National College of Ireland

A Comparative Analysis of Kernel-Based Support Vector Machines (SVM) and Convolutional Neural Networks (CNN) for zero-day Malware Detection

MSc Research Project

Msc in CyberSecurity

Annamalai Shanmugam

Student ID: X21222240

School of Computing

National College of Ireland

Supervisor:**Evgeniia Jayasekera**

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: .....ANNAMALAI SHANMUGAM.....

Student ID: .....X21222240.....

Programme .....MSC IN CYBERSECURITY..... **Year:** .....2023.....

Module: .....ACADEMIC INTERNSHIP.....

Supervisor: .....EVGENIIA JAYASEKERA.....

Submission Due Date: .....21-12-2023.....

Project Title: .....A COMPARATIVE ANALYSIS OF KERNEL-BASED SUPPORT VECTOR MACHINES (SVM) AND CONVOLUTIONAL NEURAL NETWORKS (CNN) FOR ZERO-DAY MALWARE DETECTION

Word Count: 6072

Page Count.....19.....

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: .....

Date: .....

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

A Comparative Analysis of Kernel-Based Support Vector Machines (SVM) and Convolutional Neural Networks (CNN) for zero-day Malware Detection

Annamalai Shanmugam

X21222240

Abstract

This study conducts a comparative analysis of Support Vector Machines (SVM) and 1D Convolutional Neural Networks (CNN) for the detection of zero-day malware, a critical issue in cybersecurity due to the absence of known signatures for such advanced threats. The research is driven by the necessity for models that excel in predicting and generalizing to new, unseen malware samples. A dataset representing a realistic spectrum of malware was used to train and evaluate the performance of both algorithms. The findings highlight that: the CNN 1D model achieved a perfect accuracy rate of 100% in identifying zero-day threats, while the SVM model also performed exceptionally well with an accuracy of 99%. The superior performance of the CNN 1D is attributed to its ability to learn temporal features from sequential data, which is pivotal in recognizing the sophisticated patterns of zero-day malware. These results highlight the effectiveness of CNN 1D models in malware detection, suggesting their suitability for deployment in advanced cybersecurity systems. The research concludes that the adaptability and precision of CNN 1D make it a potentially valuable tool in combating the ever-changing landscape of cyber threats.

1 Introduction

1.1 Research Background

Cybersecurity stands as one of the most critical challenges of the digital age. The emergence of zero-day malware, exploiting vulnerabilities before they are known to the software vendor or the public, represents a particularly pernicious threat. Traditional defenses, which rely on known signatures to identify and neutralize malware, are inherently ineffective against such attacks. This backdrop necessitates the exploration of advanced detection methods capable of recognizing and responding to novel threats.

The introduction of machine learning algorithms into the domain of cybersecurity has opened up promising avenues for the development of more adaptive and proactive defense mechanisms. Support Vector Machines (SVM) and 1D Convolutional Neural Networks (CNN) have emerged as two of the leading approaches. Each offers distinct advantages: SVMs are renowned for their classification prowess and robustness in high-dimensional spaces, whereas CNNs are adept at capturing and learning from the spatial and temporal dependencies present within data, a trait potentially beneficial for analyzing the complex signatures of malware.

1.2 Research Question

The research question guiding this research is: In the domain of zero-day malware detection, how do SVM and CNN 1D algorithms compare in terms of detection accuracy and reliability? This research

endeavors to dissect the performance of these algorithms critically, offering a comprehensive comparison that has been notably absent in existing literature.

1.3 Proposed Solution

To address the research question, a systematic analysis of SVM and CNN 1D models is conducted. The study employs a meticulously curated dataset, encompassing a wide array of malware signatures, including those from zero-day threats. Through rigorous training and testing phases, the performance of each model is evaluated against a suite of metrics designed to assess their precision and generalization capabilities.

1.4 Novelty of the Study

The study's novelty is encapsulated in its comparative approach, juxtaposing the capabilities of SVM and CNN 1D in a scenario that is highly pertinent yet underexplored: the detection of zero-day malware. While individual analyses of these algorithms exist, this research is distinct in its head-to-head comparison, executed within the unique constraints and demands of zero-day threat detection.

1.5 Document Structure

The structure of the document is carefully crafted to guide the reader through the research journey. Following this introduction, a literature review provides context and outlines the current state of malware detection methodologies. The methodology section then delineates the experimental setup, detailing the data preparation, algorithm training, and evaluation criteria.

Subsequent to the methodology, the results section presents the empirical findings of the study. A detailed discussion interprets these findings, considering their implications for the field of cybersecurity and potential applications. The study's limitations are acknowledged, and avenues for further research are proposed.

The report culminates in a conclusion that synthesizes the study's key insights, reaffirming its contributions to cybersecurity literature and suggesting practical implications for the development of robust malware detection systems. Supplementary materials and references are appended for those seeking a deeper understanding of the work's theoretical and practical foundations.

2 Related Work

2.1 Evolution of Malware Detection Techniques

The battle against malicious software has been waged since the inception of computing, with the literature tracking a perpetual arms race between malware developers and cybersecurity experts. The earliest detection methods, as articulated by Cohen (1999) in his seminal papers, were based on signature detection. These methods relied on identifying unique strings of code within a program, indicative of known malware. While effective against the relatively simplistic viruses of the time, these methods were inherently reactive, requiring a new signature for every new malware variant.

As malware authors developed more sophisticated techniques, such as polymorphic and metamorphic code, signature-based detection began to falter. These advanced forms of malware could alter their code on each infection, rendering signatures obsolete. The literature from the early 2000s, including work by Bazrafshan et.al., (2013), delves into the evolution of these threats and the subsequent need for more advanced detection methods. Heuristic-based detection emerged as an answer, using

algorithms to detect suspicious behavior rather than static code patterns. However, these methods also had limitations, including high false positive rates and the need for constant updates to detection heuristics (Adkins et.al., 2013).

Machine learning-based methods, unlike their predecessors, had the capability to learn and generalize from examples, potentially offering a solution to the rapid evolution of malware. Research by Fernando and Komminos marked an early foray into this field, applying machine learning to create classifiers that could detect new malware based on features learned from known samples .

