UNIVERSITY OF CAPE TOWN

DEPARTMENT OF COMPUTER SCIENCE

CS/IT Honours Project Final Paper 2022

Title: Deep Learning For Ransomware Detection

Author: Christopher Lamprecht

Project Abbreviation: DL4CN

Supervisor(s): Josiah Chavula, Geoff Nitschke

Category	*Min*	*Max*	Chosen
Requirement Analysis and Design	*0*	*20*	5
Theoretical Analysis	*0*	*25*	
Experiment Design and Execution	*0*	*20*	20
System Development and Implementation	*0*	*20*	
Results, Findings and Conclusions	*10*	*20*	20
Aim Formulation and Background Work	*10*	*15*	15
Quality of Paper Writing and Presentation	*10*		10
Quality of Deliverables	*10*		10
Overall General Project Evaluation (*this section allowed only with motivation letter from supervisor*)	*0*	*10*	
Total marks		**80**	

Deep Learning for Ransomware Detection in Encrypted Traffic

Christopher Lamprecht*
LMPCHR002@myuct.ac.za
University of Cape Town
Cape Town, Western Province, South Africa

Abstract

A recent increase in ransomware attacks has necessitated tools that can detect threats within a computer network. Machine learning (ML) has been proposed as a solution to detect ransomware within network traffic. However, whilst many ML models are developed to obtain high classification accuracy, we also test their runtime memory performance and inference times. We further test the effect training the models on a reduced feature space has on the classification and computational metrics. In this study, we evaluate multi-layer perceptrons (MLP), 1-dimensional and 2-dimensional convolutional neural networks (CNN) and long short-term memory (LSTM). Experiments are performed to give insight into each model's ability to reduce its false positives (FPs). We find a consistent increase in memory usage as the feature space increases and that the lowest amount of FPs is obtained when using 50% to 66% of the features. The study evaluates 17 models for their effectiveness in classifying ransomware network traffic. Efficiency in this context is considered a trade-off between accuracy, false positives, runtime memory usage, and inference time.

Keywords: Deep Learning, Malware, Ransomware, Network Traffic

1 Introduction

While cyberattacks have been prevalent for approximately 35 years, the recent increase in technology users has amplified the scope of technology threats, necessitating the development of new methods of protection [35]. Malicious software (malware), a type of cyberattack, is a term for software installed on a user's computer without their consent, intending to cause harm [1]. Ransomware is a form of malware that encrypts a user's files/devices, denying them access to these files/devices until a sum of money (the ransom) is paid [34]. In 2022, there were 493 million ransomware attempts, a 162.29% increase since 2019[2]. Richard et al. [43] reported that this is due to attackers now being able to use Bitcoin to receive payments, mitigating the risk of the payments being traced back to them. Since then, Mott et al. [32] have shown that ransomware has been the most common form of malware. Ransomware has also been the cause of catastrophic monetary loss, projected to reach $265 billion (USD) by the year 2031[3] . Certain variants, like the WannaCry worm [30], have even posed potential risks to human life. While many tools and practices exist for malware detection, this study primarily focuses on the detection of ransomware due to the seriousness of its impact.

Historically, networks were secured against malware attacks through a Network Intrusion Detection System (NIDS), a technology used for detecting vulnerability exploits on a network and reporting them to the administrator. A functioning NIDS can identify and block malware - or, more generally, blacklisted traffic - from entering the network. NIDS could do this by deploying a broad range of techniques to identify and classify traffic: port analysis, deep packet inspection (DPI) and statistical modelling of packet flow [8]. However, more recent standard practice around networking has made it difficult for the aforementioned detection systems to function well. Port numbers have become a less reliable indicator of an application type due to port obfuscation. This technique masks the destination port of some traffic by selecting random or changing port numbers for each communication session [56]. Similarly, the adoption of dynamic IP addresses - IP addresses that change over time - has also made it difficult for NIDS, given that NIDS rely on their ability to associate traffic from specific IPs with devices. Furthermore, much of modern internet traffic undergoes some encryption protocol; notably, in 2017 $\approx$ 75% of analyzed malware used encryption [52]. Encrypted network packets, which are network packets that have been encoded or scrambled before being transmitted over a network, present a challenge for conventional DPI inspection, as they necessitate access to the payloads contained within the packets, which are now encrypted. [40].

Novel technologies that perform intrusion detection - capable of working within the previously established networking practices - are required. Machine Learning (ML), a subset of artificial intelligence (AI) that utilises data and algorithms to replicate human learning, is a prime example of these technologies. However, many existing ML-based intrusion

[1]https://www.cisco.com/site/us/en/products/security/what-is-malware.html
[2]https://www.statista.com/statistics/494947/ransomware-attacks-per-year-worldwide/#:~:text=In%202022%2C%20organizations%20all%20around,nearly%20155%20million%20cases%2C%20respectively.
[3]https://cybersecurityventures.com/global-ransomware-damage-costs-predicted-to-reach-250-billion-usd-by-2031/#:~:text=Ransomware%20will%20cost%20its%20victims,payloads%20and%20related%20extortion%20activities.

detection models require carefully engineered feature selection, a slow and difficult task [23] [56] [42]. Deep learning (DL) algorithms have been proposed to address this challenge. These are neural networks consisting of many hidden layers between the input and output, enabling the model to understand more complex patterns in the data. This makes DL algorithms well-suited for automatic feature extraction, making them adaptable to extracting features from encrypted data [57]. Moreover, DL models can reason with a much higher dimensionality than shallow ML counterparts, enabling them to learn more complicated relationships [56]. However, The trade-off is that these algorithms can be significantly more expensive to train and run due to their network structures' large number of parameters. Frequently, this increase in resource requirements is ignored, as measurements such as accuracy are considered more important metrics when evaluating a model. As a result, deploying these DL models on resource-constrained devices like IoT and smart devices, where real-time execution is essential, can present significant challenges [28]. Similarly, resource-constrained networks, such as community networks, require lightweight DL models that are efficient in terms of memory resource usage whilst keeping a high classification accuracy [53].

This study aims to examine both quality metrics - accuracy, false positives (FPs), and false negatives (FNs) - and footprint metrics - memory consumption and inference time - related to detecting ransomware within encrypted network traffic using deep learning models. Additionally, the models are trained with inputs of varying lengths (network packets of different sizes in bytes) to assess the impact of input size on the quality and footprint metrics. The main objectives of this study are:

1. Assess the performance of MLP, 1D CNN, 2D CNN, and LSTM models in classifying encrypted network traffic as benign or ransomware, considering the accuracy, false positive/negative rate, inference time and runtime memory usage of the models.
2. Investigate the impact of using a window of samples on the accuracy, false positive rate, inference time and runtime memory consumption on an LSTM by grouping multiple samples into one sample.
3. Investigate the impact of dimensionality reduction on accuracy, false positive rate, inference time and runtime memory consumption across the deep learning architectures (MLP, 1D CNN, 2D CNN, and LSTM) by decreasing the feature space from 30 samples down to 20, 15 and 5 samples.

The rest of the paper is structured as follows: Section 2 discusses some of the topics used in this study. Section 3 discusses the related work on malware and ransomware detection. Section 4 describes the dataset, evaluation criteria, and model architectures used in the paper. Section 5 discusses the experiments performed to answer the research objectives. Section 6 presents the results of the experiments and what they imply. Lastly, Section 7 present the conclusions of the paper.

