Windows Malware Detection: Exploring from Machine Learning to Text-Based Deep Learning Approaches

Ly Ngoc Vu
Faculty of Information Technology
Industrial University of Ho Chi Minh City
Ho Chi Minh City, Vietnam
dezzhuge@gmail.com

Dang Thi Phuc
Faculty of Information Technology
Industrial University of Ho Chi Minh City
Ho Chi Minh City, Vietnam
phucdt@iuh.edu.vn
(Corresponding Author)

***Abstract*—In today's era, individuals and businesses face increasing security threats from malware, including ransomware, Trojans, and viruses. Traditional detection methods have become less effective as malware creators continually find ways to evade them. This paper proposes machine learning and deep learning models to classify malware. To enhance classification effectiveness for various types of malware, we have collected and combined numerous diverse and up-to-date datasets, totaling approximately 34,385 samples. We propose using machine learning models, including Logistic Regression, Support Vector Machines (SVC), Random Forest, and XGBoost, utilizing ensemble learning techniques such as voting and stacking to optimize the models. For deep learning models, we approach the problem in two ways: building LSTM and BiLSTM models and using the pre-trained DistilBERT model for classification, refining these models to achieve the best results. The results show that on large datasets, the models achieved high classification accuracy, with deep learning models like DistilBERT reaching up to 99% accuracy. These results indicate that our model is more effective compared to other methods on previous smaller and imbalanced datasets.**

***Index Terms*—malware detection, machine learning, deep learning, classification, ensemble learning, and transfer learning.**

I. Introduction

In today's interconnected world, digital technologies offer immense convenience and efficiency. However, they also bring escalating security threats from malware, posing significant risks to individuals and businesses alike. Traditional detection methods, like signature-based approaches and behavioral analysis, are becoming inadequate as malware creators constantly evolve their tactics [1].

Machine learning (ML) provides a powerful framework for malware detection by learning patterns from data and identifying subtle indicators of malicious behavior. By extracting features such as file attributes, behavior patterns, and code snippets, ML models can differentiate between benign and harmful files, even against novel threats [2].

A major challenge is newly emerging malware, which traditional methods fail to detect due to their unknown nature. Our research aims to enhance malware detection using ML by building models that generalize from labeled training data and recognize unseen threats. Ensemble techniques combine multiple models to improve performance, while deep learning (DL) offers advanced capabilities with neural networks [3].

We will explore DL architectures, including transfer learning and ensemble methods, to stay ahead of evolving threats. Our goal is to advance intelligent malware detection, contributing to a more secure cyber environment. Before diving into analysis and model development, we must define the scope of our detection efforts. Specifically, we focus on Portable Executable (PE) file formats. These files are executable on Windows—an operating system widely used across the globe. Unfortunately, this popularity also makes Windows a prime target for malicious hackers. As a result, bolstering our security expertise and capabilities becomes imperative.

II. Methodology

A. Proposed Machine Learning, Deep Learning, and Optimization Techniques for Malware Detection

In this paper, we propose machine learning, deep learning models, and optimization techniques to enhance the effectiveness of the malware detection problem described in Figure 1.

1) Proposed Machine Learning Model for Malware Detection: We chose Logistic Regression and Support Vector Machines (SVC) for their effectiveness in binary classification, particularly in distinguishing between benign and malicious files—critical for malware detection. Logistic Regression was selected due to its simplicity and reliability in producing probabilistic outputs, while SVC was preferred for its ability to create a hyperplane that maximally separates the two classes. Both models were optimized using GridSearchCV, with key hyperparameters like regularization strength (C) and kernel type fine-tuned to prevent overfitting.
Random Forest and XGBoost were implemented too. Random Forest was selected for its robustness to noisy data and its ability to handle the class imbalance present in the dataset. Its ensemble nature reduces variance by averaging multiple

