Opcodes to Images: A Framework for Early Detection of Ransomware by Utilising Machine Learning Techniques

Grace Forsyth

***Abstract*—Ransomware is a type of malware that is used by attackers to encrypt data on a victim's system and demand a ransom for the key. Ransomware is a devastating problem for individuals and businesses worldwide. The evolving world of technology opens the gates for information to be stolen and destroyed by ransomware, causing massive financial and personal data loss. To mitigate the harmful impact of ransomware, it is crucial to develop solutions that can prevent attacks by detecting them early. The problem with ransomware is that it is often not discovered on a system until it has executed. By then, it is too late to prevent all the damage it causes. This project aims to tackle this problem by developing a Convolutional Neural Network (CNN) trained on images created out of ransomware binaries. The model will be able to classify a file as ransomware, detecting it on its way into a system. The balanced accuracy, classification accuracy and training time was used for benchmarking. We compared this method to another state-of-the-art solution, where the training time was significantly smaller and the accuracy was effectively competitive.**

I. Introduction

RANSOMWARE is a type of malware that is executed on a system and controlled by attackers. When executed, the ransomware traverses the system and encrypts files it finds and demands a ransom for the key [1]. This means that people lose access to data and disrupts business operations. Furthermore, some ransomware will lock down machines, obstructing access until the ransom is paid. However, even if the ransom is paid, the attackers may still have exfiltrated the files they found, and will often sell the information on the dark web. Ransomware is a prominent problem today with the rise of technology, causing devastation to businesses and individuals. To truly grasp the impact of a ransomware attack, one must witness firsthand the aftermath it leaves behind. Every year, millions of people find their personal information up for sale on the dark web as a result of double-tap ransomware. Businesses suffer irreparable damage to their reputation due to ransomware attacks. The financial and personal lives of unsuspecting victims are often shattered, and the effects can be long-lasting.

One of the most challenging aspects of ransomware is its stealthy nature, frequently catching its targets off guard. Without warning or early detection mechanisms in place, victims find themselves at the mercy of ransomware attacks, with limited options for recourse. The relevance of this project to society cannot be overstated. Ransomware attacks have emerged as threat, impacting individuals, businesses, and even critical infrastructure. CERT NZ released that in just three months ransomware reports increased by 500% at the end of 2022 [2]. A ransomware attack on a small IT company at the end of 2022 in New Zealand gave attackers access to a large amount of data from different organisations the company had worked for. Thousands of people's health insurance, business and personal data was found being sold on the dark web for up to $1.58 million, even though the company had paid the very expensive ransom [3]. A recent attack on Latitude resulted in over 7 million Australia and New Zealand customers having to replace drivers licenses and look out for suspicious activity around credit card applications [4]. The two methods for malware analysis are static and dynamic. Dynamic analysis requires the malware to be executed whereas static analysis gains file information without execution. Most existing ransomware detection utilises a form of machine learning to predict malicious activity based upon various aspects of ransomware. These aspects are; signatures of the file, functions the file uses e.g. system API calls, dynamic behaviour observed in a secure environment and the instructions the ransomware gives to the computer - operational codes or opcodes. Dynamic analysis is dangerous because it requires the ransomware to be run and signature analysis requires the ransomware to be known. By utilising static analysis of opcodes, the project seeks to provide a proactive defence against ransomware attacks. To accomplish this goal, the project involves constructing a comprehensive dataset comprised of images generated from the opcodes of both ransomware and benignware (or goodware). Benignware, also referred to as goodware, describes files that are not malicious. By feeding these images into a Convolutional Neural Network (CNN) model, the system should learn to differentiate between benignware and ransomware. This was measured based on a high balanced accuracy, expected to be above 80%, which would show the CNN's ability to use these images for detecting ransomware. The CNN model achieved a high average balanced accuracy of 84.17%, effectively competing with other existing state-of-the-art solutions.

