University of Massachusetts Dartmouth

Department of Computer and Information Science
College of Engineering

Deep Learning-Based Ransomware Detection and Prevention System for Enterprise Networks

A Project in

Computer Science

by

Tejesh Kumar Venkataraman Swaminathan

Project Mentor

Dr. Gokhan Kul

Assistant Professor
Computer & Information Science

Project Readers

Dr. Adnan El-Nasan

Associate Teaching Professor
Computer & Information Science

Dr. Razieh Fathi

Assistant Teaching Professor
Computer & Information Science

Abstract

Ransomware attacks have grown to become a critical threat in modern enterprise environments, resulting in data loss, financial damage, and operational disruption. Traditional detection methods often fall short against sophisticated and evolving ransomware variants. This project presents a deep learning-based approach for ransomware detection using file system activity logs. We begin by establishing baseline machine learning models and proceed to design and evaluate multiple deep learning models including CNN, LSTM, and a hybrid CNN-LSTM architecture. The project places particular emphasis on feature engineering techniques and their measurable impact on detection accuracy. Through rigorous experiments, we observe that our hybrid model achieves the highest detection accuracy of 81.17% when trained on a feature-engineered dataset. In contrast, its performance drops drastically to 3.90% without feature engineering, clearly highlighting its significance. We also perform cross-validation to confirm the stability of our models. This project provides valuable insights into designing scalable, real-time ransomware detection systems for enterprise security.

Table of Contents

List of Tables

Chapter 1: Introduction

1.1 Motivation

The rapid rise of ransomware in recent years has posed a severe threat to enterprise networks, with attackers using encryption-based extortion to demand ransoms from victims. Unlike traditional malware, ransomware focuses on encrypting vital organizational data and locking systems, often causing massive financial and reputational damage. According to global cybersecurity reports, ransomware attacks have increased both in volume and sophistication, rendering signature-based antivirus techniques inadequate.

1.2 Need for Deep Learning-Based Detection

To effectively detect such evolving threats, security systems must go beyond static rule-based techniques. Deep learning models, with their ability to learn complex temporal and spatial patterns from large volumes of system data, present a promising solution. These models can analyse low-level file system operations and behavioural traces to identify even previously unseen ransomware activity.

1.3 Project Objectives

This project explores a deep learning-driven approach to ransomware detection.

The key objectives include:

- Designing and evaluating CNN, LSTM, and CNN-LSTM hybrid models to detect ransomware behavior from system logs.
- Studying the role of feature engineering in enhancing model performance.
- Comparing model performance with and without feature engineering.
- Establishing a robust evaluation framework using cross-validation and confusion matrices.

1.4 Challenges and Scope

Several challenges arise when developing AI-driven detection models:

- Handling imbalanced datasets across ransomware families.
- Dealing with missing values and noisy data.
- Ensuring generalization to novel or obfuscated ransomware attacks.
- Designing scalable architectures for large-scale enterprise logs.

Chapter 2: Literature Review

Ransomware has rapidly evolved into one of the most prominent threats in cybersecurity, causing billions of dollars in losses globally. The growing complexity of attacks, coupled with their targeted nature, has led researchers to investigate advanced solutions that go beyond traditional antivirus or rule-based systems. In this chapter, we examine existing approaches to ransomware detection and discuss the gaps that motivate our deep learning-based hybrid model.

2.1 Machine Leaning Approaches

Early work in ransomware detection utilized traditional machine learning (ML) algorithms such as Support Vector Machines (SVM), Decision Trees, and Random Forests [1], [3], [7], [13]. These models were primarily trained on manually extracted features including file entropy, access frequency, and registry activity. One of the most effective models among these was the Random Forest classifier, which offered robustness and interpretability. However, the key limitation of these models lies in their dependence on handcrafted features and lack of ability to automatically generalize to unseen patterns or feature representations. Furthermore, traditional ML models often suffer performance degradation when applied to large-scale or imbalanced datasets, making them less suitable for real-time detection in enterprise environments.

