Enhancing Malware Detection with TabNetClassifier: A SMOTE-based Approach

Rahimov Faridun[1], Eul Gyu Im[2]
[1]Dept. of Computer Science, Hanyang University
[2]Dept. of Computer Science, Hanyang University
farid01@hanyang.ac.kr, imeg@hanyang.ac.kr

Abstract

Malware detection has become increasingly critical with the proliferation of end devices. To improve detection rates and efficiency, the research focus in malware detection has shifted towards leveraging machine learning and deep learning approaches. This shift is particularly relevant in the context of the widespread adoption of end devices, including smartphones, Internet of Things devices, and personal computers. Machine learning techniques are employed to train models on extensive datasets and evaluate various features, while deep learning algorithms have been extensively utilized to achieve these objectives. In this research, we introduce TabNet, a novel architecture designed for deep learning with tabular data, specifically tailored for enhancing malware detection techniques.

Furthermore, the Synthetic Minority Over-Sampling Technique is utilized in this work to counteract the challenges posed by imbalanced datasets in machine learning. SMOTE efficiently balances class distributions, thereby improving model performance and classification accuracy. Our study demonstrates that SMOTE can effectively neutralize class imbalance bias, resulting in more dependable and precise machine learning models.

1. Introduction

Malware presents a significant threat to data integrity, privacy, and operational continuity, underscoring the importance of robust computer system and network security in the digital era. The term "malware," short for malicious software, encompasses a broad range of software designed to damage or exploit networks, services, or programmable devices. Developing effective countermeasures necessitates a comprehensive understanding of malware analysis. This process involves examining a malware sample to ascertain its functionality, origin, and potential impact. Effective malware defense involves deconstructing and analyzing malicious code to understand its components, execution pathways, and communication methods, with the aim of detecting, mitigating, and ultimately preventing malware attacks.

Malware analysis traditionally employs two principal approaches: dynamic analysis, observing malware in action, and static analysis, examining the malware without execution [1]. While these methods have achieved some success, they face challenges related to scalability, evasion tactics employed by malware authors, and the continuous emergence of new malware variants daily. Consequently, there is a pressing need for more advanced and adaptable solutions to effectively counter these issues.

This introduces DL, a subset of ML characterized by models trained to perform tasks based solely on text, images, or audio data [2]. Deep learning models, particularly those designed for structured data analysis like the innovative TabNet architecture, offer promising avenues for enhancing malware detection. Unlike conventional ML models that often rely heavily on extensive feature engineering, DL models possess the innate ability to autonomously identify complex patterns and relationships directly from the data.

We introduce TabNet, a state-of-the-art deep neural network (DNN) framework designed specifically for tabular data analysis, featuring the following key enhancements:

- TabNet simplifies the data preparation process by directly handling raw tabular data, eliminating the need for preprocessing.
- It employs training based on gradient descent, which enhances learning efficiency and simplifies integration into end-to-end learning pipelines.
- TabNet dynamically allocates computing resources to the most significant features, thereby improving interpretability. It does this by selectively focusing on specific features at each decision step through the application of sequential attention.

2. Literature review

Recently, DL methods have been employed to develop intelligent decision-making machines. Given the ever-evolving nature of sophisticated malware threats, researchers have devised various frameworks for malware detection. The majority of these initiatives concentrate on creating solutions for binary malware detection powered by artificial intelligence.

The application of DL methods for malware classification was examined by Olowoyo et al. [3]. By using a transfer learning strategy and representing malware as images, they are able to classify them with an average accuracy of 98.8%.

An RNN, LSTM, and GRU-based malware classification model was presented by Chen Li et al. [4]. Their findings demonstrated that the suggested RNN model, which analyzes lengthy sequences of API calls, performs well in malware classification.

The use of DL and ML in malware detection was investigated by Rathore et al. [5]. They use supervised and unsupervised learning methods, and they use opcode frequency as a feature vector. The results showed that Random Forest performed better than Deep Neural Networks and that Deep Auto-Encoders perform better than Deep Neural Networks.