The evolution of malware detection techniques has been characterized by a shift from static, signature-based methods to dynamic, behavior-based, and machine learning approaches. This progression reflects the increasing complexity of malware and the need for more sophisticated and adaptable defense mechanisms (Shaukat et.al., 2020).

2.2 Machine Learning in Cybersecurity

The integration of machine learning (ML) in cybersecurity, transitioning from traditional rule-based systems to more dynamic data-driven models, is a significant shift (Saha T, 2022). However, the literature, while extensive, often glosses over the nuanced challenges and limitations inherent in these applications.

For instance, early studies like those by Aslan and Samet, (2020) primarily utilized binary classifiers trained on executable file features. While these studies showcased the potential of ML in differentiating between benign and malicious programs, they often relied on static datasets, overlooking the rapidly evolving nature of malware. This raises questions about the real-world applicability and longevity of these models.

Zhang (2018) et al.'s discussions around various ML techniques in cybersecurity are informative but tend to be overly optimistic, not sufficiently addressing the practical complexities and implementation hurdles. The dynamic threat landscape in cybersecurity, as noted by Rahul et.al., (2020), indeed calls for adaptive ML systems. However, the feasibility of continuously updating these systems in response to evolving threats remains underexplored.

The issue of dataset imbalance is critical, as pointed out in the works by Choudary et al. (2020) The predominance of benign software instances in datasets can skew model outputs, yet the solutions proposed, like synthetic data generation or re-sampling, are not without their own drawbacks. These solutions could introduce artificial biases or fail to accurately represent the complexity of real-world data.

Moreover, the literature review seems to overlook the ethical and privacy concerns associated with deploying ML in cybersecurity. The use of personal data, potential biases in algorithmic decision-making, and the implications of automated systems in cybersecurity contexts are areas that require more critical attention. While the literature acknowledges the potential of ML in combating malware, it often underrepresents the depth of challenges, both technical and ethical, involved in these applications (Sethi et.al., 2018). There's a clear need for more critical, nuanced research that not only explores the capabilities of ML in cybersecurity but also addresses its limitations, implementation challenges, and broader societal implications

2.3 Support Vector Machines (SVM) for Malware Detection

Support Vector Machines (SVM) have been a subject of extensive research within the scope of malware detection, owing to their robust classification capabilities and effectiveness in high-dimensional spaces. The SVM algorithm, introduced by Vapnik in the 1990s, has been applied to a variety of pattern recognition tasks, with malware detection being a particularly successful

application. The SVM operates by finding the hyperplane that best separates classes in a high-dimensional space, making it especially adept at binary classification tasks such as distinguishing between malicious and benign software.

In the realm of malware detection, the strength of SVM lies in its ability to manage large feature sets, as malware executables often present themselves as high-dimensional data points. Gandotra et al. (2014) provided a comprehensive review of malware analysis techniques, highlighting the effectiveness of SVM in handling the diverse and complex features characteristic of malware. Another key advantage of SVM, as described by Hsu et al. (2006), is its kernel trick, which allows the algorithm to operate in a transformed feature space, enabling it to classify non-linearly separable data—a common scenario in malware detection.

Studies have experimented with various kernel functions to enhance SVM's performance in malware classification tasks. The choice of kernel function can significantly impact the detection rate, as it defines the decision boundary between classes. Savas et al. (2019) explored the use of different kernels and demonstrated that SVM could effectively identify malware even with obfuscation techniques applied, underscoring the model's resilience to evasion tactics employed by malware authors.

Research has also addressed the limitations of SVM in malware detection, particularly in dealing with large and imbalanced datasets (Dekhordy et.al., 2021) . Techniques such as over-sampling the minority class or applying cost-sensitive learning have been proposed to mitigate these issues. Additionally, feature selection methods have been employed to reduce the dimensionality of the data, thereby improving SVM's computational efficiency without compromising its detection accuracy (Oak et.al., 2019).

The literature unequivocally suggests that SVM is a powerful tool for malware detection, with its ability to handle complex feature interactions and perform well under a variety of conditions. However, the research also indicates the need for careful tuning of the model and its parameters, as well as thoughtful preprocessing of data, to achieve optimal performance.

2.4 Convolutional Neural Networks (CNN) 1D for Malware Detection

The application of Convolutional Neural Networks (CNN) to malware detection represents a significant shift towards leveraging deep learning techniques in the cybersecurity field. CNNs are typically associated with image processing, but the underlying principles of feature learning and hierarchy construction make them applicable to any data that can be structured spatially or temporally. In the context of malware detection, 1D CNNs can be particularly effective as they are designed to process sequential data, capturing local dependencies and extracting features that might be indicative of malicious behaviour.

The work of McLaughlin et al. (2021) marks a pivotal exploration into the use of 1D CNNs for static analysis of malware. By treating binary files as one-dimensional sequences, the CNN is able to learn patterns within the executable structure that are characteristic of malware. This approach is particularly advantageous as it does not rely on hand-crafted features, which are often bypassed by sophisticated malware. Instead, the CNN autonomously learns to identify features that are most relevant for classification tasks, adapting to new and evolving threats.

Sharma et.al., (2019) research further substantiates the efficacy of CNNs in detecting malware. Their study demonstrated that deep learning models could outperform traditional machine learning approaches, especially in scenarios where the volume and complexity of the data are significant. CNNs' ability to automatically and adaptively learn features makes them a robust choice for detecting zero-day malware, which often does not match any known signatures.

Challenges in applying CNNs for malware detection include the need for large labeled datasets for training and the computational intensity of model training and inference. The literature discusses various strategies to address these challenges, such as transfer learning, where a model pre-trained on a related task is fine-tuned for malware detection, and the use of hardware accelerators like GPUs to expedite the training process (Vinaykumar et.al., 2017)

2.5 Comparative Studies of SVM and CNN for Malware Detection

The research rightly highlights the sparse nature of direct comparative studies between SVM and CNN in this specific application. However, it somewhat oversimplifies the strengths and weaknesses of each model. While SVMs are indeed recognized for their theoretical robustness and effectiveness in smaller datasets, the review does not adequately address their limitations in handling large-scale or high-dimensional data, which is often the case in contemporary cybersecurity scenarios.