2.2 Deep Learning Approaches

With the rise of deep learning (DL), researchers began leveraging models like Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), particularly Long Short-Term Memory (LSTM) networks [2], [4], [5], [6]. CNNs, by nature, are adept at capturing spatial hierarchies and patterns, making them ideal for extracting high-level features from structured datasets. On the other hand, LSTM networks are well-suited for sequential data, as they can capture temporal dependencies across event logs and system activity sequences.

Recent studies have shown that LSTM models outperform CNNs in detecting ransomware patterns from file system access logs, especially when time-related or order-preserving data is involved. However, CNNs are more computationally efficient and faster to train, making them favourable for scenarios with limited computational resources. Despite their potential, standalone CNN or LSTM models often fail to achieve a balance between generalization and precision, especially when the input data lacks temporal richness or has redundant features.

2.3 Hybrid Deep Learning Models

To address the individual limitations of CNNs and LSTMs, hybrid models have been proposed. These architectures aim to combine the feature extraction capability of CNNs with the sequential modelling strength of LSTMs. Several academic papers have highlighted that such hybrid approaches provide better detection accuracy in security-related applications, including intrusion detection systems (IDS) and ransomware classification [4], [5], [12]. Hybrid models are especially beneficial when the input data has both spatial and sequential characteristics, as is the case in our ransomware detection dataset.

In our project, we adopted a CNN-LSTM hybrid architecture to enhance performance. Our model uses CNN layers for initial feature abstraction followed by LSTM layers for modelling event order and context. This approach allows the model to leverage both global and local features, offering a more holistic understanding of ransomware behavior.

2.4 Gaps in Current Research

Despite advancements, many existing approaches suffer from one or more of the following limitations:

- Heavy reliance on static analysis or handcrafted features
- Lack of model generalization to new or unseen ransomware families
- Poor scalability to enterprise-scale logs

- Insufficient comparison of feature engineering impacts
- Limited use of real-world or diverse datasets

Our research aims to address these gaps by:

- Using dynamic features extracted from real-world ransomware behavior (via the RanSAP dataset)
- Investigating the effect of feature engineering through controlled experiments
- Proposing and evaluating a hybrid deep learning architecture that leverages both CNN and LSTM components
- Conducting detailed cross-validation and comparative model analysis

Through this work, we contribute a well-validated and scalable model pipeline for ransomware detection that advances both the practical and theoretical understanding of deep learning applications in cybersecurity.

Chapter 3: Dataset Overview

The dataset used for this project is the RanSAP (Ransomware Storage Access Patterns) dataset, a widely recognized and extensively cited dataset in the domain of ransomware detection research. This dataset forms the foundation for all our training, testing, and evaluation processes throughout the project. Its comprehensiveness, diversity, and availability of low-level storage operation data make it a gold standard for deep learning applications in ransomware behavior analysis.

3.1 Overview of RanSAP Dataset

RanSAP is a publicly available dataset that captures fine-grained block-level storage access patterns from systems infected with various ransomware families. It simulates ransomware execution in realistic virtualized environments, collecting storage-level events such as read/write access, entropy, and sector behavior. These patterns are representative of how ransomware interacts with files and disk blocks during encryption, making it an ideal dataset for low-level behavioural analysis.

The dataset includes:

- 26 ransomware families, each captured in three settings (normal, largefiles, w10dirs), and includes benign activities for contrast.
- Over 1 billion rows of data, making it one of the most extensive ransomware behavior datasets currently available.
- Fine-grained features like Logical Block Address (LBA), offset, sector size, operation type (read/write), entropy, and write amplification factor.

Its relevance is further cemented by its citation in numerous peer-reviewed research papers and studies focusing on ransomware detection, storage forensics, and data recovery. Many recent works in top-tier conferences and journals have used RanSAP as a benchmark to evaluate detection models, especially those targeting storage-level or early-stage ransomware behavior detection [8].