Using API calls, Catak et al. [6] carried out study on malware detection. They used LSTM, K-Nearest Neighbors (KNN), Decision Trees (DT), and Support Vector Machines (SVM) as part of their machine learning method and contrasted shallow learning with deep learning algorithms. Out of the eight malware classes in the dataset, LSTM had the best accuracy and F1-score.

3. TabNet Architecture Overview

TabNet is a deep learning architecture [7] meticulously devised for the nuanced domain of tabular data. It excels in extracting intricate patterns and achieving interpretability, a trait seldom seen in complex models. Below we detail the architecture's core components:

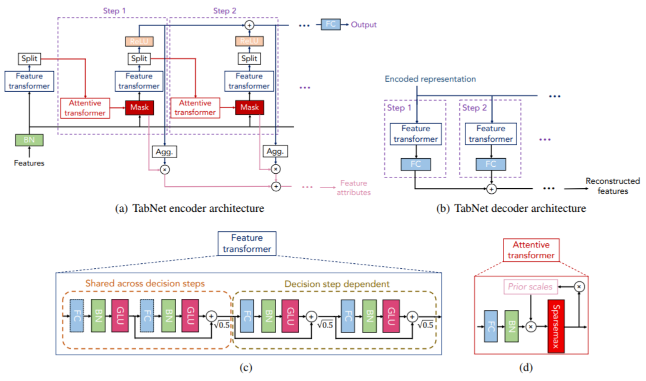
The Encoder Architecture: As depicted in Figure 1a, the TabNet encoder initiates by normalizing raw input features through Batch Normalization (BN). This standardization ensures consistent data flow throughout the network. The encoder's architecture is segmented into several decision stages, each responsible for a segment of the output. Within these stages, two critical modules operate: The *Feature Transformer*, executing non-linear transformations of the data, and the *Attentive Transformer*, which dynamically assigns weights to features. The model's predictive capabilities are progressively enhanced by the data's divided paths, which persist through subsequent decision steps and cumulatively contribute to the final output.

The Decoder Architecture: In addition to the encoder's role, the decoder, as shown in Figure 1b, is tasked with reconstructing the input features. It enhances the encoding stage by minimizing the reconstruction error, thereby retaining the most informative features for the model's predictions.

Feature Processing: As illustrated in Figure 1c, the Feature Transformer architecture encompasses both shared and step-dependent layers. The step-dependent layers introduce flexibility and allow for fine-tuning at each decision point, whereas the shared layers provide a stable foundation for transformation across the model.

The Attentive Transformer with Prior Scales: A distinctive feature of TabNet is its attentive transformer, which utilizes prior scales to modulate attention across features, ensuring a comprehensive method for feature selection. This functionality is illustrated in Figure 1d.

<Fig. 1> Tabnet Architecture

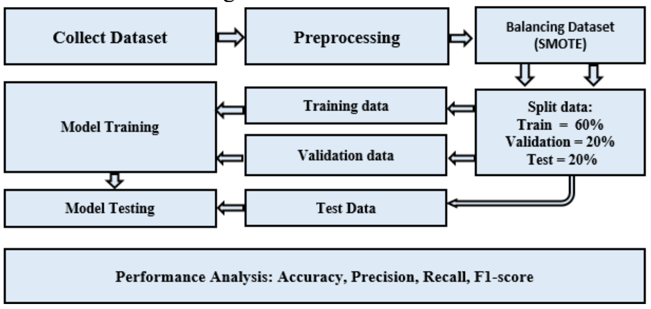


TabNet's architecture employs a sequential, attention-driven process to incrementally select and refine features, a process depicted in Figure 1. This design enables instance-wise feature selection, making it versatile across a wide range of tabular data scenarios. Moreover, the architecture's decision-making transparency stands out as a hallmark of its interpretability.

4. METHODOLOGY

In this study, we examined the effectiveness of TabNet in malware detection, a process that entails multiple steps including data collection and the identification of malicious software. Fig. 1 illustrates the system architecture of our proposed technique.