2 Background

2.1 Multi-Layer Perceptron (MLP)

A Multi-Layer Perceptron (MLP), often referred to as a feed-forward artificial neural network (ANN), consists of input, hidden and output layers. These layers consist of interconnected nodes, where each node processes the input data and passes it to the next layer using weighted connections. At every node, except for the input nodes, a weighted sum of all previous nodes is passed into a non-linear activation function [53]. For binary classification, the outputs of the last layer are passed into a sigmoid function, which maps the output to a value between 0 and 1.

MLPS can learn relationships and patterns between linear and non-linear data [15]. Despite this, an MLP can struggle to learn more complex relationships and patterns within data, highlighting its use as a baseline model. [10]

2.2 Convolutional Neural Network (CNN)

A convolutional neural network (CNN) is mainly used for image pattern recognition tasks, excelling in processing and analyzing visual data. They comprise three layers: convolutional, pooling, and fully connected layers[37]. The convolution layer employs image kernels containing a limited set of trainable parameters to capture significant spatial features from the input. The pooling layer reduces the spatial dimensionality of the data, and the fully connected layer consists of neurons connected to every neuron in the successive layer. CNNs are known for recognising patterns in highly dimensional data [38]. Additionally, CNNs make use of sparsely connected layers, where the neurons that make up a layer have an imposed limit on how many outgoing connections they can have - this has been seen to significantly reduce the complexity of the model without a large impact on performance [18]. 1D data can be applied to CNNs by converting the data into an image format; for example, a network traffic sample with 30 features can be converted into a 5×6 image.

2.3 Long Short-Term Memory (LSTM)

A Long Short-Term Memory (LSTM) model is a branch of recurrent neural networks (RNN) that can capture long-term dependencies in sequential data. Hochreiter et al. [19] introduce the LSTM as a solution to the vanishing gradient problem - a problem found during the training of neural networks where the gradients that update the network become extremely small. At each state of an LSTM, the output of the previous state is coupled with the input to produce an output and the next state. LSTMs are well suited to sequential data, time series, and any data where relationships between samples are present.

2.4 Encrypted Network Traffic

Network traffic is the data, called network packets, that flow between devices on a computer network, allowing the devices to communicate. The packets consist of the transmitted content, called the payload, and the features necessary for delivering the packets to their correct destination. Encryption happens when the data present in the network traffic is encoded or scrambled using cryptographic algorithms [26]. This is done to make communications more secure and to protect user privacy. However, some malware uses this encryption to hide its activity, disguising itself as safe encrypted traffic [39]. Encrypted network traffic also presents challenges for traditional traffic classification approaches because they rely on the payload, which is now scrambled, making it difficult to extract valuable information [45]. ML methods have been proposed to overcome this, as they can still learn information in encrypted data [6].

Vinayakumar et al. [49] show that network traffic often contains complex relationships and dependencies among different packets. Ransomware traffic is known to have behavioural characteristics, with different ransomware strains producing different network traffic with unique relationships between the network packets [31].

3 Related Work

3.1 Traditional Approaches to Traffic Classification and Malware Detection

Due to the increase in malware and network intrusions, detecting such malware before it inflicts substantial damage has become a popular practice for protecting a computer network. A common approach to network intrusion detection is deep packet inspection (DPI) [22][46]. This involves inspecting the payload of each packet. Previously, methods involved using regular expressions as signatures for the different protocols. However, these methods struggle when the regular expressions need to be updated for each new protocol released. Moreover, these methods become less efficient or unusable when dealing with encrypted traffic. To combat this, Sherry et al. [46] proposed a system to perform DPI with encrypted data through encryption schemes. However, adopting this system has a large overhead, as the system requires a different approach to the Hyper-Text Transfer Protocol (HTTPS), one of the most commonly used protocols in network traffic.

To overcome the problems of DPI, ML has been proposed as a method to classify network traffic and, hence, detect malware. Modi et al. [29] performed ransomware detection using machine learning on encrypted network traffic. Three machine learning classifiers were used: a support vector machine (SVM), random forest and logistic regression. The results show that the random forest performs best, obtaining an accuracy of 99.9% and a false-positive rate of 0%. However, the dataset used by Modi et al. contained only 20 ransomware families and fewer than 700 ransomware samples. Furthermore, the classifiers were not tested on any unseen data. Being able to detect unseen traffic is important, as Beaman et al. [2] show that new ransomware variants necessitate the capability of tools to identify not only familiar patterns but also unseen ransomware.

An issue with the ML methods mentioned above is their requirement for a carefully engineered feature selection, a slow and painful task. As a result, ANNs have been proposed due to their ability to learn these features independently. Singh et al. [47] and Ben et al. [3] use an MLP to classify benign and malware network traffic, obtaining accuracies of 99.2% and 97.6%, respectively. Despite this success, due to MLP's simpler architecture, it can struggle to learn more complex features and relationships in the data [53]. However, this simplicity also means that the MLP is less computationally expensive than more complex models [10], making it a good baseline model to evaluate the performance of those more complex models.

3.2 Deep Learning Approaches for Malware Detection

To learn more complex patterns and relationships within data, deeper ANNs have been proposed, such as CNNs and LSTMs. Mohammad et al. [23] reported that one-dimensional (1D) CNNs are the optimal choice when working with network traffic classification due to their ability to capture spatial dependencies between adjacent bytes within the network packets (1D data), which helps find patterns for each protocol or application. This is opposed to two-dimensional (2D) CNNs, which recognize patterns in 2D data, such as images. Aceto et al. [1] found that 1D and 2D filters obtain the same results and hence state that network traffic can be considered as 1D, mitigating the need for 2D filters. Furthermore, Wang et al. [50] observed that a 1D CNN achieved higher accuracy in classifying network traffic than a 2D CNN. Conversely, Chen et al. [7] employed a 2D CNN model consisting of two convolutional layers, two pooling layers, and three fully connected layers to classify protocols and applications. They transformed the initial time series data into 2D images using reproducing kernel Hilbert space (RKHS) embedding. Their 2D CNN model demonstrated superior performance, obtaining higher accuracy when compared to traditional machine learning methods and an MLP in the task of protocol and application classification. This distinction between the performance of 1D and 2D CNNs paves the way for exploring their application in our study.

The past works have demonstrated the effectiveness of using CNNs for network traffic classification. Despite being more commonly employed in image-based applications, CNNs have also achieved high levels of accuracy in this context. However, CNNs can struggle when learning long-term dependencies between the data, and the accuracies obtained could be improved by considering such factors [31]. This study addresses the limitations of CNNs in learning long-term dependencies by incorporating an LSTM (Long Short-Term Memory) model. By leveraging an LSTM, we aim to enhance classification accuracy and overcome the challenges associated with CNNs in handling long-term dependencies.

LSTMs have shown their success in network traffic classification and malware detection. Dicks [10] used an LSTM to classify network traffic into its source application. The LSTM was compared to a simpler MLP, where the LSTM outperformed the MLP in classification accuracy, showing an LSTM's success in learning relationships within network traffic. However, the LSTM did suffer from slower prediction speed and did not deal with malware network traffic classification, let alone ransomware traffic. Homayoun et al. [20] compared MLP, CNN and LSTM models, assessing their effectiveness in detecting ransomware binaries. The results demonstrated that the LSTM model outperformed the MLP and CNN in this task. Similarly, Maniath et al. [24] used an LSTM to classify ransomware traffic based on API calls, obtaining an accuracy of 99.6%, further motivating using an LSTM.