Table 1. Dataset Distribution

Ransomware Family	Number of Samples	Percentage (%)
AESCrypt	250,000	3.85%
Cerber-largefiles	250,000	3.85%
Cerber-w10dirs	250,000	3.85%
Cerber	250,000	3.85%
Darkside-largefiles	250,000	3.85%
Darkside-w10dirs	250,000	3.85%
Darkside	250,000	3.85%
Excel	250,000	3.85%
Firefox	250,000	3.85%
GandCrab4-largefiles	250,000	3.85%
GandCrab4-w10dirs	250,000	3.85%
GandCrab4	250,000	3.85%
Ryuk-largefiles	250,000	3.85%
Ryuk -w10dirs	250,000	3.85%
Ryuk	250,000	3.85%
SDelete	250,000	3.85%
Sodinokibi-largefiles	250,000	3.85%
Sodinokibi -w10dirs	250,000	3.85%
Sodinokibi	250,000	3.85%
TeslaCrypt-largefiles	250,000	3.85%
TeslaCrypt-w10dirs	250,000	3.85%
TeslaCrypt	250,000	3.85%
WannaCry-largefiles	250,000	3.85%
WannaCry-w10dirs	250,000	3.85%
WannaCry	250,000	3.85%
Zip	250,000	3.85%

3.2 Dataset Composition and Structure

Each log entry in the RanSAP dataset records a low-level storage event with the following fields:

- timestamp – When the operation occurred
- LBA – Logical block address accessed
- offset – Byte offset within the sector
- sector_size – Size of the accessed block
- operation_type – Type of operation (read/write)
- entropy – Shannon entropy of the accessed data
- write_amplification_factor – Ratio of actual data written vs original
- ransomware_family – Label indicating the malware family (or benign)

For training deep learning models, we excluded the timestamp and operation type fields during feature engineering, focusing on the remaining quantitative features to build a consistent numeric representation.

3.3 Dataset Preparation and Sampling

Due to the massive scale of the dataset (over 1.3 billion rows), we adopted a multi-stage preprocessing pipeline:

1. Cleaning: Rows with missing or null values in key fields were removed to ensure data integrity.

2. Balancing: The dataset is inherently imbalanced, with some families having millions of rows while others have fewer. We down-sampled each ransomware family to ensure class balance. For most experiments, we used either:

    - 250,000 samples per family (for feature engineering comparisons), or

- 10 million samples total (for full training of hybrid models).

3. Splitting: For all experiments, we split datasets into training and testing sets while maintaining class balance.

This careful preparation ensured that the models were trained on balanced and representative samples without being overwhelmed by families with disproportionately high volume.

3.4 Summary

The RanSAP dataset's scale, granularity, and diversity across ransomware families make it an ideal choice for training deep learning models aimed at ransomware detection. Its adoption in recent literature and its storage-level focus set it apart from traditional datasets based solely on network or binary-level features. By leveraging RanSAP, this project builds upon a strong empirical foundation, ensuring that our findings are grounded in realistic and challenging ransomware behaviours.

Chapter 4: Feature Engineering

Feature engineering plays a central role in the effectiveness of our ransomware detection models. This chapter details the process of transforming raw data into structured input features that enable our machine learning and deep learning models to learn meaningful patterns. This stage is one of the most critical components of the project, especially since we directly compare the effect of using vs. not using feature engineering in later experiments.

4.1 Raw Dataset Overview

The original RanSAP dataset contains millions of records simulating disk activity logs from systems infected with ransomware. Each entry includes the following attributes:

- Timestamp: The exact time at which the disk operation occurred.
- LBA (Logical Block Addressing): Indicates the specific block being accessed.
- Offset: The byte offset for the operation.
- Sector Size: Fixed at 4096 bytes in most cases.
- Operation Type: Indicates whether the operation was a read or write.
- Entropy: Represents the randomness of the accessed data (high entropy often suggests encryption).
- Write Amplification Factor (WAF): Measures the extent of disk writes beyond what is logically required, often inflated in ransomware behavior.
- Ransomware Family Label: A label indicating the ransomware variant (or benign activity).