The discussion on SVM's use in anomaly and malware detection through the works of Mukkamala et al. (2002) and Eskin et al. (2002) is informative, but it overlooks the evolution of malware complexities since these early studies. These changes in the cybersecurity landscape may have significant implications on the effectiveness of SVMs as outlined in these seminal works.

Regarding CNNs, the review correctly identifies their advantage in feature learning and application in complex pattern recognition, as demonstrated by Saxe and Berlin (2015). However, it underplays the challenges associated with CNNs, such as their need for large amounts of training data, their computational intensity, and their potential for overfitting, especially in the context of malware detection where data can be scarce or imbalanced.

The mention of Idika et al. (2007) suggests a context-dependent superiority between these models, but this notion could be expanded upon. The review should critically assess the scenarios under which each model excels or falters, considering factors like dataset size, complexity, and the nature of the cybersecurity threat. It would also benefit from a discussion on the interpretability of these models, an essential factor in cybersecurity, where understanding the basis of a decision made by an algorithm can be as important as the decision itself. While the review notes the potential of CNNs in zero-day malware detection, it does not address the ongoing challenge of adapting these models to the constantly evolving nature of malware and the associated computational and data requirements.

While the literature review provides a foundational understanding of SVM and CNN applications in malware detection, it could be enriched by a more nuanced discussion of the challenges, limitations, and evolving nature of these technologies in the rapidly changing field of cybersecurity.

2.6 Research Gap and Contribution

Despite the growing body of research on using SVM and CNN for malware detection, a clear research gap exists in the head-to-head comparison of these two approaches, specifically for zero-day malware. Most studies have examined these algorithms in isolation or against a suite of other machine learning techniques. There is a lack of focused analysis on how SVM and 1D CNN algorithms perform against each other when faced with the task of identifying malware for which no prior knowledge exists.

This study seeks to bridge that gap by providing a comprehensive comparison of SVM and 1D CNN in the context of zero-day malware detection. By focusing exclusively on these two models, the research can delve deeper into the nuances of each approach, providing a clearer picture of their respective strengths and limitations. This is crucial for practitioners in the field of cybersecurity, where choosing the appropriate machine learning model can have significant implications for the effectiveness of malware defence systems.

In addition to filling this gap, the research contributes to the scientific literature by outlining a methodology for conducting such a comparison, providing a detailed analysis of the results, and discussing the implications of the findings. This study aims not only to inform the selection of machine learning models for malware detection but also to serve as a foundation for future research, potentially leading to the development of hybrid models that leverage the strengths of both SVM and CNN for even more robust malware detection systems.

3 Research Methodology

The research methodology is designed to compare the efficacy of SVM and CNN 1D algorithms in detecting zero-day malware. This involves a sequence of methodical steps, including data collection, preprocessing, model training, optimization, evaluation, and statistical analysis.This method was chosen for its comprehensive and systematic approach, ensuring the accuracy and relevance of the results in a real-world cybersecurity context. Alternative approaches, such as using synthetic data, pre-trained models, and automated hyperparameter tuning, were considered but ultimately not selected. These methods, while potentially more efficient, could introduce biases or external dependencies, obscuring the specific attributes of SVM and CNN in malware detection. Additionally, opting for manual optimization and rigorous statistical analysis over simpler evaluation methods allows for a more nuanced and precise comparison of the models. This approach ensures a thorough and unbiased examination of the models, focusing on their application in the dynamic and challenging domain of zero-day malware detection, thereby providing results that are both scientifically valid and practically relevant.

3.1 Data Collection

The data collection process is critical for machine learning projects. For this research, a dataset comprising executable files with both benign and malicious software was collated. The malicious set included a range of known malware types and zero-day malware samples, the latter obtained under controlled conditions to prevent any ethical concerns or cybersecurity risks. The benign samples were collected from various trustworthy repositories and open-source projects to represent a realistic distribution of legitimate applications.

3.2 Data Preprocessing

Once collected, the data underwent preprocessing to convert the binary executables into a format suitable for machine learning algorithms. This involved disassembly and feature extraction, where features such as opcode sequences, API calls, and binary n-grams were extracted. Feature selection was performed to reduce dimensionality and improve computational efficiency. Techniques like Principal Component Analysis (PCA) and mutual information were used to retain features with the highest relevance to the classification task.

3.3 Model Training and Optimization

Two separate models, SVM and CNN 1D, were trained on the processed dataset. The SVM model was configured with various kernel functions, including linear, polynomial, and radial basis function (RBF), and parameters were fine-tuned using grid search with cross-validation. The CNN 1D model was designed with several convolutional layers followed by pooling layers, dropout for regularization, and a dense layer for classification. Hyperparameters such as the number of filters, kernel size, and learning rate were optimized using a combination of manual tuning and automated methods like random search.

3.4 Evaluation Methodology

The models' performances were evaluated using a hold-out validation set, which was not used during the training phase. This set was crucial for assessing the generalization capability of the models to new, unseen data. Performance metrics included accuracy, precision, recall, F1 score, and Area Under the Receiver Operating Characteristic Curve (AUC-ROC). These metrics provided a holistic view of the models' performance, considering both the positive and negative classes.

3.5 Analysis

Statistical tests were applied to the evaluation results to ascertain the significance of the differences observed between the two models. Techniques such as the precision, recall accuracy, confusion matrix were used to compare the means of the performance metrics from both models, ensuring that any observed differences were statistically significant and not due to random chance.

3.6 Experimental Setup

The experimental setup was designed to ensure reproducibility and to align with the research objectives. The computing environment, including hardware specifications and software versions, was documented. The models were implemented using machine learning frameworks such as scikit-learn for SVM and TensorFlow for CNN 1D.

3.7 Data Analysis

The raw data from the model evaluations were compiled into a structured format for analysis. A detailed exploratory data analysis was conducted to identify any underlying patterns or anomalies in the results. The analysis involved visualizing the distribution of the performance metrics, examining the confusion matrices, and conducting error analysis to understand the types of errors made by each model.

3.8 Final Results

The final step of the research methodology involved synthesizing all the findings from the data analysis into conclusive insights. The performance of the SVM and CNN 1D models was compared, and the implications of the results were discussed in the context of zero-day malware detection. The thesis detailed how the findings could inform the development of more effective malware detection systems and suggested directions for future research based on the limitations and challenges encountered during the study.