By providing an early detection mechanism, this project offers the potential to minimise the devastating consequences of ransomware. Timely identification of ransomware can allow for prompt action, mitigating the spread of the ransomware and limiting the damage inflicted on victims. Furthermore, it is important to consider the environmental and sustainability aspects related to the problem and the proposed solution. Ransomware attacks not only cause direct harm to individuals and businesses, but also have broader implications to the

This project was supervised by Harith Al-Sahaf (primary) and Shabbir Abbasi (secondary).

environment.

A. Goals

The overall aim of this project is to develop a static-based malware analysis method for the early detection of ransomware through converting file binaries to images and utilising a CNN for binary classification.

The main contributions of this project are:

- Extracting opcodes from ransomware and goodware files to transform into an image.
- Perform preprocessing to enable optimisation for the CNN.
- Construct a CNN that is able to differentiate between the ransomware and goodware files.
- Evaluate the performance against existing methods and analyse the results to provide insights into the model.

B. Benchmarking

We compared the results of this project with another state-of-the-art ransomware classification method; A few-shot meta-learning based Siamese neural network using entropy features [5]. We compared results in terms of balanced accuracy and training times. We used the exact same dataset as this study, therefore the comparisons are more suitable. However, a difference in terms of performance measure is that in this project we performed binary classification, and in the study, they performed multi-class classification.

C. Organisation

The following is an overview of the report's organisation: Section II discusses related work along with existing solutions and also introduces a background on ransomware analysis and machine learning techniques. Section III discusses the design and implementation of this project's proposed solution of using a CNN to classify images as ransomware. Section IV discusses the experimental settings and Section V discusses the results from the project's implemented solution, including the performance metrics mentioned above in I-B. Then, limitations of this paper are discussed in Section VIII and sustainability considerations relating to this project are addressed in Section VII.

II. Background and Related Work

This section will introduce a background in ransomware analysis and machine learning techniques. Then, related works with relevance to this project will be listed and discussed in detail.

A. Related Work

The most widely used ransomware detection for businesses is CrowdStrike [6]. CrowdStrike is an industry-leading cyber-security company that provides a range of services. The most relevant to this report is the malware (including ransomware) detection and prevention service. This utilises multi-level machine learning which combines file analysis, behavioural analysis, and indicators of attack to detect malware and ransomware [7]. Therefore, the method for detection is very large, complicated, and powerful. CrowdStrike released that their models achieved 99.4% malware detection rate in the AV-Comparatives Malware Protection Test with zero false positives [8]. With an accuracy that high, it is clear why they are industry-leading. A downside to the use of CrowdStrike is the cost. CrowdStrike is an expensive service to run, designed for relatively large organisations, and requires constant updates and large databases to keep up with the rapidly evolving new versions of malware.

Researchers have developed several machine learning models based solely on file analysis. Khan et al. proposed DNAact-Ran which involves the selection of important features from preprocessed ransomware and benignware file data using Multi-Objective Grey Wolf Optimisation (MOGWO) and Binary Cuckoo Search (BCS) algorithms [9]. These selected features are then utilised to generate a digital DNA sequence. To train the model, an active learning classification algorithm was employed alongside a linear regression machine learning algorithm to detect the ransomware family, enabling multi-class classification into different ransomware families. The proposed approach demonstrates promising results, with an accuracy rate of 87.9% achieved [9]. The main limitation of this algorithm is the expensive processes used to process, select, and create the DNA sequenced data which is convoluted compared to other processes such as focusing just on opcode frequencies or signatures.

Baldwin and Dehghantanha [10] performed classification of crypto-ransomware and benignware using static analysis. They extracted opcodes from the executables, and represented these as density histograms. They then used Sequential Minimal Optimisation (SMO) training for a support-vector machine learning algorithm implemented with a polynomial kernel to classify opcode data into ransomware or benignware. They achieved 100% accuracy with binary classification and 96.7% accuracy with multi-class classification into crypto-ransomware families. This method of relying on the opcodes required more preprocessing as opcodes like `MOV` and `PUSH` are extremely common in both ransomware and benignware [10].