This raw data is rich and expressive but contains several challenges for direct model consumption:

- Missing values in certain fields
- Mixed data types (categorical, continuous)
- Imbalanced number of samples across ransomware families
- Variability in entropy and amplification scales

4.2 Handling Missing Values

During preprocessing, we discovered that fields like entropy and write_amplification_factor had missing values for a subset of entries. Removing rows with missing values would have led to substantial data loss (~51.6%). Instead, we imputed missing values:

- Entropy: Filled using the mean entropy of the corresponding ransomware family.
- Write Amplification Factor: Filled using the median of the same feature across the entire dataset.

This ensured that imputation preserved the statistical distribution of the original data.

4.3 Balancing the Dataset

The raw dataset exhibited significant class imbalance across ransomware families. Some variants like Cerber and GandCrab had millions of samples, while others like AESCrypt had relatively fewer. To prevent overfitting and bias in favor of high-frequency families, we applied a uniform down sampling strategy, retaining only 250,000 samples per ransomware family. This brought the total number of samples to 6.5 million (26 families x 250,000 each), creating a balanced dataset for fair evaluation.

4.4 Feature Extraction

From the original attributes, we identified and retained the following numerical features as most relevant for detection:

- LBA
- Offset
- Sector Size

- Entropy
- Write Amplification Factor

The timestamp and operation_type columns were discarded as they were either redundant or non-numeric. These five features capture the behavioural essence of ransomware:

- Entropy and WAF highlight encryption patterns.
- LBA and offset reflect disk-level behavior.
- Sector size, though constant, was retained to maintain dimensional consistency.

4.5 Normalization

To ensure uniform learning and faster convergence in neural networks, we applied Min-Max normalization to all five retained features. This scaled values between 0 and 1, ensuring no feature dominated due to its numeric magnitude. Normalization significantly improved training stability and model performance, particularly in the CNN and LSTM architectures.

4.6 Feature Engineering Summary

The final dataset used for our deep learning experiments included the following features:

- LBA
- Offset
- Sector Size
- Entropy (imputed + normalized)
- Write Amplification Factor (imputed + normalized)

The final cleaned and engineered dataset was saved and was later converted to PyTorch tensors for model training.

4.7 Without Feature Engineering: A Contrast

To measure the direct impact of feature engineering, we created a parallel version of the dataset that bypassed all transformation steps. In this version:

- Missing values were not imputed.
- No normalization or scaling was applied.
- Feature types were left untouched.
- Basic filtering and down sampling were still applied to ensure uniform family distribution.

This dataset was used to retrain the CNN-LSTM hybrid model. The results clearly demonstrated a drop in performance compared to the feature-engineered dataset, as detailed in Chapter 6.

4.8 Why Feature Engineering Matters

Feature engineering not only improves model accuracy but also enhances generalization and robustness. Our experiments showed that:

- Models trained on the engineered dataset converged faster.
- LSTM and hybrid models were more stable.
- Predictions on minority classes improved significantly.

By isolating the effect of feature engineering, we confirmed its critical role in ransomware detection, validating this phase as a cornerstone of the project.

In the next chapter, we delve into the models we trained on these datasets and present their results and comparative analysis.

Chapter 5: Model Architectures and Experimental Results

In this chapter, we present the core implementation of our project, focusing on the development, training, and evaluation of multiple deep learning models for ransomware detection. The models are tested using the feature-engineered version of the RanSAP dataset. Each model is evaluated using various classification metrics such as accuracy, precision, recall, and F1-score. The goal of this chapter is to establish a strong foundation for understanding how each model performs, leading to the final selection of a hybrid architecture.

5.1 Convolutional Neural Network (CNN) Model (with conv1D layers)

The initial CNN model used a single 1D convolutional layer followed by ReLU activation and a fully connected layer. While this model demonstrated decent performance in feature extraction, it lacked the depth and capacity to effectively capture patterns across all ransomware families. Its best accuracy reached approximately 60%, with some families misclassified due to limited abstraction.

Architecture:

- Input shape: [Batch, 1, 6] (after reshaping features)
- Conv1D Layer: 32 channels, kernel size 3
- BatchNorm + ReLU Activation
- Fully Connected Layers: 128 units followed by output layer of 26 classes

Results:

- Accuracy: 60.21%
- Precision: 64.67%
- Recall: 60.21%
- F1-Score: 56.93%

The basic CNN model captured localized patterns well but underperformed due to lack of temporal or sequential learning [2], [4]. The limited number of features (only 6 per sample) also restricted CNN's learning capacity in this setup.