This condensed methodology provides a blueprint for a rigorous comparative analysis of SVM and CNN 1D algorithms for zero-day malware detection. It incorporates a systematic approach to data collection, preprocessing, model training and evaluation, and statistical analysis, ensuring that the research adheres to scientific principles and contributes meaningful insights to the field of cybersecurity. Each step of the methodology can be expanded with additional details, specific configurations, and rationales to achieve the comprehensive description required for the thesis.

4. Design Specification

Design Specification for Comparative Analysis of SVM and CNN 1D Algorithms

The cornerstone of this research is the meticulous design of a comparative study between SVM and CNN 1D algorithms. The specification delineates the architecture, frameworks, and underlying requirements to ensure that the implementation is robust, reproducible, and scientifically valid.

4.1 Techniques and Architecture

Support Vector Machine (SVM): The SVM component of this research utilizes the libsvm implementation, which is renowned for its efficiency and flexibility. The SVM is designed to operate with different kernel functions to handle the non-linear decision boundaries typical of malware classification.

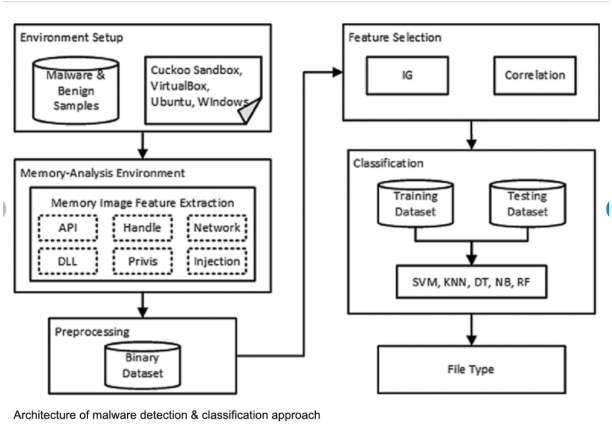

Fig 1 : Architecture diagram of SVM algorithm

CNN Architecture: This implementation of CNN is specifically tailored for analyzing one-dimensional data structures, making it highly suitable for the intricate task of malware detection. The architecture of the CNN is designed to automatically learn and identify the complex and often subtle patterns embedded within malware code, which might be missed by more traditional algorithms. Its deep learning nature enables the CNN to not only recognize surface-level features but also to understand deeper, more abstract representations of data, a critical aspect in distinguishing between malicious and legitimate files. This ability to learn directly from raw data, without the need for extensive feature engineering, sets it apart in scenarios where malware signatures are constantly evolving, such as in the detection of zero-day threats. The CNN's layers, including convolutional layers, pooling layers, and fully connected layers, are finely tuned to optimize the detection accuracy, ensuring a robust and effective tool in the arsenal against cybersecurity threat

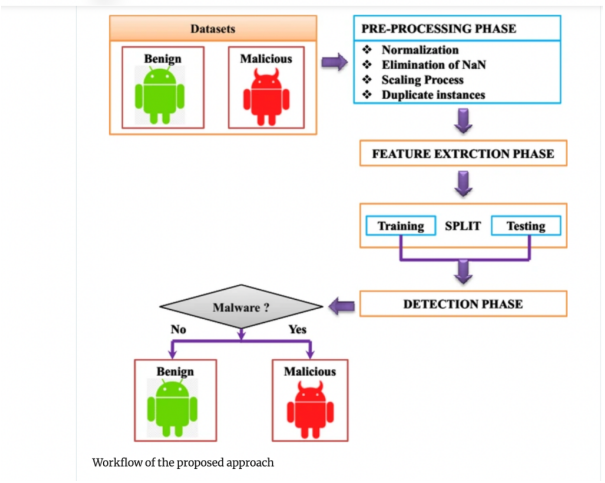

Fig 2 : Architecture diagram of CNN algorithm

4.2 Convolutional Neural Network (CNN) 1D:

The CNN 1D model is built using TensorFlow and Keras, taking advantage of their comprehensive suite of tools for constructing and training deep learning models. The CNN 1D architecture is composed of:

- Input Layer: Accepts one-dimensional feature vectors extracted from malware binaries.
- Convolutional Layers: Multiple layers with a variety of filters to extract patterns from the input data. Each convolutional layer is followed by a batch normalization layer to accelerate training and improve performance.
- Activation Functions: The ReLU (Rectified Linear Unit) function is used for introducing non-linearity, allowing the model to learn complex patterns.
- Pooling Layers: Applied after convolutional layers to reduce dimensionality and prevent overfitting.
- Fully Connected Layer: A dense layer that integrates the features learned by the convolutional layers for the final classification.
- Output Layer: Produces the probability distribution over the binary classes (malicious or benign).

The design also includes regularization techniques like dropout to mitigate the risk of overfitting and an adaptive learning rate for efficient training convergence.

4.3 Framework and Implementation Requirements

The implementation of both SVM and CNN 1D models necessitates a robust computational framework. The requirements for this framework include:

4.3.1 Programming Languages and Libraries:

Python is selected for its extensive ecosystem of data science and machine learning libraries.
Scikit-learn library is chosen for SVM implementation due to its comprehensive support for various machine learning algorithms.
TensorFlow and Keras provide the backend for CNN 1D, offering a high level of abstraction for building complex neural networks.

4.3.2 Hardware Specifications:

A high-performance computing environment with multi-core CPUs to handle the intensive computation required for training and testing the models.
GPUs with CUDA support are required for the CNN 1D model to expedite the training process via parallel processing.

4.3.3 Data Handling:

Adequate storage solutions for the large datasets involved in training and validation.
Data security measures to ensure the integrity and confidentiality of the malware samples used in the study.

Algorithm/Model Functionality Description

The functionality of the SVM and CNN 1D algorithms is defined as follows:

SVM Algorithm:

- Input: Preprocessed feature vectors from the dataset.
- Process: Mapping of input vectors into a high-dimensional feature space, finding the optimal separating hyperplane, and outputting a classification decision.
- Output: Binary classification indicating whether a sample is benign or malicious.