Another study performed by Khammas [11] focuses on extracting hierarchical features from ransomware families, since each family shares common features. Byte-level static analysis is performed by extracting features directly from raw executable file bytes, using N-gram features. Feature selection is performed using the Gain Ratio method to reduce feature dimensionality and select the most important features. The random forest classifier is used for classification. The classifier combines predictions from multiple decision trees through majority voting. The experiments demonstrate a high accuracy of 97.74%. Overall, the framework achieves effective ransomware attack detection by leveraging static analysis, feature extraction, feature selection, and the random forest classifier [11]. While an effective technique, the preprocessing process of extracting hierarchical features is relatively lengthy.

Zhang et al. [12] collected 1,787 ransomware samples from eight different families as well as 100 benignware

samples. They transformed the opcode sequences into N-gram sequences and selected important N-grams based on Term Frequency —Inverse Document Frequency (TF-IDF) ranking [12]. Using these N-grams, they built classification models using five machine learning algorithms. The experiments involved different N-gram lengths (1, 3, and 4) and varied feature dimensions. The Random Forest algorithm yielded the best results, achieving an accuracy of 91.43%. The F1-measure for the "Wannacry" ransomware family remained consistently high at 99.0%. The binary classification between ransomware and benignware reached an accuracy of 99.3%. The study highlights the effectiveness of using the Random Forest algorithm for accurately classifying ransomware based on opcodes and achieving high detection performance for specific ransomware families [12]. However, there is a high imbalance in the ransomware vs benignware class samples.

In a similar study, Zhang et al. [13] collected a dataset of ransomware samples from different families and trusted software samples. They extracted features from the opcode sequences of the ransomware samples using static analysis. The features are also ranked using TF-IDF, and selected N-grams are used to build classification models using machine learning algorithms such as SA-CNN, Naïve Bayes and k-Nearest Neighbour (k-NN). To address the challenges of handling long opcode sequences, the authors propose a framework based on deep learning and self-attention mechanisms. The opcode sequences are partitioned, and a Self-Attention based Convolutional Neural Network (SA-CNN) is constructed to capture long-range dependencies. The outputs of SA-CNNs are concatenated to form sequential intermediate features, which are further processed using a bi-directional self-attention network for classification. The average F1-score, which was used to measure the model, was 0.873. This was the highest compared to other machine learning algorithms such as Naïve Bayes and k-NN [13].

Another paper proposed a state-of-the-art solution for ransomware classification using entropy features [5]. The paper proposes the approach of using deep learning techniques to detect and classify different ransomware classes. A key obstacle they tackle involves the complexity of crafting a proficient deep-learning solution, primarily arising from the scarcity of training samples and the bias inherent in models trained with limited data. Therefore, the paper proposes a few-shot-meta-learning based Siamese Neural Network, which utilises the entropy feature directly obtained from ransomware files for classification in comparison to images. The proposed approach generates significantly more accurate weight factors as compared to a model with a limited number of training values. The authors evaluate their proposed model on a ransomware dataset containing 11 different classes and benchmark it with various deep learning models. Their experimental results show that their proposed model is highly effective, providing a weighted F1-score of over 86 percent. The authors acknowledge, however, that their model is significantly more complex than other models and therefore in this project we had to consider the limitations this imposed such as training times and higher computational costs [5].

B. Ransomware Analysis

As mentioned previously, the two types of malware and ransomware analysis are static and dynamic. Static analysis of ransomware involves examining the ransomware without actively running it. This is the safest and easiest type of ransomware analysis. Using tools, analysts can examine the imports, strings and APIs malware uses, which hints at the functionality. Like any executable file, ransomware at a low level is constructed from a set of instructions telling the system what to do when it is run. These instructions are known as operational codes or opcodes. Static analysis enables us to examine these opcodes and from there, we can make an informed decision of what the ransomware does without running it. Dynamic analysis of ransomware involves the examination and observation of its behaviour in a controlled environment. It typically involves running the ransomware sample in a secure and isolated environment, such as a virtual machine, to analyse its interaction with the system, file modifications, network communications, encryption processes, and any other malicious activities. Dynamic analysis often requires more resources, time, and expertise to set up and conduct experiments in a controlled environment.