5.2 Improved CNN Model with Fully Connected Layers (FCNN)

To address limitations in the original CNN architecture, it was transitioned to a Fully Connected CNN (FCNN) model. This version removed the convolutional layers and directly applied multiple fully connected (dense) layers over the feature vectors. The motivation for this shift was twofold:

- Feature Simplicity: The dataset features (like entropy, LBA, offset, etc.) are already in a flattened, tabular format. Applying convolution on such data didn't offer much advantage, since there is no spatial or temporal locality that convolution can exploit.
- Learning Nonlinear Relationships: Fully connected layers can better capture complex nonlinear relationships across features. When combined with dropout for regularization, the FCNN was able to generalize better.

Architecture:

- Fully Connected Layers: 128 → 64 → 26 (output)
- Activation: ReLU
- Dropout: 30%

Results:

- Accuracy: 70.88%
- Precision: 73.66%
- Recall: 70.88%
- F1-Score: 68.44%

The FCNN achieved a higher accuracy of ~70.88% with stronger F1-scores compared to the original CNN, confirming its ability to learn from non-sequential features effectively.

5.3 Long Short-Term Memory (LSTM) Model

The LSTM model was initially proposed due to the sequence-like nature of some features. Despite the dataset not having explicit time-series data, the ordering of events and subtle patterns across samples appear to benefit from LSTM's recurrent architecture [4], [5], [11].

Architecture:

- LSTM Layer: Hidden size 128, 2 layers
- Dropout Layer: 30%
- Fully Connected Layer with 128 units
- Output Layer: 26 classes

Metrics:

- Accuracy: 71.62%
- Precision: 74.94%
- Recall: 71.62%
- F1-Score: 70.18%

This result indicates that the model is effective at capturing sequential or structured patterns within the data. Its balanced recall and precision also indicate consistent classification across ransomware families.

5.4 Cross Validation Analysis

A Stratified 3-Fold Cross-Validation was performed using both the FCNN and LSTM models to ensure model robustness and evaluate consistency across splits.

Process:

- Stratified K-Fold was used to maintain the label (ransomware family) distribution across each fold.
- In each fold:

  - 2/3 of the data was used for training, and 1/3 for validation.
  - The model was trained using the same hyperparameters as earlier experiments.
  - Metrics were collected after each fold.

CNN Cross-Validation Results (FCNN):

- Fold 1:
  - Accuracy: 60.01%
  - Precision: 61.67%
  - Recall: 60.01%
  - F1 Score: 56.93%

- Fold 2:
  - Accuracy: 62.45%
  - Precision: 63.91%
  - Recall: 62.45%
  - F1 Score: 60.72%

- Fold 3:
  - Accuracy: 61.18%
  - Precision: 64.69%
  - Recall: 61.18%
  - F1 Score: 58.58%

Despite the FCNN model's ability to generalize well during regular training, it showed a slight drop in performance in the cross-validation setup. This suggests that its performance may be more sensitive to data splits and highlights the importance of cross-validation for robustness testing.

LSTM Cross-Validation Results:

- Fold 1:
    - Accuracy: 66.60%
    - Precision: 65.91%
    - Recall: 66.60%
    - F1 Score: 64.76%
- Fold 2:
    - Accuracy: 66.83%
    - Precision: 67.90%
    - Recall: 66.83%
    - F1 Score: 64.05%
- Fold 3:
    - Accuracy: 63.59%
    - Precision: 66.67%
    - Recall: 63.59%
    - F1 Score: 61.37%

The LSTM model outperformed the FCNN model in cross-validation, suggesting that even without explicit timestamps, the ordering of I/O operations or structured behavior in ransomware activities gives the LSTM model an edge.

Table 1: Average Results Across 3 Folds.

Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**
CNN (Fully Connected)	61.21%	63.42%	61.21%	58.74%
LSTM	65.67%	66.83%	65.67%	63.39%

5.5 Hybrid CNN-LSTM Model

Combining the localized feature extraction capability of CNNs with the sequential dependency modeling of LSTMs has proven to be an effective strategy in multiple cybersecurity applications, including ransomware detection [4], [5], [12]. Our hybrid architecture aims to leverage these complementary strengths to boost detection performance.

Architecture:

- Conv1D Layer: 32 channels, kernel size 3, ReLU
- Batch Normalization
- LSTM Layer: Hidden size 64, 1 layer
- Fully Connected Layers: 128 → Dropout (30%) → 26 classes

Results:

- Accuracy: 81.17%
- Precision: 85.66%
- Recall: 81.17%
- F1-Score: 80.64%

The hybrid model achieved the best performance of all models, successfully capturing both spatial and structural patterns. This justified our decision to use it as the final architecture for further experiments in later phases.

5.6 Final Model Selection

The hybrid CNN-LSTM model was selected as the final model because:

- It combines the strengths of CNNs (efficient feature extraction) and LSTMs (temporal/sequence learning).
- It achieved the highest performance in all evaluation metrics.
- It provides better generalization during inference, especially under real-world ransomware simulation.

The hybrid model was able to compensate for individual weaknesses of CNN or LSTM when used alone, resulting in a robust and high-performing architecture.

Table 2: Models Comparison and Insights.

Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**
CNN (Conv1D)	60%	0.6347	0.6000	0.5642
FCNN (Dense)	70.88%	0.7494	0.7162	0.7018
LSTM	71.62%	0.7494	0.7162	0.7018
CNN-LSTM Hybrid	81.17%	0.8566	0.8117	0.8064

Table 3: Advantages and Disadvantages of Each Model.

Model	**Advantages**	**Disadvantages**
CNN (Conv1D)	Efficient for structured input; fast inference	Underperforms on tabular data; poor abstraction
FCNN (Dense)	Great for tabular data; captures nonlinear patterns	Slightly sensitive to feature distribution and scaling
LSTM	Handles sequence-like behavior and ordering well	Slower to train; may overfit if patterns are weak
Hybrid CNN-LSTM	Combines spatial feature learning and temporal reasoning	Larger model size; longer training times

Chapter 6: With vs Without Feature Engineering

A critical objective of this project was to evaluate the impact of feature engineering on ransomware detection performance using deep learning models. While earlier chapters focused on model architecture and performance metrics, this section presents a comparative study of our models trained with and without feature engineering (FE), demonstrating the tangible benefits of careful feature preparation.

6.1 Understanding Feature Engineering

Feature engineering is the process of transforming raw data into features that better represent the underlying problem to predictive models, thus improving their accuracy. In our project, we applied the following feature engineering steps to the RanSAP dataset:

1. Handling Missing Values: Columns such as entropy and write_amplification_factor had missing values. These were imputed using appropriate statistical techniques (e.g., mean or median).

2. Down-sampling: We balanced the dataset by down-sampling each ransomware family to a uniform sample size of 250,000 to address class imbalance.

3. Normalization: Continuous numeric features were normalized to a uniform scale to ensure efficient model convergence during training.

4. Tensor Formatting: The dataset was reshaped and encoded into the correct tensor format compatible with PyTorch models.

These transformations were intended to enhance the model's learning capacity and improve convergence, especially for deep learning models like CNNs and LSTMs.

6.2 Dataset Without Feature Engineering

To evaluate the necessity and effectiveness of feature engineering, we prepared a separate dataset without any form of normalization or feature transformation. This "raw" dataset retained all numeric fields in their original form and was only subjected to the minimal preprocessing necessary for compatibility with PyTorch. Specifically:

- The dataset was cleaned only to remove unusable entries.
- No imputation or normalization was performed.
- Raw fields like entropy, LBA, offset, sector_size, and write_amplification_factor were directly used.
- Categorical fields (e.g., operation_type) were excluded as the raw format could not support effective encoding without preprocessing.

This allowed us to isolate the effect of feature engineering and establish a direct performance comparison.

