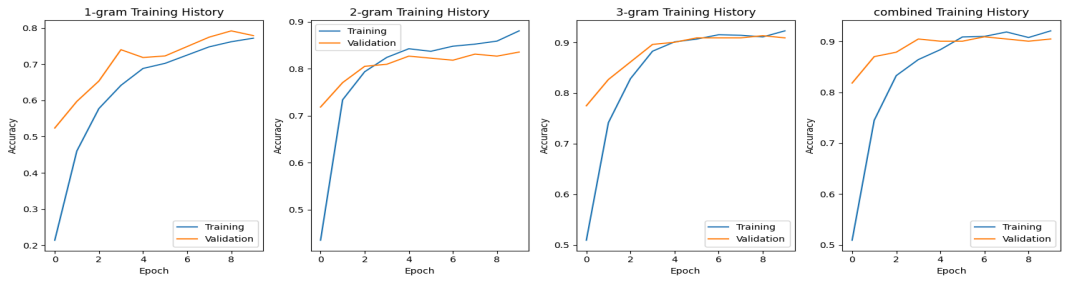

Figure 2: Accuracy Improving with Training Epochs for CNN Classifier

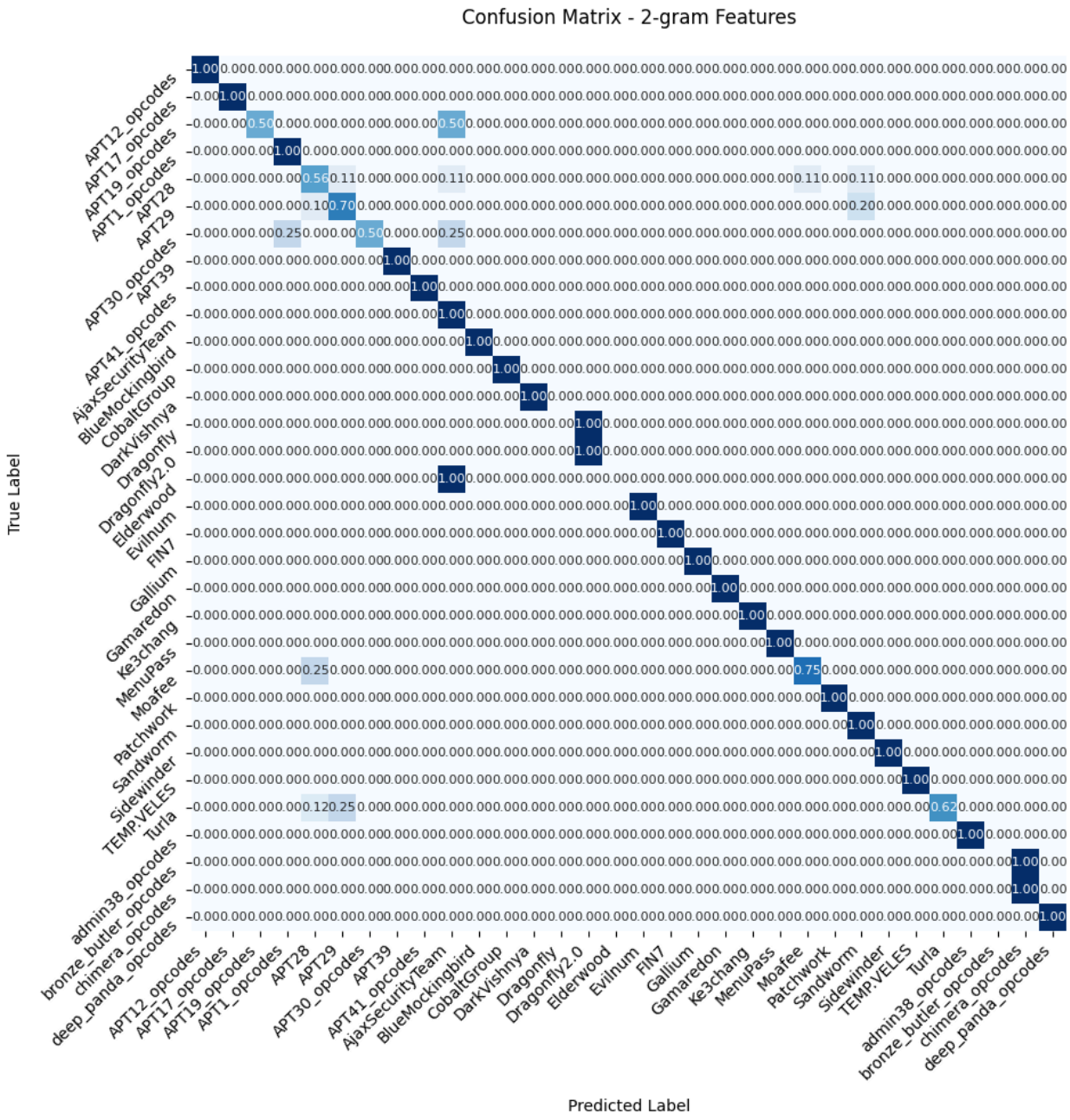

Figure 3: Confusion matrix for CNN trained on 2-gram opcodes from balanced data set