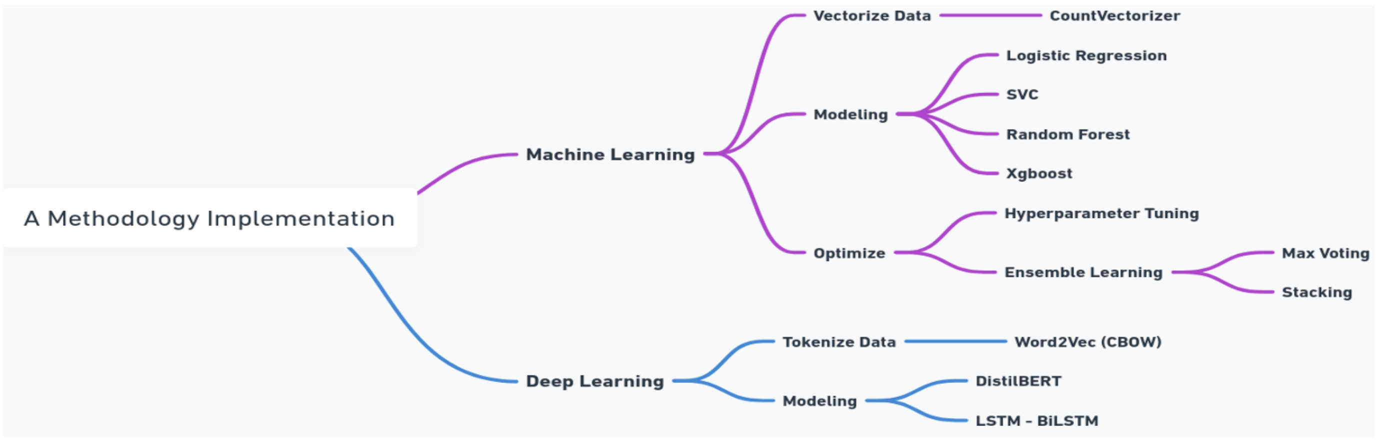


Fig. 1. A Methodology Implementation.

decision trees. XGBoost, on the other hand, was used for its efficiency and flexibility in gradient-boosting tasks. It handles large datasets effectively, and its hyperparameters, such as learning rate, max depth, and subsampling ratio, were fine-tuned to maximize accuracy and computational efficiency.

B. Ensemble Learning: Combining Models for Enhanced Predictions

Ensemble techniques like stacking combine the strengths of individual classifiers, allowing the model to capture different aspects of malware behavior, which leads to improved overall performance.

C. Proposed Deep Learning Model for Text-Based Malware Detection

Recurrent Neural Networks (RNNs) are widely used in natural language processing due to their ability to retain memory, essential for understanding sequential data. However, RNNs struggle with long sequences, which led to the development of Long Short-Term Memory (LSTM) networks.

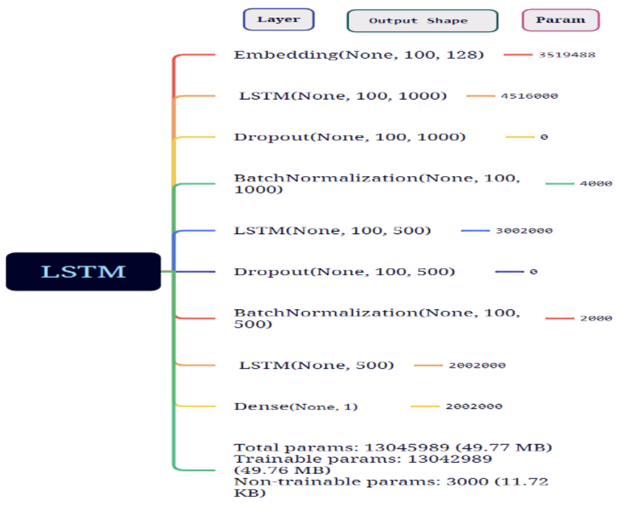
Long Short-Term Memory (LSTM) [8]: LSTM networks excel at retaining information across long sequences, enabling them to handle complex language structures and address the limitations of traditional RNNs in capturing long-term dependencies.

Bidirectional Long Short-Term Memory (BiLSTM) [9]: BiLSTM improves on LSTM by processing input bidirectionally, enhancing contextual understanding, particularly beneficial in tasks like sentiment analysis.

In our study, we implemented LSTM and BiLSTM architectures (Fig. 2 and 3). Pre-trained models helped mitigate BiLSTM's computational cost while improving real-time performance and interpretability.