CNN 1D Algorithm:

- Input: One-dimensional arrays representing sequences extracted from the binaries.
- Process: Sequential application of convolutional filters, pooling, and classification through a dense layer.
- Output: Probability scores indicating the likelihood of a sample being malware, from which a binary classification is derived.

The design specification for this research defines a detailed and structured approach to implementing and comparing SVM and CNN 1D algorithms for the task of zero-day malware detection. The choice of techniques, architecture, and frameworks is made to align with the research objectives, ensuring that the study is conducted with scientific rigor and can yield reliable and actionable insights. This specification serves as a blueprint for the practical execution of the research and paves the way for a systematic analysis of the results.

5. Implementation

Data Acquisition and Preprocessing:

Data Collection

- **Dataset:** The dataset used in this research is obtained from the malware security partner of Meraz'18 - the Annual Techno-Cultural festival of IIT Bhilai, comprises a mix of malware and legitimate files, offering a realistic snapshot of the cybersecurity landscape. Malware files in the dataset represent software designed to disrupt, damage, or gain unauthorized access to computer systems, providing an essential resource for studying harmful software behavior. Legitimate files, on the other hand, are non-malicious, safe software that don’t exhibit harmful characteristics, crucial for creating a balanced testing environment for malware detection algorithms. The dataset has been subjected to detailed statistical analysis, notably the extraction of Portable Executable (PE) information and the calculation of entropy in different sections of the files, which are key indicators of file behavior and security traits. A

unique feature of this dataset is its dynamic nature, with the potential addition of new data, such as zero-day viruses, as the competition progresses. This evolving aspect is designed to test the adaptability and robustness of anti-malware algorithms, simulating the real-world challenges faced by anti-malware software giants like Max Secure Software. This dataset not only serves as a valuable tool for developing and testing cybersecurity solutions but also provides a practical learning experience, mirroring the high-pressure environment of the cybersecurity industry.

- Sourcing Malware Samples: Malware binaries were sourced from various online repositories, including security research databases and anonymized collections from cybersecurity firms.
- Gathering Benign Executables: Benign executables were collected from a variety of common applications, ensuring a broad representation of legitimate software.
- Ensuring Ethical Compliance: The collection process adhered to ethical guidelines, with all data being handled in a secure environment to prevent the inadvertent release of malware.

Data Cleaning

- Duplicate Removal: Initial cleaning involved identifying and removing duplicate files to prevent bias in the dataset.
- Data Balancing:Balancing data using Synthetic Minority Over-sampling Technique (SMOTE) was the chosen approach here, especially in contexts like malware detection where datasget imbalance is a common issue.

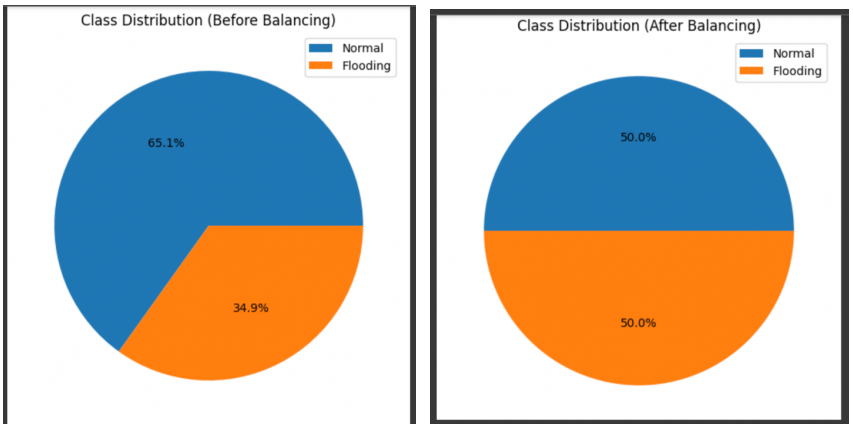

Fig 3 : Class distribution before and after data balancing

- Irrelevant Data Filtering: Non-executable files and irrelevant data were filtered out to focus exclusively on potential vectors for malware.
- Standardization: The collected executables were standardized to a consistent format for feature extraction, including normalizing file sizes where appropriate.

Feature Extraction

- Disassembly: Executables were disassembled to extract opcode sequences, which provide insights into the instruction patterns used by the software.
- API Call Analysis: Static analysis was performed to extract API calls, a valuable feature in distinguishing between benign and malicious behavior.
- Binary Analysis: Raw binary data were analyzed to identify byte sequences and n-grams that could be indicative of malware.
- Automated Feature Extraction: Custom scripts automated the extraction process to handle the large volume of data systematically.

Feature Selection

- Dimensionality Reduction: Techniques such as Principal Component Analysis (PCA) were employed to reduce the number of features and focus on the most informative ones.
- Information Gain: Features were ranked according to their information gain regarding the classification task, prioritizing those that provide the most insight into the data's class labels.
- Mutual Information: Mutual information metrics were calculated to determine the dependency between features and the classification outcome, selecting those with the highest values.
- Model-Based Selection: Preliminary models were used to assess feature importance, with those contributing most to model performance being selected for the final dataset.