While the LSTM may seem superior to network malware detection due to the underlying relationships with network traffic, Zeng et al. [56] show a CNN outperforms an LSTM in classification accuracy. Their work presented a framework for deep learning-based encrypted network traffic classification and intrusion detection and tested a CNN and an LSTM for performing the same task. The paper accurately classifies encrypted traffic, with CNN obtaining an accuracy of 99.85%. In contrast, the LSTM obtains an accuracy of 99.41% for intrusion detection. To test this, our study compares CNNs against an LSTM to assess which model performs better in ransomware detection.

On top of detection tools being able to classify traffic as ransomware, they also need to accurately detect benign traffic and keep their false positive rate (FPR) low. Nguyen et al.[33] emphasizes the significance of maintaining a low FPR in malware detection. The authors highlight the potential strain a system with a high FPR can impose on a network, leading to serious malware threats being overlooked or ignored. Furthermore, the authors address the challenge of establishing an acceptable threshold for FPR, stating that there is no universally agreed-upon measure. This is emphasized by Mell et al. [27], who discuss the difficulty of measuring false alarms due to the diversity of networks. This lack of consensus adds complexity to the task of effectively balancing the trade-off between accurate detection and minimizing false positives in malware detection systems. As a result, our study investigates the ML model's abilities to keep a low FPR.

4 Methods

4.1 Dataset

This section discusses the dataset that will be used in this paper. Selecting a good dataset is important to ML as it affects the overall success of the ML model [4]. Despite the comprehensive amount of research that has been done on ransomware detection using DL, the datasets used in the evaluation of the past research are rarely made public. On top of the limited amount of datasets available, for this study, the datasets selected need to fulfil the following criteria: they must contain encrypted network traffic, they must contain ransomware-specific traffic, and they must consist of relatively new (past five years) network traffic that resembles the current flow of network traffic and malware. Following these criteria, we found the Open Access (OA) Dataset[5] [4] and the dataset provided by Modi et al. [29].

4.1.1 Scenario. The OA Dataset is based on a file-sharing network. This dataset assumes that files are accessed from central repositories using a file-sharing protocol, a technique commonly employed by many corporations to allow flexible access to files. The encryption observed within the OA dataset arises from two main reasons: first, file-sharing protocols or encrypted VPNs to secure the traffic, and second, the ransomware itself that writes encrypted content.

The OA dataset comprises two subsets: benign network traffic and infected ransomware network traffic. The dataset is unbalanced, containing significantly more benign traffic than infected traffic. To address this issue, SMOTE[6] (Synthetic Minority Over-sampling Technique) is employed to oversample the infected subset to a ratio of 7:3, as depicted in Fig 1. SMOTE was chosen due to its successful application in various studies. Rustam et al. [44] used SMOTE to resample their data, resulting in a classification accuracy increase of 4%. Gicic et al. [17] use SMOTE to oversample their imbalanced dataset, which increased the prediction accuracy of their system. Oversampling the dataset was preferred, as recent literature [25] indicates that it yields better results than its counterpart, undersampling. A further weakness of the OA dataset is that the network traffic is obtained in a network of just over 300 staff users. In contrast, a network with many more users may produce different traffic. This can make it difficult for the DL models to generalise to networks with more or fewer users and different network environments. However, this is the case with most network traffic datasets, as no network environment is the same [27]. For

[4]https://www.aeologic.com/blog/the-importance-of-evaluating-datasets-for-ai-development/#:~:text=Data%20is%20the%20lifeblood%20of,also%20the%20potential%20for%20application.

[5]http://dataset.tlm.unavarra.es/ransomware/papers.html

[6]https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.SMOTE.html

this reason, Gibert et al. [16] discuss the difficulty of comparing malware detection accuracies across different works due to the differences in the datasets.

The dataset provided by Modi et al. [29] only contains the traffic from 20 ransomware, many of which are already included in the OA Dataset; hence, this dataset is not used.

Numpy[7] and Pandas[8] were used to read in and manipulate the data.

4.1.2 Infected Dataset.

The infected traffic in the OA dataset contains over 120 hours of ransomware traffic, consisting of 94 samples from 43 ransomware families, with some families having multiple samples. The ransomware families and the year in which the most recent binary was obtained can be seen in Table 8 in the appendix. The traffic has been captured and updated from 2015 to 2021. Different samples have different amounts of network packets as each sample varies in traffic trace duration time.

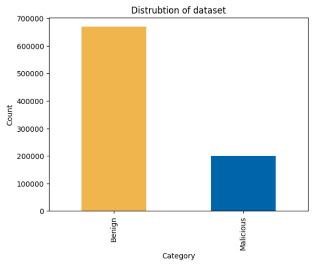


Figure 1. Dataset Distribution

4.1.3 Benign Dataset. The benign traffic in the OA dataset contains normal traffic - network traffic where user activity was not significant - generated by around 300 users on a network campus over one week (excluding weekends). The network traffic for each day is put into its subset.

4.1.4 Preprocessing. Four data groups were made to train, test, and assess the models. These groups are the train set, validation set, test set, and extra test set. Each subset contains benign and ransomware network traffic, except the additional test set, which contains only benign traffic for false positive testing. The test set contains unseen network traffic. However, it also contains ransomware from the same families that are present in the train set.

4.2 Evaluation Criteria

The deep learning models are tested on the following criteria: *Model Accuracy*, *False Positives*, *False negatives*, *Runtime Memory* and *Inference Time*. The effectiveness of the models is determined by considering a trade-off among these criteria. However, accuracy and false positive rate are the most important factors to be prioritized. Numerous metrics are employed to evaluate the models, and ascertaining the model's overall effectiveness requires consideration of all these metrics.

False Positives (FPs) are given higher importance than accuracy due to their ability to give a true performance indication and the effect they can have on organisations using the ransomware detection system [33][12][5]. Whilst accuracy is not as important as FPs, it indicates the model's overall performance and remains unbiased toward any particular class. Inference time and runtime memory usage are given lower importance due to unreliable measurement methods and variability. Park et al. [41] show that inference time can be sped up using more powerful hardware, meaning that inference time can change according to which computer the models are run on. Courtaud et al. [9] delve into the inaccuracy of memory inference measurements caused by induced delays. Similarly, Section 4.2.5 discusses a paper [14], which provides insights into the unpredictability of measuring runtime memory usage.

However, it should be noted that the inference time and runtime memory usage measurements were all obtained in the same environment, under the same conditions, and therefore still have an impact on the importance of the models. Additionally, the importance of the metrics can be adjusted if a user wishes to favour them differently. For example, if the user is more interested in inference time, they can give it higher importance.

4.2.1 Accuracy. Accuracy is the number of correct predictions divided by the total number of predictions and characterizes the model's overall performance across all classes. It is useful when the different classes all have the same importance. In this case, each ransomware strain is assumed to have an equal capability to inflict damage, and therefore, accuracy provides a good indication of the model's performance. However, in the case of an imbalanced dataset, such as the one used in this study, accuracy can be misleading, as a high accuracy can still be obtained if only the majority class is correctly predicted. Therefore, other metrics are considered when evaluating the model's performance.

4.2.2 False Positives. In the context of this study, a false positive (FP), is when a model classifies benign traffic as ransomware. An FP will require the network owner to shut down the network and suspend business operations until the ransomware or false positive is thoroughly inspected. Brewer et al. [5] report that this can result in a loss of productivity and safety, impacting the security team's response to real threats[9].