D. Transfer Learning: Leveraging Pretrained Models for New Tasks

Transfer learning enables the use of pre-trained models for novel tasks, avoiding the need to build models from scratch.



Layer	Output Shape	Param
Embedding	(None, 100, 128)	3519488
LSTM	(None, 100, 1000)	4516000
Dropout	(None, 100, 1000)	0
BatchNormalization	(None, 100, 1000)	4000
LSTM	(None, 100, 500)	3002000
Dropout	(None, 100, 500)	0
BatchNormalization	(None, 100, 500)	2000
LSTM	(None, 500)	2002000
Dense	(None, 1)	2002000

Total params: 13045989 (49.77 MB)
Trainable params: 13042989 (49.76 MB)
Non-trainable params: 3000 (11.72 KB)

Fig. 2. LSTM Model Architecture.

In malware detection, we fine-tuned existing models with new data, saving both time and resources.

BERT, developed by Google AI [10], is a state-of-the-art language model known for its bidirectional contextual understanding, built on the Transformer framework. Key components of BERT include:

- **Self-Attention Layer**: Enables each word to consider all others simultaneously, providing comprehensive relationships and context.
- **Feed-Forward Network**: Adds non-linearity to help the model learn complex word relationships.

BERTBASE (12 layers, 768 hidden units, and 12 attention heads) is compact, while BERTLARGE (24 layers, 1024 hidden units, and 16 attention heads) offers greater capacity for more complex tasks.

DistilBERT for Malware Detection [11]: While BERT's architecture is effective, its scale can pose computational challenges. DistilBERT, a smaller, faster version of BERT,

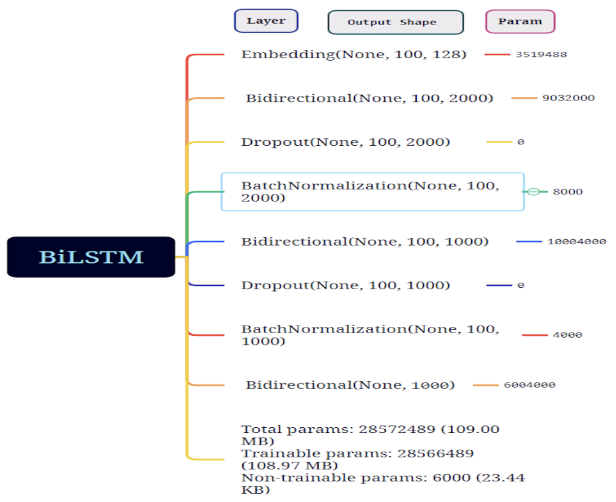

Layer
Output Shape
Param
BiLSTM
Embedding(None, 100, 128) 3519488
Bidirectional(None, 100, 2000) 9032000
Dropout(None, 100, 2000) 0
BatchNormalization(None, 100, 2000) 8000
Bidirectional(None, 100, 1000) 10004000
Dropout(None, 100, 1000) 0
BatchNormalization(None, 100, 1000) 4000
Bidirectional(None, 1000) 6004000
Total params: 28572489 (109.00 MB)
Trainable params: 28566489 (108.97 MB)
Non-trainable params: 6000 (23.44 KB)

Fig. 3. BiLSTM Model Architecture.

leverages knowledge distillation to reduce size while maintaining comparable performance in NLP tasks. It is ideal for resource-limited environments and real-time inference. We propose using DistilBERT (Fig. 4) for its efficiency in handling text-based malware detection tasks.

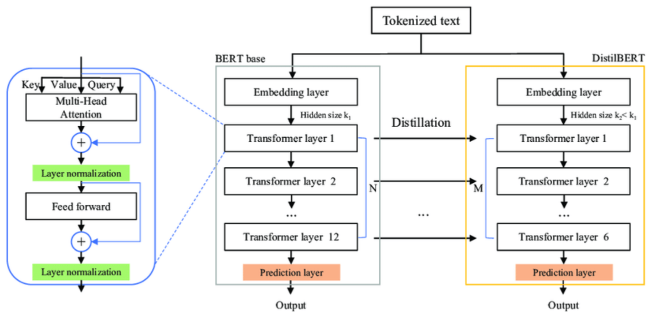


Fig. 4. DistilBERT Model Architecture [11].