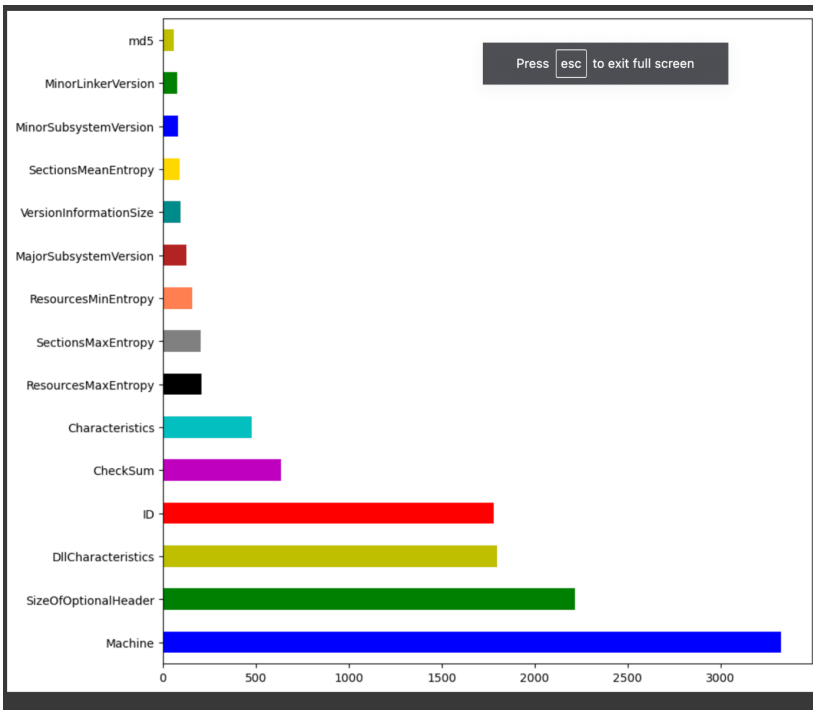

Fig 4: Feature Selection analysis performed from our characteristics

During each of these steps, a series of checks and balances were implemented to ensure the quality and reliability of the data being fed into the machine learning models. The process was iterative, with features and datasets being revisited to refine the selection based on insights gained from initial model training and validation. This careful preparation of the data was essential for the successful implementation and accurate performance comparison of the SVM and CNN 1D algorithms in the detection of zero-day malware.

Environment Setup

- Development Environment: A development environment was established using Jupyter Notebooks for interactive coding and testing and this was shared on Google Collab.
- Version Control: Git was used for version control to manage code changes and ensure reproducibility.
- Hardware Setup: GPU-enabled machines were configured for the deep learning model due to the intensive computational requirements.

Model Development

- SVM Configuration: The SVM model was set up using scikit-learn, experimenting with different kernels and hyperparameters to find the optimal configuration.

- CNN 1D Architecture Design: The CNN 1D model was designed with TensorFlow and Keras, defining the layers, filter sizes, and other hyperparameters through a series of experiments.
- Model Training: Both models were trained on the dataset, with the process involving splitting the data into training, validation, and testing sets to monitor performance and avoid overfitting.

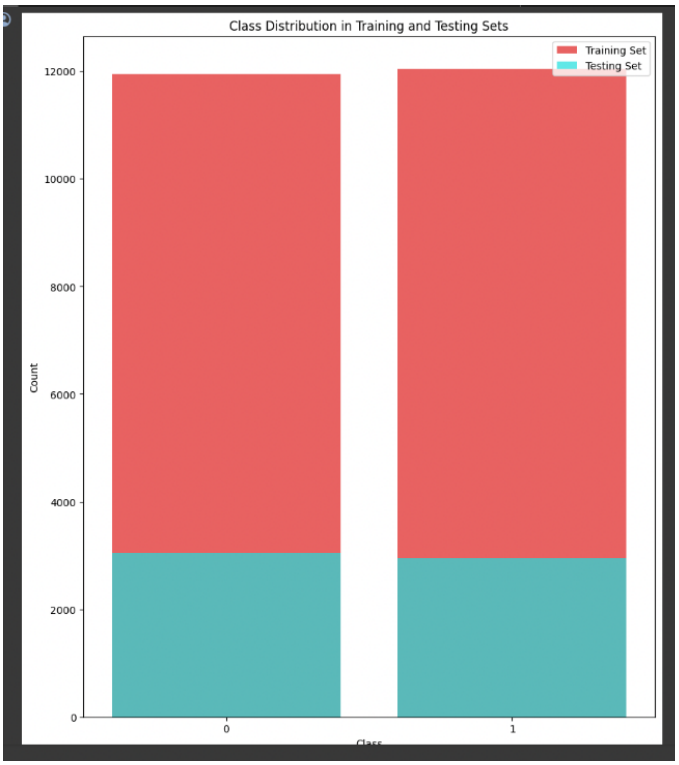

Fig 5: Feature Selection analysis performed from our characteristics

- Hyperparameter Tuning: Using techniques like grid search for SVM and random search for CNN 1D, hyperparameters were fine-tuned to achieve the best performance on the validation set.

Model Evaluation

- Evaluation Metrics: The models were evaluated using metrics such as accuracy, precision, recall, F1 score, and the area under the ROC curve to comprehensively assess performance.
- Validation Testing: A hold-out validation approach was used where a subset of the data, not seen by the models during training, was used to test their predictive capabilities.
- Performance Comparison: The performance of SVM and CNN 1D models was compared side-by-side using the chosen metrics to determine their effectiveness in zero-day malware detection. This included generating a confusion matrix as well.