6.3 Experimental Setup

For consistency, the same CNN-LSTM hybrid model architecture was used for both the engineered and non-engineered datasets. We trained the model on:

- Engineered Dataset: the final dataset (after all preprocessing and normalization).

- Raw Dataset: Created from the balanced dataset (after initial preprocessing and cleaning), reduced to 250,000 samples per family, split into train and test (200k/50k per family), and converted to tensors without any feature engineering.

Training and testing were performed using the same number of epochs, batch size, and model hyperparameters.

6.4 Results and Observations

Table 4: Results of Datasets with and without Feature Engineering

Dataset Type	Accuracy	Precision	Recall	F1-Score
With Feature Engineering	81.17%	85.66%	81.17%	80.64%
Without Feature Engineering	3.90%	0.82%	3.90%	0.41%

The hybrid model trained on the engineered dataset achieved over 81% accuracy, demonstrating strong predictive performance across multiple ransomware families. In stark contrast, the same model trained on the raw, non-engineered dataset failed to generalize and exhibited extremely poor accuracy (3.90%).

6.5 Analysis and Conclusion

These results confirm that feature engineering plays a vital role in model training for ransomware detection. Without proper normalization and transformation, deep learning models struggled to extract meaningful patterns from the data. Prior studies have emphasized the critical role of feature engineering — including normalization, imputation, and class balancing — in improving detection accuracy for ransomware and other malware [6], [9], [10].

The significant performance difference between the two datasets validates our hypothesis that:

- Raw telemetry data from storage systems, while rich, requires processing to be usable.
- High variance in scale among features like LBA and entropy can mislead model training.
- Missing or improperly scaled features can skew the decision boundaries of deep learning models.

Chapter 7: Conclusion

In this project, we set out to address a critical cybersecurity concern in the modern digital landscape—ransomware. These attacks, capable of crippling systems and causing devastating data loss, require robust and adaptive detection mechanisms. We proposed a deep learning-based solution that utilizes behavioural features from system-level I/O traces to identify ransomware activity in real time. Through the application of a structured multi-phase methodology, we rigorously designed, trained, and evaluated various machine learning and deep learning models, culminating in the development of a high-performing hybrid CNN-LSTM architecture.

The project began with a comprehensive literature review of existing methods and their limitations, identifying the gap for a data-driven, real-time, and scalable ransomware detection system. We selected the RanSAP dataset—a well-established and extensively used benchmark in ransomware research—owing to its rich diversity in ransomware families and behavioural logs. Preprocessing and feature engineering were carefully applied to ensure data quality and to extract critical behavioural indicators such as entropy and write amplification.

Our approach involved experimenting with models like CNN, LSTM, and a fully connected variant of CNN (FCNN). After analysing model performance in terms of accuracy, precision, recall, and F1-score, we found that the hybrid CNN-LSTM model yielded the best results, achieving over 81% accuracy on the final feature-engineered dataset. This model effectively captured the structured patterns in I/O behavior while benefiting from CNN's ability to extract localized features.

Finally, we trained and tested the hybrid model on an unprocessed version of the dataset. The drastic performance drop (accuracy of ~3.9%) in the raw model reinforced the significance of preprocessing and engineered features. We further strengthened our findings by conducting 3-fold stratified cross-validation, which confirmed that the LSTM model consistently outperformed CNN due to its ability to leverage sequential patterns and structured ordering in the data.

Chapter 8: Future Work

This work lays a strong foundation for future advancements in ransomware detection. Potential extensions of this project could involve:

- Integrating advanced feature selection techniques to further improve performance and reduce noise.
- Exploring real-time deployment within enterprise monitoring systems using live I/O streams.
- Simulating sandboxed environments with live ransomware samples in a more controlled lab setup.
- Incorporating temporal or graph-based models to understand behavioural shifts over time.
- Exploring adversarial robustness to make the detection system resistant to evasion techniques.

In conclusion, our hybrid CNN-LSTM model, paired with well-crafted feature engineering, presents a powerful, scalable, and interpretable solution to ransomware detection, aligning well with the real-time demands and operational constraints of modern enterprise networks.