E. Evaluating Model Performance

Accuracy [12]: The ratio of correct predictions to the total data used to evaluate the model.
Precision: The ratio of correct predictions to the total predictions made for a specific class. It indicates the confidence level of the model.
Recall: The ratio of correct predictions to the actual total for a specific class. It shows the completeness of the model in predicting all cases of a particular class.
F1-score: The harmonic mean of precision and recall, calculated using the formula:

$$\text{F1-score} = \frac{2\times(\text{precision}\times\text{recall})}{\text{precision}+\text{recall}}$$

F1-score reflects the overall accuracy of the model across all classes.

III. Dataset

A. Data Acquisition

We collected PE files from multiple sources:

- VX Underground
- SourceForge
- Softonic
- VirusShare

The dataset, containing 35,000 data points, was obtained via web scraping and parsed with the Pefile library, ensuring data accuracy through strict quality control. The initial dataset is shown in Table I.

B. Comparison with other datasets

In this study, we utilize a new dataset consisting of 34,385 samples, including 17,150 benign and 17,235 malware samples, which are balanced and derived from multiple sources (VX Underground, VirusShare, Softonic). To highlight the effectiveness of our dataset, we compare it with two well-known datasets in the field of malware detection, SOREL-20M and BODMAS, as well as other datasets.

- **Size**: described in Table II.
- **Balance**: Most datasets are not balanced in terms of the ratio between malware and benign samples. For example, large datasets like SOREL-20M and BODMAS contain more malware than benign samples. However, the dataset we collected is balanced (50% benign, 50% malware).
- **Data Types**: Depending on the purpose of malware classification, the types of data in the datasets vary. For example, the SOREL-20M dataset includes PE files, features from VirusTotal, and EMBER; BODMAS focuses on APTs and malicious campaigns, while our dataset focuses on PE files from VX Underground, VirusShare, and Softonic.
- **Purpose, Method and Strengths**: The data types are collected based on the goal of classifying different types of malware. For example, the SOREL-20M dataset is used for large-scale malware detection, focusing on scalability and advanced models. It is a large dataset with many features suitable for general malware detection. The BODMAS dataset is used to detect sophisticated malware campaigns and advanced threats, focusing on advanced attacks and real-world sophisticated malware samples. The Android Malware Dataset focuses on malware on the Android operating system. Our dataset focuses on PE files from Virus VX Underground and VirusShare, collected from diverse sources, and is suitable for use with ML and DL models.

In summary, our dataset provides a balanced and up-to-date representation of malware and benign samples, with a 50-50 split between 17,150 benign files and 17,235 malware files. Compared to larger datasets like SOREL-20M and BODMAS, which are designed for large-scale and complex deep learning applications, our dataset is specifically tailored for medium-scale systems where computational resources may be limited. This balance ensures more generalized model performance

TABLE I
DATASET STRUCTURE INFORMATION

Attribute	Description	Value Type	Example
File Size	Size of the file	Number	14868480
File Format	Format of the file	Number	332
Number of Sections	Number of sections in the file	Number	9
DOS Header Magic	DOS header magic number	Number	23117
DOS Header Lfanew	Offset to the PE header	Number	288
PE Header Signature	Signature of the PE header	Number	17744
PE Header Machine	Machine type for the file	Number	332
PE Header Number of Sections	Number of sections in the PE header	Number	3
Sections	Details of the file sections	Object	{'section_1_name':'.text','section_1_size'}...
Import Table	Details of imported symbols	Object	{'quazip.dll':['?extractFile@JlCompress@@SA?...']}
Export Table	Details of exported symbols	Object	NaN (Not available)
Type	Type of file	String	Malware/Benign

and reduces class bias, which is prevalent in other large-scale datasets suffering from class imbalance.