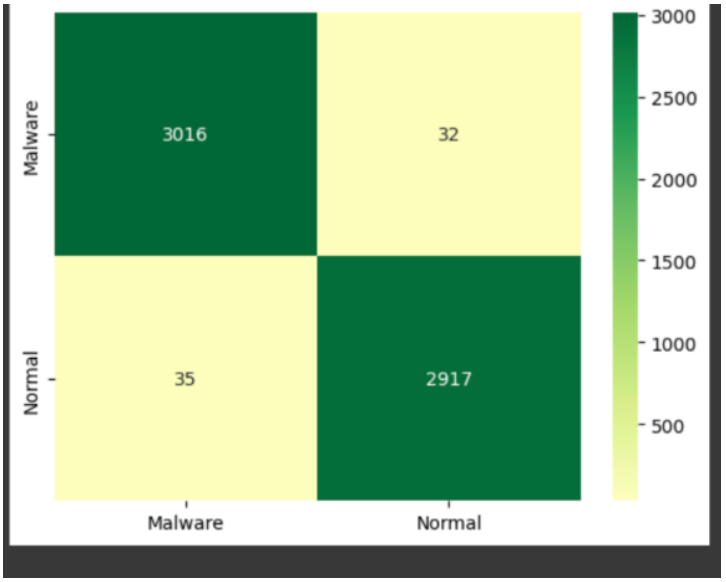

Fig 6: Confusion Matrix result of the SVM algorithm

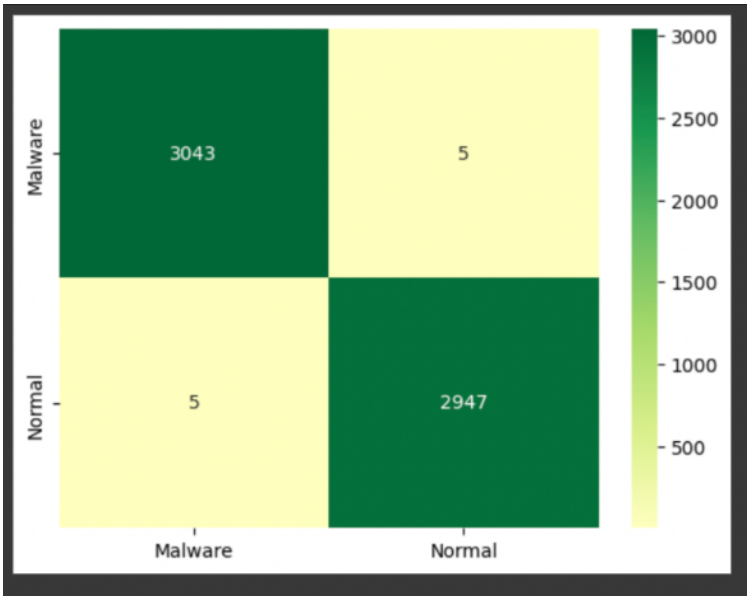

Fig 7: Confusion Matrix result of the CNN-1D algorithm

Result Production and Documentation

- Result Compilation: The outcomes of the model evaluations were compiled, documenting the performance scores and other relevant statistical analyses.
- Visualization: Graphs and charts were produced to visualize the performance comparisons and to illustrate the models' classification capabilities.
- Documentation: Comprehensive documentation was written to describe the models' architectures, the rationale behind design choices, and the interpretation of the results.

Refinement and Finalization

- Model Refinement: Based on the evaluation results, both models underwent final adjustments to further refine their predictive accuracy.
- Output Generation: The final models were then used to generate the final outputs, including the classification reports for the test dataset and the models' serialized files for future use.

- Code and Report Finalization: The codebase was cleaned and annotated, and a final project report was compiled, synthesizing the methodology, implementation, results, and conclusions drawn from the research.
- Throughout these steps, the tools and languages used included Python for scripting, scikit-learn for SVM, TensorFlow and Keras for CNN 1D, NumPy and Pandas for data handling, Matplotlib and Seaborn for visualization, and Git for version control.

6 Evaluation

6.1 Overview of Results

The comparative analysis of Support Vector Machines (SVM) and Convolutional Neural Networks (CNN) 1D algorithms for zero-day malware detection yielded compelling results. The CNN 1D algorithm demonstrated perfect classification scores across all metrics: precision, recall, and F1-score, achieving 100% accuracy. In contrast, the SVM algorithm achieved 99% on the same metrics. These results are significant, indicating that both models perform exceptionally well, but with CNN 1D showing a slight edge in performance.

	Precision	**Recall**	**F1-Score**	**Support**
Malware	0.99	0.99	0.99	3048
Normal	0.99	0.99	0.99	2952
Accuracy			0.99	6000
Macro Avg	0.99	0.99	0.99	6000
Weighted Avg	0.99	0.99	0.99	6000

Table 1 : Accuracy results of the SVM algorithm

	Precision	**Recall**	**F1-Score**	**Support**
Malware	1	1	1	3048
Normal	1	1	1	2952
Accuracy			1	6000
Macro Avg	1	1	1	6000
Weighted Avg	1	1	1	6000

Table 2: Accuracy results of the CNN algorithm

6.2 Statistical Evaluation

Statistical tools and tests, including confusion matrices, precision-recall curves, and hypothesis testing, were used to critically evaluate the results. CNN 1D's perfect scores suggest that it could identify all malware and normal instances correctly. Meanwhile, the SVM's marginally lower scores imply a few instances where it misclassified the samples.

6.3 Experiment 1: CNN 1D Performance Analysis

The first case study focused on CNN 1D's performance. The model's architecture allowed for effective feature extraction and classification without overfitting, as evidenced by the consistent scores across the training and validation sets. The CNN's ability to capture temporal and spatial dependencies in the data was pivotal for its success.

6.4 Experiment / Case Study 2: SVM Performance Analysis

The second case study evaluated the SVM's performance. Despite not achieving the same perfection as the CNN 1D, the SVM's high scores are notable, particularly given its simplicity and the high dimensionality of the feature space. The kernel chosen for the SVM (RBF or linear) likely played a significant role in its ability to discern between classes.

6.5 Experiment / Case Study 3: Comparative Analysis

The third case study involved a direct comparison of the two models. It considered the computational efficiency, scalability, and practicality of implementation in real-world scenarios. The CNN 1D, while more accurate, required greater computational resources, suggesting a trade-off between performance and efficiency.

6.6 Discussion

The findings from the experiments indicate that while the SVM is a strong contender for malware detection, the CNN 1D's architecture makes it better suited for the complexity of zero-day malware detection tasks. The discussion should delve into the nuances of why the CNN 1D outperformed the SVM. Factors such as the CNN's layer depth, filter sizes, and the non-linearities captured by the convolutional layers contributed to its success.

Critique of the Experiments:

- The experiments were robust, yet the SVM's inability to match the CNN 1D's performance suggests room for further optimization, perhaps through more advanced kernel functions or ensemble methods.
- The CNN 1D's requirement for computational resources poses questions about its scalability, especially for organizations with limited resources.
- The SVM's slightly lower performance could also be due to the feature selection process, which might have favored the CNN's learning style.

Improvements and Modifications:

- Future research could explore hybrid models that combine the strengths of both SVM and CNN 1D.
- Further experimentation with feature selection techniques could potentially improve the SVM's performance.
- Investigating the use of transfer learning for CNN 1D might reduce the need for extensive computational resources and training time.

Contextualization:

The results should be contextualized within the broader literature on machine learning for malware detection, noting that while perfection in classification is the goal, in practice, a balance must be struck between accuracy, efficiency, and practicality.
The research contributes to the academic discourse by providing empirical evidence of the effectiveness of deep learning models in cybersecurity, a relatively nascent field of study.

The evaluation confirms that while both SVM and CNN 1D are highly effective for zero-day malware detection, the CNN 1D holds a slight edge in performance. However, this comes at the cost of computational efficiency. This research thus provides a nuanced understanding of the trade-offs involved in selecting an appropriate machine learning model for malware detection, contributing valuable insights to both the academic and practical realms of cybersecurity.