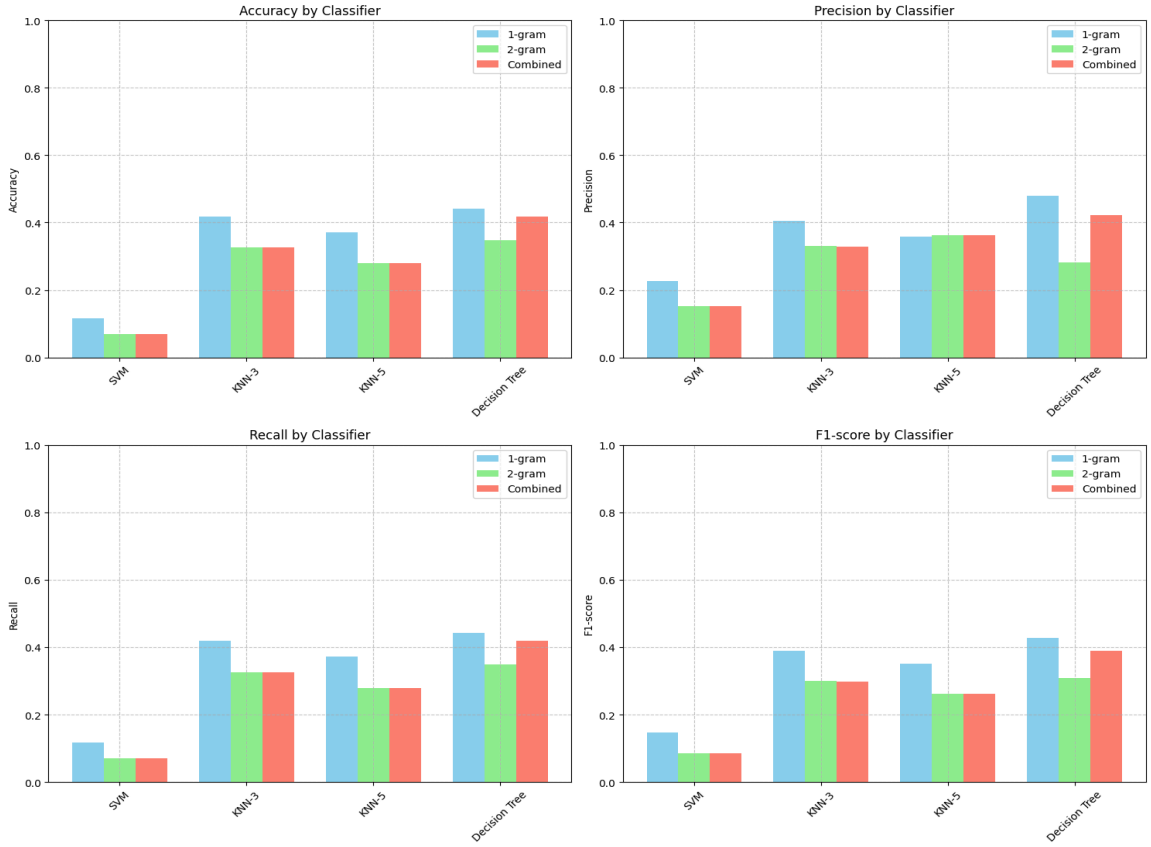

Figure 4: Results for Traditional Classifiers Trained on Unbalanced Data

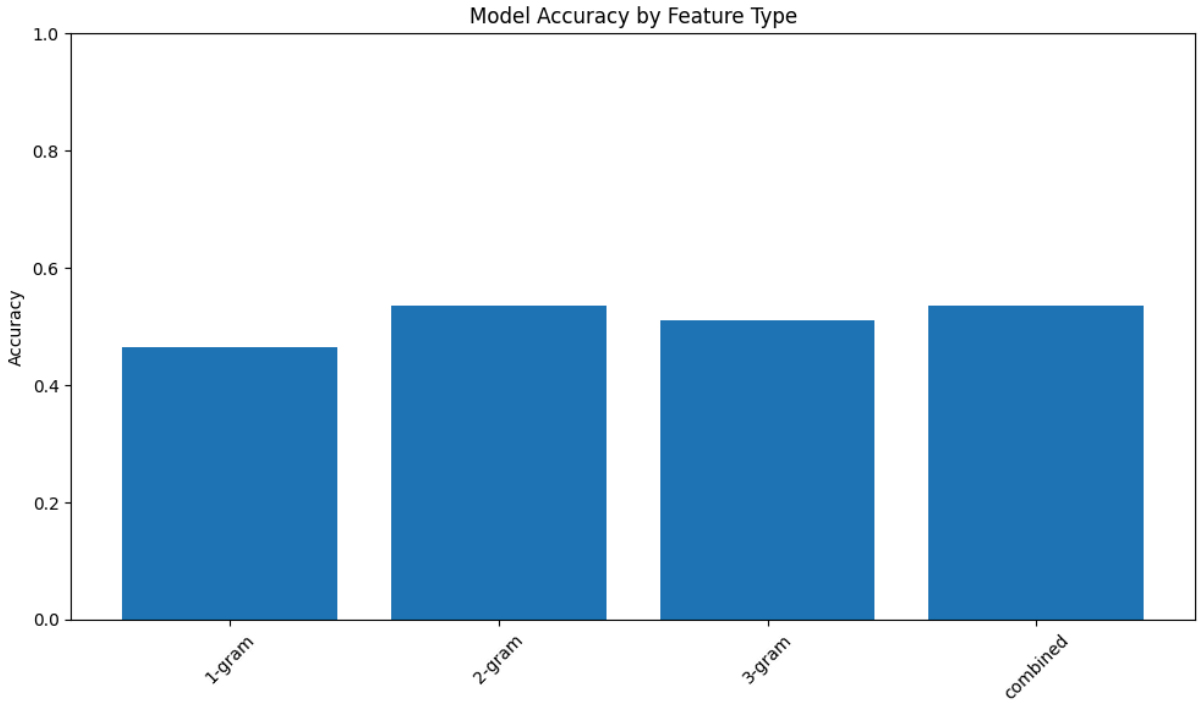

Figure 5: Results for CNN Classifiers Trained on Unbalanced Data

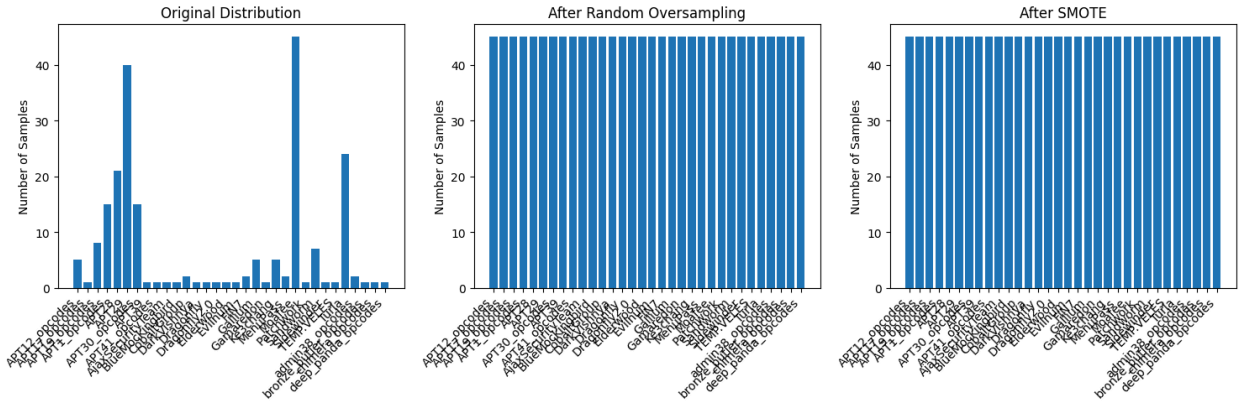

Figure 6: Unbalanced class distributions and balanced data set after ROS + SMOTE

Results

The performance of the classifiers on the pre-balanced and post-balanced datasets demonstrates the effectiveness of the feature engineering and hybrid oversampling strategies employed in this study (**Figure 1)**. The bar graphs in **Figure 4 and Figure 5** further highlight the performance differences between traditional classifiers and the CNN classifier from Mclaughlin et al. 2017. All diagrams that highlight the results can be found in the accompanying Jupyter Notebook, 'Malware_CNN_Notebook.ipynb', which includes all confusion matrices and result graphs from this report.