As a result, a model that can detect ransomware well but produces false positives on benign network traffic is as detrimental as a model with a poor ability to detect ransomware. Either way, the organisation/network owner will experience disruptions and downtime.

[7] https://numpy.org/
[8] https://pandas.pydata.org/

[9] https://www.lupovis.io/the-dangers-of-false-positives-in-cybersecurity-and-how-to-avoid-them/

Furthermore, while accuracy is a valuable metric, its performance indication can be obscured if its proportion to the number of samples is small [33]. This aspect is particularly relevant to this study, as a model may exhibit high accuracy, but if it processes a large number of samples, such as network traffic, and produces a significant number of FPs, it can negatively impact the security of a network. The potential impact of false positives on a network underscores the importance of considering both accuracy and false positive rate as crucial evaluation metrics for the model's effectiveness [12].

Establishing a baseline for an acceptable FP rate is difficult, as different tools and organisations deal with FPs differently. Nguyen et al. [33] state that there is no threshold for an appropriate number of FPs and propose a system for selecting a threshold. However, Yang et al. [54] proposes a DL model called *DeepMal* for detecting malware traffic. *DeepMal* can detect 70% of malware whilst keeping the false alarm rate below 3%. This is the baseline that the models use for their FPR to be deemed 'successful'.

4.2.3 False Negatives. False negatives (FNs) - ransomware traffic classified as benign - also hold significance due to their ability to ascertain the model's reliability. A model might demonstrate proficiency in detecting benign traffic, yielding a low FP rate. However, it could still exhibit subpar performance when identifying ransomware incorrectly, categorizing it as benign.

Zhu et al. [58] and Yewale et al. [55] report the importance of measuring the FNs and FPs in tandem when using ML models to detect malware to evaluate the overall performance. However, while FNs remain important, recent literature emphasizes that false positives (FPs) receive greater attention in the context of malware detection systems. Hence, they are given higher significance in the evaluation process [5][12].

4.2.4 Inference Time. For ransomware detection to be successful, it must be able to detect the ransomware quickly. If the tool does not detect the ransomware before too much data is encrypted, it can be considered useless [13]. Therefore, inference time - the time taken for a model to make a prediction - is another metric used to determine the success of the models in this study. There is no 'standard' time to detect ransomware, as each ransomware family behaves and acts differently. To determine a baseline inference time that a DL model needs to achieve, we use an experiment done by Splunk's SURGe team[10] whereby they timed how long it took some of the most popular ransomware families (some of which occur in the dataset) to encrypt 100 000 files. It was reported that it took 42 minutes and 54 seconds to encrypt all the files (38.85 files are encrypted every second). This means that it takes 25.74 milliseconds on average to encrypt one file. This is the inference time the DL models must achieve to obtain a successful inference time.

To determine the inference time for a single sample, we record the cumulative time taken by the model during each iteration of the inference process using *Pythons time module*[11]. Then, we divide this accumulated time by the total number of samples in the test set to obtain the desired metric. This method of measuring inference time has been used successfully by Wang et al. [51]

4.2.5 Runtime Memory Usage. DL models that consume excessive memory during runtime can lead to failed jobs and decreased productivity. Furthermore, such models may encounter challenges when deployed on systems with limited computational resources, such as small IoT devices[28]. Therefore, the memory used up at runtime is an important metric to evaluate the effectiveness of a DL model. However, Gao et al. [14] discuss the limitations and difficulties of measuring accurate runtime memory usage for the following reasons: hidden implementation details, the difficulty of extracting hidden factors and the differences in the types of DL models. As a result, this study compares the runtime memory usage of various models within the same environment and conditions rather than establishing a specific threshold for what qualifies as 'acceptable' runtime memory usage.

4.3 Architecture and Hyperparameter Selection

Hyperparameters are modifiable parameters not learned from the data but set by the user before training. The values of the hyperparameters control the learning process and, consequently, the model's performance. Therefore, selecting the right ones is essential. We used KerasTuner*, a hyperparameter optimization framework that automates much of the work to find the best hyperparameters for our DL models. Josi et al. [21] demonstrated the effectiveness of utilizing KerTuner to fine-tune hyperparameters, resulting in a test accuracy of 80.96%. This serves as motivation for its adoption in this study. The hyperparameters that undergo tuning are the neurons in each layer, activation functions at each layer if dropout is needed, and the specific dropout values to be used.

The models created and used in this study were implemented using the open-source library *Keras*[12], which provides an interface to the *TensorFlow library*[13] - an open-source library for ML. The model architectures and hyperparameter values can be seen in Table 5 in the appendix.

[10] https://www.darkreading.com/application-security/here-s-how-fast-ransomware-encrypts-files

[11] https://docs.python.org/3/library/time.html

[12] https://www.tensorflow.org/guide/keras

[13] https://www.tensorflow.org/

5 Experiment Procedure

5.1 Summary of experiments performed

Table 1. Experiments and Research Objectives

Experiment	Description	Brief Deception Of Target Objective
1	Establishing a simple MLP baseline and training DL models.	Objective 1: Evaluate ML models for encrypted traffic classification.
2	Varying window on input.	Objective 2: Sliding window
3	Testing for false positives.	Objective 1&2
4	Varying input feature-length.	Objective 3: Dimensionality reduction effect on models

5.2 *Experiment 1:* Establishing MLP baseline and training DL models

To determine whether using DL models for ransomware detection is worth the additional cost, a simple MLP is used as a benchmark model to perform the same task. Firdausi et al. [11] demonstrated the success of using an MLP for malware detection, highlighting its value as a benchmark model. The other DL models (CNNs and LSTMs) use features that an MLP does not have, such as convolutional layers and gates. The MLP can also be used to decide whether these additional features are necessary or not when it comes to ransomware detection. Moreover, the MLP can be trained and tested using the same input data provided to the DL models. In contrast, other ML models might necessitate a different format of input data and may struggle to handle the high dimensionality of network traffic.

From here, the DL models (1D CNN, 2D CNN and LSTM) are trained and tested, obtaining each model's evaluation metrics mentioned in Section 4.2.

5.3 *Experiment 2:* Using a sliding window

The LSTM model undergoes a variation for this experiment: a sliding window of samples is employed. In contrast to the previous approach of analyzing individual samples in isolation, this experiment involves taking multiple samples, grouping them into a window, and treating the windowed set as a single sample for evaluation, making a window LSTM (WLSTM). The LSTM is chosen in this experiment due to its success in past works [24][20][10]. Oord et al. [36] use a sliding window approach with time series data to capture dependencies over time to generate realistic audio samples, showing its success in capturing temporal dependencies. To our knowledge, the windowed approach has not been used for ransomware classification in previous works.

5.4 *Experiment 3:* Testing for false positives

Two methods are employed to evaluate the FPs obtained by the DL models. The first method involves counting the number of FPs the models generate on the test set. The second method utilizes six days of exclusively benign network traffic to assess how many FPs the DL model produces over six days without ransomware traffic. This returned number is divided by the number of samples in the traffic to determine the number of FPs per record. This gives a more realistic overview of the ability of the models, as just looking at the false positives on the test set does not give a true representation. For example, if we have a 1% false positive rate (FPR) and the test set consists of 100 samples, with five being malicious, then the expected number of FPs is 95*1%=0.95 FPs[14]. Whereas, if there were one million samples, this would bump to 9 999.95 FPs. Hence, testing for the number of FPs received on a purely benign dataset shows the model's true performance under real conditions. Furthermore, this method tested the model's generalisability to new, unseen, benign traffic.