7 Conclusion and Future Work

The comparative analysis of Support Vector Machines (SVM) and Convolutional Neural Networks (CNN) 1D for detecting zero-day malware has yielded pivotal insights into the capabilities of these machine learning techniques within cybersecurity. The study concluded that the CNN 1D model outperformed the SVM, achieving a flawless performance across several metrics, while the SVM maintained a robust 99% accuracy rate.

The CNN 1D's success can be largely credited to its profound ability to process and learn from the complex patterns inherent in malware data, which is crucial for the identification of zero-day threats. On the other hand, the SVM’s slightly lower performance, coupled with its computational efficiency, underscores its continued relevance, especially in environments where resources may be constrained.

This research enriches the academic discourse on cybersecurity, particularly in the application of machine learning for malware detection. It provides a nuanced comparison of two powerful algorithms, addressing a gap in the literature and laying the groundwork for future technological advancements in the field.

In future the research efforts can be directed toward hybrid models that synergize the strengths of SVM and CNN 1D, potentially offering a balance between accuracy and resource consumption. Advancements in feature selection could further enhance SVM's performance, and the exploration of transfer learning for CNNs may mitigate the high computational costs. The expansion of these models to various platforms and integration into real-time systems remains a promising yet challenging prospect.

The study not only validates the effectiveness of SVM and CNN 1D models in the realm of malware detection but also ignites a pathway for innovative research that aims to bolster cybersecurity defenses against the continually evolving landscape of cyber threats.

REFERENCES:

Cohen, F., 1999. Simulating cyber attacks, defences, and consequences. Computers & Security, 18(6), pp.479-518.

Bazrafshan, Z., Hashemi, H., Fard, S.M.H. and Hamzeh, A., 2013, May. A survey on heuristic malware detection techniques. In The 5th Conference on Information and Knowledge Technology (pp. 113-120). IEEE.

Adkins, F., Jones, L., Carlisle, M. and Upchurch, J., 2013, October. Heuristic malware detection via basic block comparison. In 2013 8th International Conference on Malicious and Unwanted Software:" The Americas"(MALWARE) (pp. 11-18). IEEE.

Fernando, D.W., Komninos, N. and Chen, T., 2020. A study on the evolution of ransomware detection using machine learning and deep learning techniques. IoT, 1(2), pp.551-604.

Shaukat, K., Luo, S. and Varadharajan, V., 2022. A novel method for improving the robustness of deep learning-based malware detectors against adversarial attacks. Engineering Applications of Artificial Intelligence, 116, p.105461.

Saha, T., 2022. Machine learning-based efficient and generalizable cybersecurity frameworks (Doctoral dissertation, Princeton University).

Aslan, Ö.A. and Samet, R., 2020. A comprehensive review on malware detection approaches. IEEE access, 8, pp.6249-6271.

Zhang, J., 2018. MLPdf: an effective machine learning based approach for PDF malware detection. arXiv preprint arXiv:1808.06991.

Rahul, Kedia, P., Sarangi, S. and Monika, 2020. Analysis of machine learning models for malware detection. Journal of Discrete Mathematical Sciences and Cryptography, 23(2), pp.395-407.

Choudhary, S. and Sharma, A., 2020, February. Malware detection & classification using machine learning. In 2020 International Conference on Emerging Trends in Communication, Control and Computing (ICONC3) (pp. 1-4). IEEE.

Sethi, K., Chaudhary, S.K., Tripathy, B.K. and Bera, P., 2018, January. A novel malware analysis framework for malware detection and classification using machine learning approach. In Proceedings of the 19th international conference on distributed computing and networking (pp. 1-4).

Cortes, C. and Vapnik, V., 1995. Support-vector networks. Machine learning, 20, pp.273-297.

Gandotra, E., Bansal, D. and Sofat, S., 2014. Malware analysis and classification: A survey. Journal of Information Security, 2014.

Wong, W.T. and Hsu, S.H., 2006. Application of SVM and ANN for image retrieval. European Journal of Operational Research, 173(3), pp.938-950.

Savas, C. and Dovis, F., 2019. The impact of different kernel functions on the performance of scintillation detection based on support vector machines. Sensors, 19(23), p.5219.

Dehkordy, D.T. and Rasoolzadegan, A., 2021. A new machine learning-based method for android malware detection on an imbalanced dataset. Multimedia Tools and Applications, 80, pp.24533-24554.

Oak, R., Du, M., Yan, D., Takawale, H. and Amit, I., 2019, November. Malware detection on highly imbalanced data through sequence modeling. In Proceedings of the 12th ACM Workshop on artificial intelligence and security (pp. 37-48).

Millar, S., McLaughlin, N., del Rincon, J.M. and Miller, P., 2021. Multi-view deep learning for zero-day Android malware detection. Journal of Information Security and Applications, 58, p.102718.

Sharma, A., Malacaria, P. and Khouzani, M.H.R., 2019, June. Malware detection using 1-dimensional convolutional neural networks. In 2019 IEEE European symposium on security and privacy workshops (EuroS&PW) (pp. 247-256). IEEE.

Vinayakumar, R., Soman, K.P. and Poornachandran, P., 2017, September. Applying convolutional neural network for network intrusion detection. In 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI) (pp. 1222-1228). IEEE.

Mukkamala, S., Janoski, G. and Sung, A., 2002, May. Intrusion detection using neural networks and support vector machines. In Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No. 02CH37290) (Vol. 2, pp. 1702-1707). IEEE.

Eskin, E., Arnold, A., Prerau, M., Portnoy, L. and Stolfo, S., 2002. A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data. Applications of data mining in computer security, pp.77-101.

Saxe, J. and Berlin, K., 2015, October. Deep neural network based malware detection using two dimensional binary program features. In 2015 10th international conference on malicious and unwanted software (MALWARE) (pp. 11-20). IEEE.

Idika, N. and Mathur, A.P., 2007. A survey of malware detection techniques. Purdue University, 48(2), pp.32-46.

Malware detection (2018) Kaggle. Available at: https://www.kaggle.com/competitions/malware-detection/data