References

[1] L. Moujoud, M. Ayache and A. Belmekki, “A state-of-the-art survey on ransomware detection using machine learning and deep learning,” in *Modern Artificial Intelligence and Data Science*, vol. 1102, Springer, Cham, 2023, pp. 245–270. doi: 10.1007/978-3-031-33309-5_15.
[2] P. K. Gurumallu, R. Dembala, D. Y. Gowda, A. K. M. Annaiah, M. K. M. V. Kumar, H. Gohel and D. Y, “Exploring deep learning approaches for ransomware detection: A comprehensive survey,” *Recent Advances in Computer Science Communications*, vol. 18, pp. 1–19, 2025. doi: 10.2174/0126662558279673240515054741.
[3] R. B. M. Khammas, “Ransomware detection using random forest technique,” *ICT Express*, vol. 6, no. 4, pp. 325–331, 2020. doi: 10.1016/j.icte.2020.11.001.
[4] U. Zahoora, A. Khan, M. Rajarajan, et al., “Ransomware detection using deep learning based unsupervised feature extraction and a cost sensitive Pareto ensemble classifier,” *Scientific Reports*, vol. 12, no. 15647, 2022. doi: 10.1038/s41598-022-19443-7.
[5] I. Bello, H. Chiroma, U. A. Abdullahi, et al., “Detecting ransomware attacks using intelligent algorithms: Recent development and next direction from deep learning and big data perspectives,” *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, pp. 8699–8717, 2021. doi: 10.1007/s12652-020-02630-7.
[6] S. Homayoun, A. Dehghantanha, M. Ahmadzadeh, S. Hashemi, R. Khayami, K. R. Choo and D. E. Newton, “DRTHIS: Deep ransomware threat hunting and intelligence system at the fog layer,” *Future Generation Computer Systems*, vol. 90, pp. 94–104, 2019. doi: 10.1016/j.future.2018.07.045.
[7] H. Daku, P. Zavarsky and Y. Malik, “Behavioral-based classification and identification of ransomware variants using machine learning,” in *Proc. 17th IEEE Int. Conf. Trust, Security and Privacy in Computing and Communications (TrustCom)*, New York, USA, 2018, pp. 1560–1564. doi: 10.1109/TrustCom/BigDataSE.2018.00224.
[8] L. F. Maimó, A. H. Celdrán, Á. L. P. Gómez, F. J. G. Clemente, J. Weimer and I. Lee, “Intelligent and dynamic ransomware spread detection and mitigation in integrated clinical environments,” *Sensors*, vol. 19, no. 5, p. 1114, 2019. doi: 10.3390/s19051114.
[9] J. A. Gómez-Hernández, L. Álvarez-González and P. García-Teodoro, “R-Locker: Thwarting ransomware action through a honeyfile-based approach,” *Computers & Security*, vol. 73, pp. 389–398, 2018. doi: 10.1016/j.cose.2017.11.019.
[10] A. Cohen and N. Nissim, “Trusted detection of ransomware in a private cloud using machine learning methods leveraging meta-features from volatile memory,” *Expert Systems with Applications*, vol. 102, pp. 158–178, 2018. doi: 10.1016/j.eswa.2018.02.039.
[11] M. Jemal and D. C.-T. Lo, “Detection of ransomware attack using deep learning,” in *Proc. IEEE Conf. Dependable and Secure Computing (DSC)*, Tampa, FL, USA, 2023, pp. 1–9. doi: 10.1109/DSC61021.2023.10354186.
[12] G. O. Ganfure, C.-F. Wu, Y.-H. Chang and W.-K. Shih, “DeepWare: Imaging performance counters with deep learning to detect ransomware,” *IEEE Trans. Comput.*, vol. 72, no. 3, pp. 600–613, Mar. 2023. doi: 10.1109/TC.2022.3173149.
[13] D. Smith, S. Khorsandroo and K. Roy, “Machine learning algorithms and frameworks in ransomware detection,” *IEEE Access*, vol. 10, pp. 117597–117610, 2022. doi: 10.1109/ACCESS.2022.3218779.