TABLE II
COMPARING DATASET CHARACTERISTICS

Dataset	Benign	Malware	Total Number of Data
Microsoft BIG2015 [13]	0	10,868	10,868
VirusShare [14]	0	17,235	17,235
Android Malware Dataset [15]	9,476	5,560	15,036
SOREL-20M [16]	8.6M	11.4M	20M
BODMAS [17]	57,293	77,142	134,437
Our Dataset	17,150	17,235	34,385

C. Analysis and Preprocessing Data

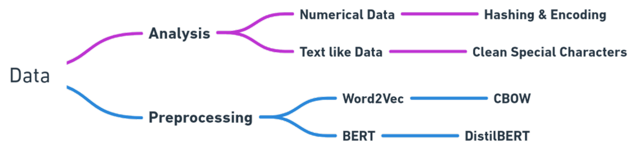


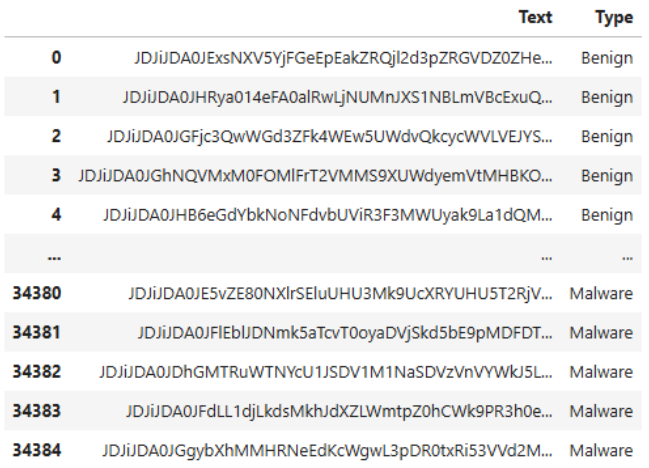
Fig. 5. Data Processing Workflow.

The data processing flow is depicted in Fig. 5 Data Flow Diagram. We will now proceed to examine each section in detail.

1) Hashing - Encoding: In our research, we use bcrypt hashing and base64 encoding to convert numerical values into string representations.

- Bcrypt is a secure hashing function that uses salts and an iterative process to increase computational cost, protecting against brute-force attacks.
- Base64 is a binary-to-text encoding that translates binary data into ASCII for transmission in non-8-bit clean channels, ensuring data integrity.

After analyzing the data, we have a dataset as shown in Fig. 6. This dataset includes all the models used in this study.



	Text	Type
0	JDJiJDA0JExsNXV5YjFGeEpEakZRQjl2d3pZRGVDZ0ZHe...	Benign
1	JDJiJDA0JHRya014eFA0alRwLjNUMnJXS1NBLmVBcExuQ...	Benign
2	JDJiJDA0JGFjc3QwWGd3ZFk4WEw5UWdvQkcycWVLVEJYS...	Benign
3	JDJiJDA0JGhNQVMxM0FOMlFrT2VMMS9XUWdyemVtMHBKO...	Benign
4	JDJiJDA0JHB6eGdYbkNoNFdvbUViR3F3MWUyak9La1dQM...	Benign
...	...	...
34380	JDJiJDA0JE5vZE80NXlrSEluUHU3Mk9UcXRYUHU5T2RjV...	Malware
34381	JDJiJDA0JFlEblJDNmk5aTcvT0oyaDVjSkd5bE9pMDFDT...	Malware
34382	JDJiJDA0JDhGMTRuWTNYcU1JSDV1M1NaSDVzVnVYWkJ5L...	Malware
34383	JDJiJDA0JFdLL1djLkdsMkhJdXZLWmtpZ0hCWk9PR3h0e...	Malware
34384	JDJiJDA0JGgybXhMMHRNeEdKcWgwL3pDR0txRi53VVd2M...	Malware

Fig. 6. Analyzed data.

2) Word2Vec: Word2Vec groups words with similar meanings using machine learning. It's crucial for natural language processing and effective in classifying malware. CBOW (faster training, better performance with common words) and Skip-Gram (captures infrequent words) are the main architectures. We've chosen CBOW for our experiments.