5.5 *Experiment 4:* Varying input feature-length

When inputting data into models, Lotfollahi [23] demonstrated that larger feature spaces can bring greater computational cost. As a result, a common approach to try and reduce the computational cost of ML models is to train the models on a reduced feature set. Weisz [53] demonstrated that when using a decreased feature space, the prediction accuracy decreases. However, this decrease results in more packets being predicted per second.

All measurements were done under the same environment and conditions for inference time and runtime memory usage metrics. The experiment was run on a 64-bit operating system (OS) with an 11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz 2.80 GHz processor and 16 GB of RAM. There were no applications open other than a terminal.

6 Results and Discussion

6.1 Model Comparison

Table 2 presents the percentage accuracy, FPs, FNs, average inference time and average runtime memory usage obtained by the binary classifier models when detecting benign/ransomware traffic on the test set. The average values across ten inferences were obtained for inference time and runtime memory usage.

6.1.1 Accuracy and False Positives for Ransomware Detection. The LSTM model is shown to obtain the highest accuracy. We theorise that this is due to the LSTM's ability to handle sequential network traffic, learn long-term dependencies, extract features and deal with noisy data, supporting the finding done in previous work by Vinayakumar et al.[49]. This was expected due to the LSTM's ability to learn sequential data, such as network traffic. Conversely, the windowed LSTM (WLSTM) performance was not as favourable regarding accuracy and FPs. This outcome suggests that grouping samples into a window diminishes the temporal resolution of the data. This contrasts the expected results, whereby a

[14]https://textbook.cs161.org/network/intrusion-detection.html

Table 2. Percentage accuracy, False Positives, False Negatives, Inference Time and Runtime Memory Usage for each model on the OA dataset

	%Accuracy	False Positives	False Negatives	Inference Time (μs)	Memory Usage (*MiB*)
MLP	99.20	629	229	23	1961
1D CNN	99.18	2	48	34	1972
2D CNN	96.95	0	11353	37	1993
LSTM	99.85	530	19	89	1745
WLSTM	98.79	1054	75	26	6379

windowed approach would increase the temporal resolution [36]. The 2D CNN obtained the worst accuracy. However, it obtained the lowest number of FPs, along with the 1D CNN. This indicates that the benign network traffic may exhibit more spatial patterns than the CNNs can capture. Still, ransomware traffic may contain complex patterns that the CNNs cannot capture, thereby agreeing with previous work showing the dependencies within ransomware traffic. This would result in the CNNs being able to obtain fewer FPs but not performing well when classifying ransomware traffic. This is further shown by looking at the FNs obtained by the 2D CNN, meaning that it could not classify much of the ransomware traffic. Notably, the MLP obtained a higher accuracy than both CNN models but reported many FPs. The results also suggest that the imbalanced dataset may have affected the learning of the models. With more benign network traffic, the models may be able to generalise better on the benign traffic and not so well on ransomware traffic, explaining the lower accuracies but better false positive return rate. This also indicates that the LSTM may handle this class imbalance better.

6.1.2 Inference Time and Runtime Memory Usage for Ransomware Detection. The MLP achieves the fasted average inference time of 0.0023 *milliseconds*, and the LSTM is the slowest with 0.0089 *milliseconds*. This aligns with previous research demonstrating an MLP as a more lightweight model choice. We hypothesize that this is attributed to the simpler architecture of MLPs and their utilization of matrix multiplications, which are inherently faster than the calculations in LSTMs or CNNs. Both CNN models achieve similar inference times, having a difference of 0.0003 *milliseconds*. However, CNNs might exhibit swifter inference times than LSTMs, owing to their capability to parallelize convolutions and pooling operations. The LSTM achieves the lowest runtime memory usage, with all other models obtaining similar values. We also hypothesize that this is because the LSTM processes data sequences step by step, holding less memory. However, the WLSTM exhibits diminished memory efficiency during runtime, implying that each window of samples needs increased memory resources for data storage and processing. This trade-off in memory efficiency is a notable advantage regarding faster inference times. All models achieve an acceptable inference time of below 25.74 milliseconds, as discussed in Section 4.2.4.

6.2 False positives

Table 3. False Positives obtained on six-day dataset

	Total	False Positives per 100 records
MLP	87549	1.6381
1D CNN	638	0.0119
2D CNN	216345	4.0479
LSTM	213	0.0040
WLSTM	1479	0.0277

Table 3 presents the FPs and FPs per 100 records that each binary classifier model obtained over six days when classifying benign traffic. The dataset used to obtain these results contains only benign traffic, categorized across the six distinct days during which the network traffic was recorded. The table displays then shows the number of FPs obtained per 100 records. This was obtained by dividing the FPs accumulated over the six days by the total number of samples in the six traffic days, and then further multiplying this to get the FPs per 100 records.

The LSTM and 1D CNN outperform MLP and 2D CNN in their ability to minimise the number of false positives they return. This contradicts the results in Section 6.1, where the 2D CNN obtained the lowest FPs. This could be due to the 2D CNN being overfitted to the test set but not generalizing well to new data. The success of the LSTM and 1D CNN aligns with previous work, showing their success in working with network traffic [20] [23]. The WLSTM obtained low FPs, indicating that it can identify meaningful relationships within benign traffic. This contrasts with the previous results, where the WLSTM obtained many FPs on the test set. We hypothesise that the WLSTM model is more sensitive to the test set, which includes both ransomware and benign traffic, while the additional test set only comprises benign traffic.

Whilst the accuracies of the models never differ by more than 2.5%, as seen in Section 6.1, and while the results show low values for FPs per record, Section 5.4 discusses how small changes in these values can result in many more FPs returned. This is likely due to the vast amounts of samples present in network traffic. Even relatively small networks can generate several terabytes of data daily [48].

As seen in Table 3, the LSTM and MLP obtain significantly fewer FPs than the CNN models, with the LSTM achieving the lowest FPs per record. We speculate that this results from both benign and ransomware traffic exhibiting temporal patterns and long-term dependencies [31] that the LSTM can capture, therefore agreeing with previous work and motivating its use in ransomware detection. However, the WLSTM accumulates more FPs per record than the MLP and LSTM, indicating that the windowed approach diminishes the model's ability to learn patterns and relationships. This further agrees with the discussion in Section 6.1.

To answer the first research objective, the results show that there isn't a universally "perfect" model for detecting ransomware within encrypted network traffic. Instead, the choice of model should be tailored to the network's specific requirements. For instance, if an organization's security team aims to minimize FPs and computational resources are not a constraint, a model with low FPR but high runtime memory usage could be the preferred option.

To answer the second research objective, our findings indicate that employing a window of samples in conjunction with an LSTM, where four samples are grouped as one, does not yield favourable outcomes regarding ransomware classification.

6.3 Varied Input Length

Figure 2a presents how the evaluation accuracy for each model changes based on how many features are inputted to each model. The number of features used for this experiment is 5, 15, 20 and 30. The depicted trend in the figure demonstrates that when the maximum features are inputted, all models obtain high accuracies. This is not the case when fewer features are used. Whilst there are changes in model accuracy across the different-sized inputs, there is no consistent trend across all models. This does not align with previous work, whereby more features resulted in greater accuracies [53]. This indicates that feature redundancy or irrelevance might exist for the dataset utilized in this study. In such cases, the latter features in the samples might not be as significant as the initial ones. However, on closer inspection, the LSTM consistently maintains high accuracy across all features. This indicates that the LSTM can adapt well to the various levels of input information.