Comparison of Classifier Performance

Among the classifiers, the CNN model trained on opcode frequency features achieved the highest overall performance after balancing, with an accuracy of 0.9028 and an F1-score of 0.8800. This indicates the effectiveness of leveraging frequency-based representations combined with the hybrid oversampling approach to address class imbalances. In contrast, the CNN model trained on raw opcode sequences performed poorly, with a post-balance accuracy of only 0.3721 and an F1-score of 0.42. This result highlights the importance of feature engineering in capturing meaningful patterns for malware classification.

The SVM, KNN, and Decision Tree classifiers showed varying levels of improvement post-balancing, with KNN-3 achieving an accuracy of 0.8958 and an F1-score of 0.8690, closely trailing the CNN. While these traditional classifiers benefited from balancing and engineered features, they were outperformed by the CNN, emphasizing the advantages of deep learning in capturing complex patterns in malware behavior.

Contrasting Results with the Previous Paper

In our previous work, where 1-gram and 2-gram features were primarily used, the best-performing traditional classifiers were lower than the performance of the CNN in the current study. The inclusion of 3-gram features and a combined feature set in this study significantly

enhanced the ability to distinguish between APT groups with subtle sequential differences in opcode usage. The CNN approach, inspired by the methodology outlined by McLaughlin et al., clearly demonstrates a significant advantage over traditional classifiers due to its ability to automatically learn hierarchical features directly from the data. Unlike traditional models like SVM, KNN, or Decision Trees, which rely on manually engineered features, the CNN's convolutional layers identify complex spatial and sequential patterns in the opcode frequency matrices, making it better suited for nuanced classification tasks like malware detection. **Figure 2** illustrates the CNN's learning trajectory, with the accuracy steadily improving across training epochs and plateauing at its peak around the 10th epoch. This progressive improvement highlights the model's capacity to optimize feature extraction as it iteratively refines its filters, contrasting with traditional classifiers that often struggle to generalize as effectively with high-dimensional data. This adaptability enables the CNN to leverage the combined n-gram feature set more efficiently, yielding superior classification performance across imbalanced and diverse APT group datasets.

Conclusion

This study demonstrates the efficacy of leveraging CNN models for malware classification, particularly in distinguishing between APT groups. By incorporating frequency-based n-gram features alongside a hybrid oversampling strategy, we achieved significant improvements in classification performance compared to traditional classifiers and raw opcode-based CNN approaches. The results highlight the importance of feature engineering in enhancing the CNN’s ability to capture both sequential and structural patterns within opcode data, enabling more accurate identification of malware types associated with specific APT groups.

Our findings also underscore the adaptability of deep learning models to complex multi-class classification tasks. While the reference paper focused on binary malware detection, we extended their approach to a more granular classification problem, demonstrating the CNN’s versatility when combined with tailored preprocessing and balanced datasets. Moreover, the transition to more modern tools like Keras and hardware like the M1 Mac illustrates the feasibility of implementing high-performing machine learning solutions in resource-constrained environments.

Despite the promising results, this work is not without limitations. The increased feature dimensionality of the combined n-gram approach imposes higher computational costs, and the reliance on frequency-based features sacrifices some of the sequential nuances preserved in raw opcode representations. Future work could explore hybrid models that integrate sequential and frequency-based features, as well as alternative neural network architectures like transformers, to further enhance malware classification accuracy and efficiency.

In conclusion, this research validates the usefulness of CNNs as a robust tool for malware classification in cybersecurity, providing a foundation for more sophisticated machine learning-driven solutions to combat evolving threats. By bridging methodologies from prior work with modern innovations, this study lays the groundwork for future advancements in malware detection and threat intelligence.

References

Alkhawaldeh, I. M., Albalkhi, I., & Naswhan, A. J. (2023). Challenges and limitations of synthetic minority oversampling techniques in machine learning. World Journal of Methodology, 13(5), 373–378. https://doi.org/10.5662/wjm.v13.i5.373

Kamalov, F., Leung, H., & Cherukuri, A. K. (2023). Keep it simple: random oversampling for imbalanced data. 2023 Advances in Science and Engineering Technology International Conferences, ASET 2023, 1–4. https://doi.org/10.1109/aset56582.2023.10180891

McLaughlin, N., Del Rincon, J. M., Kang, B., Yerima, S., Miller, P., Sezer, S., Safaei, Y., Trickel, E., Zhao, Z., Doupé, A., & Ahn, G. J. (2017). Deep Android Malware Detection. *CODASPY '17: Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy*, 301–308. https://doi.org/10.1145/3029806.3029823