3) Preprocessing Data for DistilBERT model: When applying these models to tokenized data, several crucial steps are involved:
- Tokenization: This step involves splitting the text into manageable pieces.
- Vectorization: This step converts these tokens into numerical representations.
- Model Processing: The vectorized tokens are then analyzed by the model.
- Interpretation: This step involves understanding the model's results about the current task.
DistilBERT and other advanced models require tokenization to function effectively. By breaking down complex texts into more manageable parts, tokenization enables accurate lan-

TABLE III
PERFORMANCE OF BASELINE MACHINE LEARNING MODELS

	Precision	**Recall**	**F1-Score**	**Accuracy**
Logistic Regression	0.990086	0.990232	0.990111	0.990112
Random Forest	0.990000	0.990382	0.990402	0.990403
SVC	0.990000	0.988060	0.988075	0.988076
XGBoost	0.990000	0.991542	0.991566	0.991566

TABLE IV
PERFORMANCE OF HYPERPARAMETER TUNING MACHINE LEARNING MODELS

	Precision	**Recall**	**F1-Score**	**Accuracy**
Logistic Regression	0.989932	0.993119	0.989820	0.989821
Random Forest	0.988901	0.988780	0.988802	0.988803
SVC	0.989941	0.989796	0.989820	0.989821
XGBoost	0.991686	0.991541	0.991565	0.991566

guage analysis. Converting unstructured text into a structured format allows these models to reach their full potential, ensuring high-performance natural language processing applications.

IV. Experimental Results

A. Settings of Experiments

The classifiers were trained using features extracted from the dataset using Python's Scikit-learn and PyTorch Lightning. All experiments were conducted on a server running 64-bit Ubuntu with an E5-2699 CPU and a P40 GPU, equipped with 48GB of RAM.

B. Results

The accuracy results of the baseline ML models on the test set are described in Table III. The results indicate that the ML models perform effectively on large datasets and achieve high accuracy. Similarly, the results of the models fine-tuned using hyperparameter tuning techniques, described in Table IV also yield high accuracy and show little difference compared to the baseline ML models. The accuracy results with XG-Boost achieving the highest accuracy of 99.15%. Hyperparameter tuning through GridSearchCV brought slight improvements, showing the models were already optimized.

The accuracy results of the ensemble techniques, as shown in Table IV, reveal that training multiple models on the same dataset significantly boosts accuracy. Among these, the stacking model achieves the highest accuracy by effectively combining the strengths of individual models and minimizing their weaknesses, resulting in a more robust overall performance. Notably, the stacking model achieves an accuracy of 99.12%, showcasing the advantages of ensemble methods in enhancing predictive reliability.

We trained the LSTM and BiLSTM models for 10 epochs each, while for DistilBERT, we followed the authors' recommendation and limited training to 5 epochs (as training beyond 4 epochs is not advised). The training results of these models are described in Fig 7, 8, and 9. The results show LSTM convergence over 10 epochs, with minimal overfitting as training and validation accuracies stay closely aligned.

TABLE V
COMPARISON OF VOTING TECHNIQUES ACCURACY

Voting Technique	**Precision**	**Recall**	**F1-Score**	**Accuracy**
Baseline Max Voting	0.99113	0.991237	0.991106	0.991129
Hypertuning Max Voting Stacking	0.9902145	0.9900885	0.990111	0.990112
Stacking	0.990214	0.990088	0.990111	0.990112

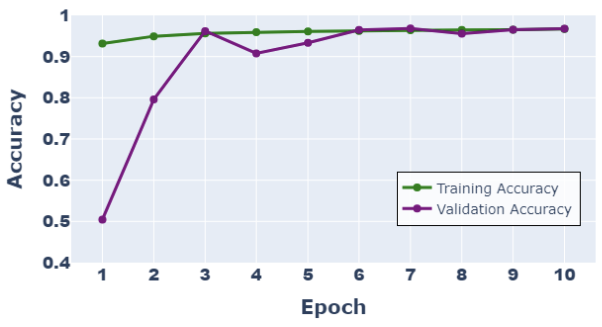


Fig. 7. Training vs Validation Accuracy for the LSTM Model.