Figure 2b depicts the FPs accumulated per 100 records by each model concerning the number of features used for model training. These FPs were obtained from the additional benign test set. Observing the figure reveals a "sweet spot" for all models, where the lowest number of false positives occurs when the models are trained with 15 and 20 features. From this, we hypothesize that employing a small number of features, such as 5, may not provide the models with sufficient information for effective learning. Conversely, utilizing all available features might introduce redundancy and confusion in the final portions of the feature sequence, possibly due to overlap and leading to an increased number of false positives. To investigate this further, future work could include random sample selection experiments. These experiments would help determine whether the training success of the models depends more on the number of features utilized or the specific features chosen. The models could also be tested on a different dataset to see if the same patterns occur when inputting a similar amount of features.

On further inspection, the MLP is an outlier in that it returns no false positives when working with only five features. This indicates that a simpler model architecture, such as found in an MLP, is more effective in learning meaningful patterns when working with fewer features.

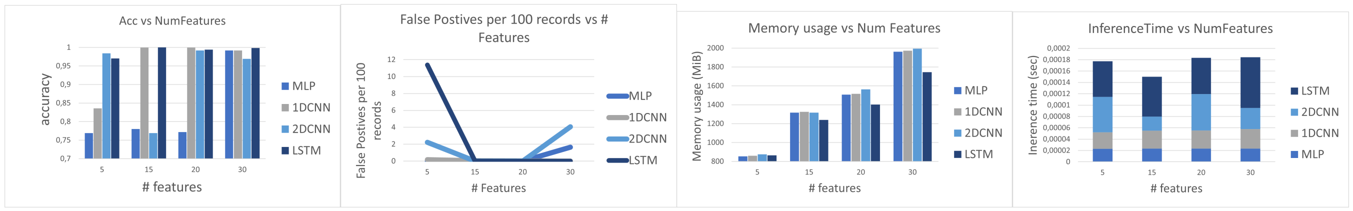
While specific models exhibit promising outcomes in terms of accumulated FPs when trained on fewer samples, it's important to analyze these FPs in conjunction with the accuracies of the models, as presented in Table 6. The 2D CNN, returning 0 FPs for 15 features, only obtains an accuracy of 76.92%. The MLP, obtaining 0 FPs for five features, has an accuracy of 76.92%. Models that return a low amount of false positives but cannot classify ransomware are not useful to the user. The LSTM, however, obtaining 0 FPs on 20 features, still keeps a high accuracy of 99.97%, indicating that the LSTM can still learn the temporal differences and long-term relationships in the data using 20 features.

Table 4. False positives rates across different features

	5 features	**15 features**	**20 features**	**30 features**
MLP	100%	95.25%	98.55%	0.0070%
1D CNN	70.76%	0.013%	0%	0.0023%
2D CNN	0.0097%	100%	100%	0%
LSTM	11.96%	68.55%	0.51%	1.87%

Table 4 present the FPRs for the various models across all feature sets. The FPRs of the models were obtained from the test set, not the additional benign set. According to prior literature [54], an acceptable FPR is considered below 3% while maintaining a classification accuracy above 70%. Similarly to inference time, there is no trend in the FPRs obtained by the models across the different feature sets. Interestingly, the 2D CNN, which accumulates a daily average of 0 FPs for 20 and 15 features, obtains an FPR of 100% for both feature sets. This suggests that the 2D CNN is more sensitive to certain features in the test set that are not present in the additional benign test set, such as ransomware network traffic.

Figure 2c depicts that as more features are input into the models, the runtime memory usage increases consistently across all models. This aligns with previous work [53]. We



(a) Accuracy **(b)** False Positives **(c)** Runtime Memory Usage **(d)** Inference time

Figure 2. Comparison of the model metrics at different numbers of features

speculate that this is due to features adding to the dimensionality of the data, thereby requiring more memory to store and process the data. When considered alongside Figure 2a, this correlation provides additional motivation to assess the models' success beyond sole accuracy. For instance, delving into the model's memory usage can yield more profound insights when observing scenarios where accuracy remains unchanged.

Figure 2d shows that the change in inference time for most models across different features is not as prominent, with the biggest change observed being only 37 μs. This indicates that the number of features might not as strongly influence inference time. We postulate that this is due to the data being processed in the same batch sizes and the memory access patterns of the models being similar. However, there is a clear distinction between the models having different inference times, with the LSTM having the greatest inference times across all features and the MLP the least. This could result from the complex architecture of the LSTM requiring greater computations to make an inference and the simpler MLP architecture requiring less.

The ratio of feature change to the alteration in other metrics is worth noting. The transition from 30 features to 20 marks a 33.33% difference in input size. Similarly, utilizing the minimum number of features leads to an 83.33% reduction in the number of features employed. While these are large changes in the feature space, the runtime memory usage and inference time do not undergo such large changes. The greatest change in runtime memory usage due to a decreased input is 1171.26 MiB, and for inference time, 39 μs. This suggests that despite its model type, each model has an inference time and runtime memory usage overhead, regardless of the number of features inputted into the model.

For the third research objective, our findings indicate that training models with inputs of varying sizes impact accuracy, false positives (FPs), and runtime memory usage, with the 20-feature LSTM model showing the best results across these metrics. However, The LSTM models achieve the slowest inference time and MLPs the fastest. Despite this, there is no significant effect on inference time when using inputs of various sizes. This suggests that dimensionality reduction does not have a large influence on the inference time of DL models for ransomware classification. Accuracy trends are not consistently influenced by reducing the number of features across all models; tailored feature sets are crucial for optimal accuracy. Notably, the LSTM consistently performs well across all input sizes, providing evidence that suggests the data in network packets is sequential. The optimal range for minimizing FPs accumulated is using the first 50% to 66% of the entire feature set. The results also show that larger inputs lead to increased runtime memory usage.

7 Conclusions

In conclusion, this study evaluated the performance of various ML models for classifying encrypted network traffic as ransomware.

We found that there is no universally perfect model, and the choice should align with specific network requirements. Additionally, we observed that the number of features inputted into a model affects model performance, with the 20-feature LSTM model consistently performing well. Dimensionality reduction did not greatly impact inference time but increased runtime memory usage. Lastly, combining a sliding window of inputs with an LSTM did not yield favourable results. Our findings emphasize the need for tailored model selection, optimized input sizes, and carefully designed feature sets for effective ransomware detection in encrypted network traffic.

For future work, network traffic could be collected from multiple networks to make the models more generalisable to different network environments. Secondly, to further investigate dimensionality reduction, rather than exclusively selecting the first x features, future studies could involve randomly selecting features to assess the significance of feature quantity versus feature selection. Extending dimensionality reduction experiments to different network traffic datasets can also test whether similar effects manifest across various datasets. To reduce the inaccuracies when measuring runtime memory usage and inference time, future work could further include measuring these metrics in a more isolated environment, where these metrics can easily be controlled.

References

[1] Aceto, G., Ciuonzo, D., Montieri, A., and Pescapé, A. Mobile encrypted traffic classification using deep learning: Experimental evaluation, lessons learned, and challenges. *IEEE Transactions on Network and Service Management 16*, 2 (2019), 445–458.

[2] Beaman, C., Barkworth, A., Akande, T. D., Hakak, S., and Khan, M. K. Ransomware: Recent advances, analysis, challenges and future research directions. *Computers Security 111* (2021), 102490.