<Fig. 1> Tabnet Architecture


Collect Dataset
Preprocessing
Balancing Dataset (SMOTE)
Model Training
Training data
Validation data
Split data: Train = 60% Validation = 20% Test = 20%
Model Testing
Test Data
Performance Analysis: Accuracy, Precision, Recall, F1-score

We utilized TensorFlow, an open-source machine learning library developed by Google, to construct our deep learning models. TensorFlow's architecture is both versatile and efficient, facilitating the construction and training of neural networks. Utilizing its high-level Keras API allowed us to rapidly and effortlessly build our models, while TabNet calculations were implemented to expedite training. Our investigation primarily focused on assessing the TabNet model's effectiveness.

The Synthetic Minority Over-Sampling Technique (SMOTE) represents an innovative method for addressing unbalanced datasets, widely used in machine learning [8]. Unlike simple oversampling, SMOTE generates synthetic samples from the minority class to balance class distribution, thereby avoiding the repetition of minority class instances and mitigating overfitting risks. SMOTE fosters the creation of a more representative and varied dataset within the feature space for classifier training by interpolating new instances between existing minority samples. This approach enhances the model's ability to generalize by offering a deeper insight into the minority class's feature space.

4.1. Description of Dataset

The dataset used in this study consists of 138,047 PE (Portable Executable) header samples, which are provided by H. Rathore [5]. It includes 41,323 benign and 96,724 malware samples. The dataset is available on a GitHub repository [9]. The SMOTE was employed to address the class imbalance and enhance the representation of the minority benign class in subsequent models. Detailed statistics for the dataset are presented in Tab. 1.

<Tab. 1> Statistics of Dataset Before and After Balancing

Condition	**Sample Counts**		**Total**
	Benign	**Malicious**	
Imbalanced	41,323	96,724	138,047
Balanced	96,724	96,724	193,448

4.2. EXPERIMENTAL SETUP

The studies were carried out on a computer system equipped with an Intel64 Family 6 Model 165 Stepping 3 CPU and 16GB of RAM. No additional GPUs were used for computations. The models were implemented and analyzed in Python. Several major libraries were used in our research, including NumPy for numerical computing, TensorFlow and PyTorch for developing and training neural network models, and Scikit-Learn and Pandas for data manipulation and machine learning. This software stack provides a reliable and adaptable computer environment for our experiments. Tab. 2 provides detailed hardware and software specifications.

<Tab. 2> System Specifications

Component	Specification
CPU	Intel Core i9-10900K (3.70 GHz) 9900K (9th Gen)
GPU	NVIDIA GeForce RTX 2080 Ti
RAM	64GB - 3600 MHz
Language	Python
Software	NumPy, TensorFlow, Scikit-Learn, Pandas, PyTorch

4.1. Data Preprocessing

The datasets were subjected to crucial preprocessing steps before the training of the model to enhance data quality and focus:

- Check for missing value.
- Remove "Name" and "hash" columns.
- Normalization of features for training efficiency.

4.2. Data Split

The datasets were subjected to crucial preprocessing steps before the training of the model to enhance data quality and focus:

- 60\% for training
- 20\% for testing.
- 20\% for validation.

4.3. Training Scenarios Overview

Our study utilized the TabNetClassifier in two principal scenarios to assess its effectiveness in detecting malware across different class distributions. The key differentiator between these scenarios was the class balance, which was adjusted using the SMOTE.

Scenario 1: Imbalanced Dataset

Initially, the model underwent training using the original dataset, notably imbalanced with a predominance of malicious software instances. This setup was designed to evaluate the model's inherent capacity to manage class imbalances.

Scenario 2: Balanced Dataset via SMOTE

Subsequently, SMOTE was applied to create a balanced dataset, ensuring equal representation of malware and benign instances. This adjustment tested the hypothesis that a balanced class distribution would enhance detection sensitivity.

Common Model Configuration

For both scenarios:

- Epochs: Training was limited to 100 epochs, with early stopping based on validation performance.
- Batch Sizes: A batch size of 1024 and a virtual batch size of 128 were used.
- Optimizer: The Adam optimizer with a learning rate of 0.02 was employed.
- Activation Functions: Sparsemax was utilized for the attention mechanism, and ReLU for feature transformation, maintaining the model's interpretability and efficiency.

The complete design of the proposed methods is outlined in Tab. 3.