The training results for the LSTM, BiLSTM, and DistilBERT models are presented in Tables VI. Among these deep learning models, DistilBERT achieved the highest accuracy at 99%, surpassing LSTM's 96.74% and BiLSTM's 95%. This indicates that the transfer learning approach used in DistilBERT is particularly effective for malware classification.

TABLE VI
COMPARISON OF DEEP LEARNING MODELS

Model	**Precision**	**Recall**	**F1-Score**	**Accuracy**
DistilBERT	0.9864895	0.986522	0.9864705	0.986471
LSTM	0.9674135	0.967449	0.967413	0.967413
BiLSTM	0.9507005	0.9499545	0.9500705	0.950102

Advanced optimization techniques like fine-tuning and GridSearchCV significantly enhanced model performance in this study. Although the initial models demonstrated high

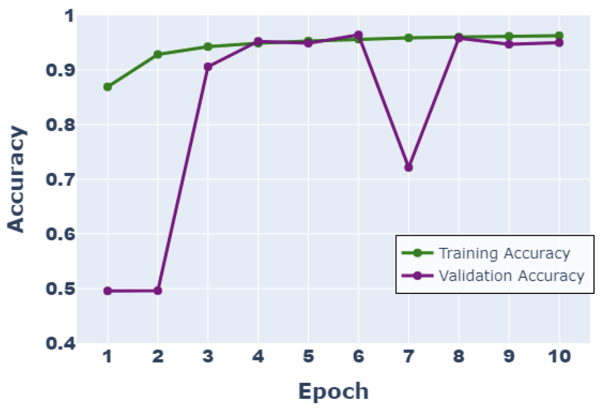


Fig. 8. The accuracy on the training set and validation set of the BiLSTM model.

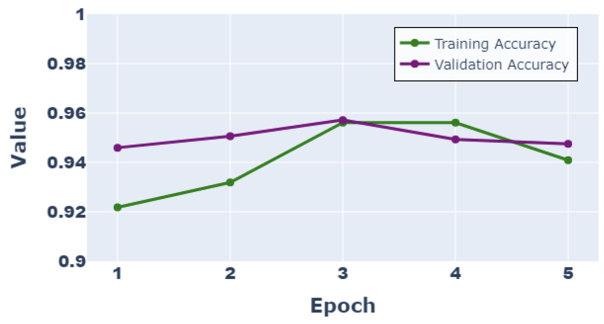
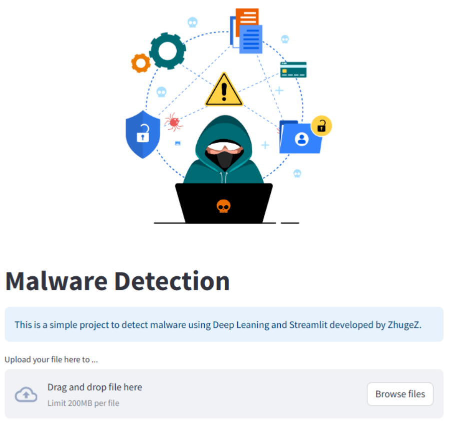


Fig. 9. The accuracy on the training set and validation set of the DistilBERT model.

accuracy, fine-tuning further improved DistilBERT's accuracy to an impressive 99%. Concurrently, GridSearchCV identified the optimal parameter configurations for the machine learning models, boosting accuracy while minimizing overfitting. Though these improvements may appear marginal, they are crucial in ensuring the model's stability and effectiveness when applied in real-world scenarios.

V. System Deployment


Malware Detection

This is a simple project to detect malware using Deep Leaning and Streamlit developed by ZhugeZ.

Upload your file here to ...

Fig. 10. Application UI.

Our malware detection solution, deployed as a web app using Streamlit(Fig. 10), offers an intuitive interface for real-time analysis of uploaded Windows PE files, classified by pre-trained models. This system provides proactive defense against cyber threats. Future enhancements will include real-time network monitoring, automated model updates, and improved file integrity checks to detect unauthorized changes. Automatic updates will ensure the system stays ahead of emerging threats.

VI. Conclusion

This research successfully demonstrated the effectiveness of machine learning and deep learning models, particularly DistilBERT, in detecting malware, achieving an accuracy of up to 99%. By incorporating transfer learning and optimizing hyperparameters, the models' performance and generalization were significantly improved, ensuring robustness against new malware threats.