[3] Ben Abdel Ouahab, I., Elaachak, L., and Bouhorma, M. Image-based malware classification using multi-layer perceptron. In *Networking, Intelligent Systems and Security: Proceedings of NISS 2021* (2022), Springer, pp. 453–464.

[4] Berrueta, E., Morato, D., Magaña, E., and Izal, M. Open repository for the evaluation of ransomware detection tools. *IEEE Access 8* (2020), 65658–65669.

[5] Brewer, R. Ransomware attacks: detection, prevention and cure. *Network Security 2016*, 9 (2016), 5–9.

[6] Bu, Z., Zhou, B., Cheng, P., Zhang, K., and Ling, Z.-H. Encrypted network traffic classification using deep and parallel network-in-network models. *Ieee Access 8* (2020), 132950–132959.

[7] Chen, Z., He, K., Li, J., and Geng, Y. Seq2img: A sequence-to-image based approach towards ip traffic classification using convolutional neural networks. In *2017 IEEE International conference on big data (big data)* (2017), IEEE, pp. 1271–1276.

[8] Cheng, Q., Wu, C., Zhou, H., Kong, D., Zhang, D., Xing, J., and Ruan, W. Machine learning based malicious payload identification in software-defined networking. *Journal of Network and Computer Applications 192* (2021), 103186.

[9] Courtaud, C., Sopena, J., Muller, G., and Gracia Pérez, D. Improving prediction accuracy of memory interferences for multicore platforms. In *2019 IEEE Real-Time Systems Symposium (RTSS)* (2019), pp. 246–259.

[10] Dicks, M., Tooke, J., and Weisz, S. A comparative evaluation of deep learning approaches to online network traffic classification for community networks.

[11] Firdausi, I., Erwin, A., Nugroho, A. S., et al. Analysis of machine learning techniques used in behavior-based malware detection. In *2010 second international conference on advances in computing, control, and telecommunication technologies* (2010), IEEE, pp. 201–203.

[12] Fukushima, Y., Sakai, A., Hori, Y., and Sakurai, K. A behavior based malware detection scheme for avoiding false positive. In *2010 6th IEEE Workshop on Secure Network Protocols* (2010), pp. 79–84.

[13] Gandotra, E., Bansal, D., and Sofat, S. Zero-day malware detection. In *2016 Sixth International Symposium on Embedded Computing and System Design (ISED)* (2016), pp. 171–175.

[14] Gao, Y., Gu, X., Zhang, H., Lin, H., and Yang, M. Runtime performance prediction for deep learning models with graph neural network. In *2023 IEEE/ACM 45th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)* (2023), pp. 368–380.

[15] Gardner, M., and Dorling, S. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric Environment 32*, 14 (1998), 2627–2636.

[16] Gibert, D., Mateu, C., and Planes, J. The rise of machine learning for detection and classification of malware: Research developments, trends and challenges. *Journal of Network and Computer Applications 153* (2020), 102526.

[17] Gicić, A., and Subasi, A. Credit scoring for a microcredit data set using the synthetic minority oversampling technique and ensemble classifiers. *Expert Systems 36*, 2 (2019), e12363.

[18] Goodfellow, I., Bengio, Y., and Courville, A. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[19] Hochreiter, S., and Schmidhuber, J. Long short-term memory. *Neural computation 9*, 8 (1997), 1735–1780.

[20] Homayoun, S., Dehghantanha, A., Ahmadzadeh, M., Hashemi, S., Khayami, R., Choo, K.-K. R., and Newton, D. E. Drthis: Deep ransomware threat hunting and intelligence system at the fog layer. *Future Generation Computer Systems 90* (2019), 94–104.

[21] Joshi, S., Owens, J. A., Shah, S., and Munasinghe, T. Analysis of preprocessing techniques, keras tuner, and transfer learning on cloud street image data. In *2021 IEEE International Conference on Big Data (Big Data)* (2021), pp. 4165–4168.

[22] Kumar, S., Dharmapurikar, S., Yu, F., Crowley, P., and Turner, J. Algorithms to accelerate multiple regular expressions matching for deep packet inspection. *ACM SIGCOMM computer communication review 36*, 4 (2006), 339–350.

[23] Lotfollahi, M., Jafari Siavoshani, M., Shirali Hossein Zade, R., and Saberian, M. Deep packet: A novel approach for encrypted traffic classification using deep learning. *Soft Computing 24*, 3 (2020), 1999–2012.

[24] Maniath, S., Ashok, A., Poornachandran, P., Sujadevi, V., AU, P. S., and Jan, S. Deep learning lstm based ransomware detection. In *2017 Recent Developments in Control, Automation & Power Engineering (RDCAPE)* (2017), IEEE, pp. 442–446.

[25] Marqués, A. I., García, V., and Sánchez, J. S. On the suitability of resampling techniques for the class imbalance problem in credit scoring. *Journal of the Operational Research Society 64*, 7 (2013), 1060–1070.

[26] McGaughey, D., Semeniuk, T., Smith, R., and Knight, S. A systematic approach of feature selection for encrypted network traffic classification. In *2018 Annual IEEE International Systems Conference (SysCon)* (2018), IEEE, pp. 1–8.

[27] Mell, P., Hu, V., Lippmann, R., Haines, J., and Zissman, M. An overview of issues in testing intrusion detection systems.

[28] Menghani, G. Efficient deep learning: A survey on making deep learning models smaller, faster, and better. *ACM Computing Surveys 55*, 12 (2023), 1–37.

[29] Modi, J., Traore, I., Ghaleb, A., Ganame, K., and Ahmed, S. Detecting ransomware in encrypted web traffic. In *Foundations and Practice of Security: 12th International Symposium, FPS 2019, Toulouse, France, November 5–7, 2019, Revised Selected Papers 12* (2020), Springer, pp. 345–353.

[30] Mohurle, S., and Patil, M. A brief study of wannacry threat: Ransomware attack 2017. *International journal of advanced research in computer science 8*, 5 (2017), 1938–1940.

[31] Morato Oses, D., Berrueta, E., Magaña, E., and Izal, M. A chronological evolution model for crypto-ransomware detection based on encrypted file-sharing traffic. *Available at SSRN 4074557*.

[32] Mott, G., Turner, S., Nurse, J. R., MacColl, J., Sullivan, J., Cartwright, A., and Cartwright, E. Between a rock and a hard (ening) place: Cyber insurance in the ransomware era. *Computers & Security 128* (2023), 103162.

[33] Nguyen, A. T., Raff, E., Nicholas, C., and Holt, J. Leveraging uncertainty for improved static malware detection under extreme false positive constraints. *arXiv preprint arXiv:2108.04081* (2021).

[34] O'Gorman, G., and McDonald, G. *Ransomware: A growing menace.* Symantec Corporation Arizona, AZ, USA, 2012.

[35] O'Kane, P., Sezer, S., and Carlin, D. Evolution of ransomware. *Iet Networks 7*, 5 (2018), 321–327.

[36] Oord, A. v. d., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499* (2016).

[37] O'Shea, K., and Nash, R. An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458* (2015).

[38] O'Shea, K., and Nash, R. An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458* (2015).

[39] Papadogiannaki, E., and Ioannidis, S. A survey on encrypted network traffic analysis applications, techniques, and countermeasures.

ACM Computing Surveys (CSUR) 54, 6 (2021), 1–35.

[40] Papadogiannaki, E., Tsirantonakis, G., and Ioannidis, S. Network intrusion detection in encrypted traffic. In *2022 IEEE Conference on Dependable and Secure Computing (DSC)* (2022), pp. 1–8.