<Tab. 3> Comparison of Training Scenarios Before and After Applying SMOTE

Feature	**TabNetClassifier**	
	Before SMOTE	**After SMOTE**
Class Distribution	Imbalanced	Balanced
Sampling Technique	None	SMOTE
Max Epochs	100	100
Batch Size	1024	1024

Virtual Batch Size	128	128
Optimizer	Adam	Adam
Learning Rate	0.02	0.02
Attention Activation Func.	Sparsemax	
Feature Transf. Activation Func.	ReLU	

4.4. Evaluation Criteria

We use an extensive collection of evaluation criteria, each providing distinct insights into different facets of the behavior of our model, to gauge its performance:

- **Accuracy** is defined as the number of correct predictions made by the model, in contrast to all predictions ever made.
- **Precision** measures the proportion of correctly predicted positive values.
- **Recall** measures the percentage of actual positive values correctly predicted by the algorithm.
- **F1-Score** is the harmonic mean of precision and recall.

Each of these metrics contributes to a holistic understanding of the model's effectiveness, facilitating informed adjustments and improvements.

5. RESULTS AND DISCUSSION

The evaluation of the TabNetClassifier's efficacy in malware detection showed significant improvements following the application of SMOTE. Initially, the model achieved a high accuracy of 99.03%, despite being tested on an imbalanced dataset. The use of SMOTE to balance the dataset led to a marginal increase in accuracy to 99.10%. More notably, both precision and recall were enhanced, with precision reaching 99.03% and recall improving to 99.19%, reflecting a heightened sensitivity in detecting malware instances. These improvements collectively highlight the beneficial impact of SMOTE on the model's performance, particularly in effectively identifying the minority class. Comparison results are detailed in Tab. 4.

<Tab. 3> Model Performance

	Accuracy	Precision	Recall	F1-Score
Pre-SMOTE	**99.03%**	**98.24%**	**98.49%**	**98.37%**
Post-SMOTE	**99.10%**	**99.03%**	**99.19%**	**99.11\%**

6. RESULTS AND DISCUSSION

The findings of this study underscore the effectiveness of the TabNetClassifier in malware detection, particularly highlighting its interpretability and performance. Notably, the application of SMOTE has demonstrated a significant enhancement in precision, recall, and F1-score metrics, emphasizing the importance of balanced datasets in improving model training outcomes. Looking ahead, future research endeavors will explore alternative data balancing techniques and investigate the integration of TabNet with other models to further enhance cybersecurity defenses.

ACKNOWLEDGEMENT

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. NRF-2022R1A4A1032361).

REFERENCES

[1] Raghuraman, Chandni, et al. "Static and dynamic malware analysis using machine learning." *First International Conference on Sustainable Technologies for Computational Intelligence: Proceedings of ICTSCI 2019*. Springer Singapore, 2020.

[2] Aslan, Ömer, and Abdullah Asim Yilmaz. "A new malware classification framework based on deep learning algorithms." *Ieee Access* 9 (2021): 87936-87951.

[3] Olowoyo, Olufikayo, and Pius Owolawi. "Malware classification using deep learning technique." *2020 2nd International Multidisciplinary Information Technology and Engineering Conference (IMITEC)*. IEEE, 2020.

[4] Li, Chen, and Junjun Zheng. "API call-based malware classification using recurrent neural networks." Journal of Cyber Security and Mobility 10.3 (2021): 617-640.

[5] Rathore, Hemant, et al. "Malware detection using machine learning and deep learning." Big Data Analytics: 6th International Conference, BDA 2018, Warangal, India, December 18–21, 2018, Proceedings 6. Springer International Publishing, 2018.

[6] Catak, Ferhat Ozgur, et al. "Deep learning based Sequential model for malware analysis using Windows exe API Calls." PeerJ Computer Science 6 (2020): e285.

[7] Arik, Sercan O., and Tomas Pfister. "Tabnet: Attentive interpretable tabular learning. arXiv 2019." arXiv preprint arXiv:1908.07442 (1908).

[8] Soltanzadeh, Paria, and Mahdi Hashemzadeh. "RCSMOTE: Range-Controlled synthetic minority over-sampling technique for handling the class imbalance problem." Information Sciences 542 (2021): 92-111.

[9] https://github.com/PacktPublishing/Mastering-Machine-Learning-for-Penetration-Testing/tree/master/Chapter03.