A key contribution is the development of a balanced, up-to-date dataset, essential for enhancing model accuracy. Future work will focus on using advanced techniques like reinforcement learning and Generative Adversarial Networks (GANs) to create adaptive systems that can detect more sophisticated malware. Further improvements will involve refining the dataset and exploring unsupervised learning to discover new threats.

By continuously refining these detection models and integrating cutting-edge learning techniques, the goal is to build a more resilient and adaptive malware detection system capable of tackling the complexity of real-world cyber threats.

References

[1] D. Gibert, J. Planes, C. Mateu, and Q. Le, 'Fusing feature engineering and deep learning: A case study for malware classification', Expert Systems with Applications, vol. 207, p. 117957, 2022.

[2] D. Gibert, 'Machine Learning for Windows Malware Detection and Classification: Methods, Challenges and Ongoing Research', arXiv [cs.CR]. 2024.

[3] U. Divakarla, K. H. K. Reddy, and K. Chandrasekaran, 'A Novel Approach towards Windows Malware Detection System Using Deep Neural Networks', Procedia Computer Science, vol. 215, pp. 148–157, 2022.

[4] X. Zou, Y. Hu, Z. Tian and K. Shen, "Logistic Regression Model Optimization and Case Analysis," 2019 IEEE 7th International Conference on Computer Science and Network Technology (ICCSNT), Dalian, China, 2019, pp. 135-139, doi: 10.1109/ICCSNT47585.2019.8962457.

[5] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf, "Support vector machines," in IEEE Intelligent Systems and their Applications, vol. 13, no. 4, pp. 18-28, July-Aug. 1998, doi: 10.1109/5254.708428

[6] L. Breiman, 'Random Forests', Machine Learning, vol. 45, no. 1, pp. 5–32, Oct. 2001.

[7] T. Chen and C. Guestrin, 'XGBoost: A Scalable Tree Boosting System', CoRR, vol. abs/1603.02754, 2016

[8] R. C. Staudemeyer and E. R. Morris, 'Understanding LSTM - a tutorial into Long Short-Term Memory Recurrent Neural Networks', CoRR, vol. abs/1909.09586, 2019

[9] S. Zhang, D. Zheng, X. Hu, and M. Yang, 'Bidirectional Long Short-Term Memory Networks for Relation Classification', in Proceedings of the 29th Pacific Asia Conference on Language, Information and Computation, 2015, pp. 73–78.

[10] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, 'BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding', CoRR, vol. abs/1810.04805, 2018

[11] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, 'DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter', CoRR, vol. abs/1910.01108, 2019.

[12] H. Dalianis, 'Evaluation Metrics and Evaluation', 05 2018, pp. 45–53.

[13] M. Ahmadi, D. Ulyanov, S. Semenov, M. Trofimov, and G. Giacinto, 'Novel Feature Extraction, Selection and Fusion for Effective Malware Family Classification', in Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy, New Orleans, Louisiana, USA, 2016, pp. 183–194.

[14] D. A. Noever and S. E. M. Noever, 'Virus-MNIST: A Benchmark Malware Dataset', CoRR, vol. abs/2103.00602, 2021.

[15] S. Y. Yerima and S. Sezer, "DroidFusion: A Novel Multilevel Classifier Fusion Approach for Android Malware Detection," in IEEE Transactions on Cybernetics, vol. 49, no. 2, pp. 453-466, Feb. 2019, doi: 10.1109/TCYB.2017.2777960.

[16] R. Harang and E. M. Rudd, 'SOREL-20M: A Large Scale Benchmark Dataset for Malicious PE Detection', arXiv [cs.CR]. 2020.

[17] L. Yang, A. Ciptadi, I. Laziuk, A. Ahmadzadeh and G. Wang, "BODMAS: An Open Dataset for Learning based Temporal Analysis of PE Malware," 2021 IEEE Security and Privacy Workshops (SPW), San Francisco, CA, USA, 2021, pp. 78-84, doi: 10.1109/SPW53761.2021.00020.