[41] Park, J., Naumov, M., Basu, P., Deng, S., Kalaiah, A., Khudia, D., Law, J., Malani, P., Malevich, A., Nadathur, S., et al. Deep learning inference in facebook data centers: Characterization, performance optimizations and hardware implications. *arXiv preprint arXiv:1811.09886* (2018).

[42] Rezaei, S., and Liu, X. Deep learning for encrypted traffic classification: An overview. *IEEE communications magazine 57*, 5 (2019), 76–81.

[43] Richardson, R., and North, M. M. Ransomware: Evolution, mitigation and prevention. *International Management Review 13*, 1 (2017), 10.

[44] Rustam, Z., Utami, D. A., Hidayat, R., Pandelaki, J., and Nugroho, W. A. Hybrid preprocessing method for support vector machine for classification of imbalanced cerebral infarction datasets. *International Journal on Advanced Science Engineering Information Technology 9*, 2 (2019), 685–691.

[45] Shen, M., Ye, K., Liu, X., Zhu, L., Kang, J., Yu, S., Li, Q., and Xu, K. Machine learning-powered encrypted network traffic analysis: a comprehensive survey. *IEEE Communications Surveys & Tutorials* (2022).

[46] Sherry, J., Lan, C., Popa, R. A., and Ratnasamy, S. Blindbox: Deep packet inspection over encrypted traffic. In *Proceedings of the 2015 ACM conference on special interest group on data communication* (2015), pp. 213–226.

[47] Singh, J., and Singh, J. Malware classification using multi-layer perceptron model. In *International Conference on Innovative Computing and Communications: Proceedings of ICICC 2020, Volume 2* (2021), Springer, pp. 155–168.

[48] Villa, A., and Varki, E. Characterization of a campus internet workload. In *Proceedings of CATA* (2012), pp. 140–148.

[49] Vinayakumar, R., Soman, K. P., and Poornachandran, P. Applying deep learning approaches for network traffic prediction. In *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)* (2017), pp. 2353–2358.

[50] Wang, W., Zhu, M., Wang, J., Zeng, X., and Yang, Z. End-to-end encrypted traffic classification with one-dimensional convolution neural networks. In *2017 IEEE international conference on intelligence and security informatics (ISI)* (2017), IEEE, pp. 43–48.

[51] Wang, X., Zhao, F., Lin, P., and Chen, Y. Evaluating computing performance of deep neural network models with different backbones on iot-based edge and cloud platforms. *Internet of Things 20* (2022), 100609.

[52] Wang, Z., Fok, K. W., and Thing, V. L. Machine learning for encrypted malicious traffic detection: Approaches, datasets and comparative study. *Computers & Security 113* (2022), 102542.

[53] Weisz, S. Network tra ic classification using two-dimensional convolutional neural networks for community networks.

[54] Yang, C., Xu, J., Liang, S., Wu, Y., Wen, Y., Zhang, B., and Meng, D. Deepmal: maliciousness-preserving adversarial instruction learning against static malware detection. *Cybersecurity 4* (2021), 1–14.

[55] Yewale, A., and Singh, M. Malware detection based on opcode frequency. In *2016 International Conference on Advanced Communication Control and Computing Technologies (ICACCCT)* (2016), pp. 646–649.

[56] Zeng, Y., Gu, H., Wei, W., and Guo, Y. $deep-full-range$: a deep learning based network encrypted traffic classification and intrusion detection framework. *IEEE Access 7* (2019), 45182–45190.

[57] Zheng, W., Zhong, J., Zhang, Q., and Zhao, G. Mtt: an efficient model for encrypted network traffic classification using multi-task transformer. *Applied Intelligence 52*, 9 (2022), 10741–10756.

[58] Zhu, Z., and Dumitraş, T. Featuresmith: Automatically engineering features for malware detection by mining the security literature. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (2016), pp. 767–778.

A Architectures

Table 5. Model Architectures

Model	Layer Type	# Neurons	Activation
MLP (lr=0.0001)	Input	30	-
	Dense	168	ReLu
	Dropout	0.35	-
	Dense	168	ReLu
	Dropout	0.35	-
	Dense	1	Sigmoid
1D CNN (lr=0.001)	Input	30	-
	Conv1D	104	Tanh
	MaxPool1D	2	-
	Dropout	0.5	-
	Flatten	-	-
	Dense	72	Tanh
	Dropout	0.5	-
	Dense	1	Sigmoid
2D CNN (lr=0.01)	Input	30	-
	Conv2D	8	ReLU
	MaxPool2D	2	-
	Dropout	0.25	ReLU
	Flattern	-	-
	Dense	136	ReLU
	Dropout	0.25	-
	Dense	1	Sigmoid
LSTM (lr=0.001)	Input	30	-
	LSTM	8	ReLu
	Dropout	0.5	-
	LSTM	8	ReLu
	Dropout	0.5	-
	Dense	1	Sigmoid

B Metric results

Table 6. # Features, Accuracy and False Positives

Model	# Features	Accuracy	False Positives per 100 records
MLP	30	0.9920	1.6381
	20	0.7717	0.0002
	15	0.7802	0.0011
	5	0.7692	0
1D CNN	30	0.9918	0.0119
	20	0.9998	0.0193
	15	0.9996	0.0388
	5	0.8359	0.1716
2D CNN	30	0.9695	4.0479
	20	0.9992	0
	15	0.7692	0
	5	0.9840	2.21589
LSTM	30	0.9985	0.0040
	20	0.9987	0
	15	0.9999	0.0089
	5	0.9705	11.3870

Table 7. Inf time and Memory usage for different features

Model	# features	Inf Time (μ *s*)	Mem Usage (*MiB*)
MLP	30	23	1960.88
	20	23	1507.25
	15	23	1316.78
	5	23	854.3132
1DCNN	30	34	1971.89
	20	32	1516.15
	15	32	1324.64
	5	29	859.188
2DCNN	30	37	1992.802
	20	64	1563.86
	15	25	1317.04
	5	62	875.542
LSTM	30	89	1745.03
	20	64	1402.74
	15	7	1239.03
	5	63	863.69

C Ransomware Families

Table 8. Ransomware Families and Year of most recent sample

Ransomware Families		
No.	**Family**	**Year of most recent sample**
1	Aleta	2017
2	Bart	2016
3	BitPaymer	2018
4	Cerber	2021
5	CryLock	2021
6	CrypMIC	2016
7	CrypFile2	2016
8	CryptoFortress	2015
9	CryptoMix	2016
10	CryptoShield	2017
11	Crysus	2021
12	Cryxox	2018
13	CTBLocker	2017
14	Dharma	2017
15	Diablo	2017
16	DMALocker	2016
17	Eris	2019
18	GrandCrab	2019
19	GlobeImposter	2017
20	Jaff	2017
21	Locky	2016
22	MakTub	2018
23	Maze	2019
24	Mole	2018
25	MRCR	2017
26	Netwalker	2021
27	Odin	2016
28	Phobos	2019
29	RansomX	2020
30	Razi	2021
31	Revenge	2018
32	Ryuk	2021
33	Sage	2018
34	Scarab	2019
35	Shade	2021
36	Shaofao	2020
37	Sodinokibi	2021
38	Spora	2017
39	Stop	2019
40	TeslaCrypt	2015
41	VirLock	2017
42	WannaCry	2021
43	Zeus	2017

D Formulas

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

$$FalsePositiveRate = \frac{FP}{FP + TN}$$