This project aims to address the challenge of detecting ransomware, by developing a machine learning model that can detect ransomware at an early stage, by using static analysis techniques before it has a chance to run on a system. The idea of using machine learning to detect ransomware early in this project stems from the fact that many ransomwares perform similar tasks such as traversing through file systems and encrypting files, so they will also have similar opcodes. Using these opcodes gained from static analysis, we can effectively create a deep learning model that will be able to classify a file as ransomware or benignware (free of ransomware).

C. Machine Learning Techniques

There are many Machine Learning (ML) techniques that are designed for different purposes and datasets. Supervised learning is where the ML model learns from labelled training data. It is well known for specific tasks such as classification and regression where the target output is known during training. Unsupervised learning involves training a model on unlabelled data to discover patterns, structures or relationships. Clustering algorithms such as K-Means or DBSCAN are often used in unsupervised learning of sequential data. Sequential data has dependencies or relationships within the data, such as text and speech. Semi-supervised learning combines labelled and unlabelled data to train an ML model. This aims to use the unlablled data to improve the model's performance. RNN/LSTM as well as Multilayer Perceptron (MLP) neural networks can be used in semi-supervised learning. A Multilayer Perceptron (MLP) is a type of feedforward artificial neural network (ANN) that consists of multiple layers of interconnected nodes, also known as neurons. MLPs are versatile and can be applied to various tasks including classification, regression, pattern recognition, and function approximation. Reinforcement learning involves learning through interactions with an environment and receiving feedback in the form of

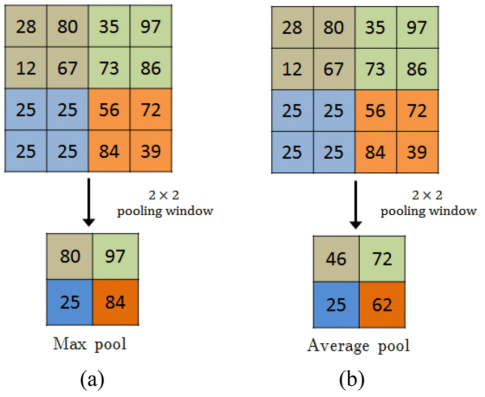

Fig. 1. Illustration of max pooling and average pooling [14].

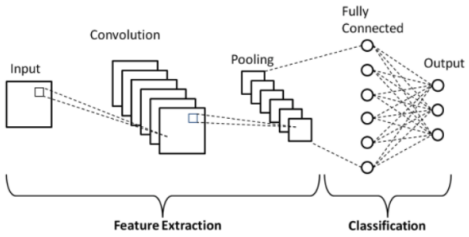

Fig. 2. CNN basic visualisation [15].

rewards or penalties. This technique is useful for training autonomous vehicles to make decisions, such as lane changes for example.

The algorithm chosen to classify opcode images in this project is based on utilising CNNs for image classification. A CNN is best for image data due to the layers and filters it uses to detect edges as well as its ability to perform well with large amounts of data. The key components of a CNN include convolutional layers, pooling layers, fully connected layers, and activation functions. Commonly, the input image is preprocessed and then passed through a convolutional layer. This layer is responsible for applying filters, or kernels, to the image to extract features by using elemental-wise multiplication and summing. An activation function ReLU (Rectified Linear Unit) is applied elementwise to introduce non-linearity into the network. Afterwards, it is passed through a pooling layer. The pooling layer downsizes the data by performing average or maximum calculations on it, as represented in Fig. 1.

This process is repeated however many times suitable for the feature extraction before the image data is "flattened". This fully connects the layers by performing a matrix multiplication between the flattened input and a set of learnable weights, followed by the ReLU activation function again. Next, it is passed to an output layer, which uses the Sigmoid activation function to classify the image. The network is then trained using gradient descent optimisation methods, where the weights of the network are adjusted based on the calculated gradients, therefore minimising a defined loss function. A basic visual overview of the steps is presented in Fig. 2.

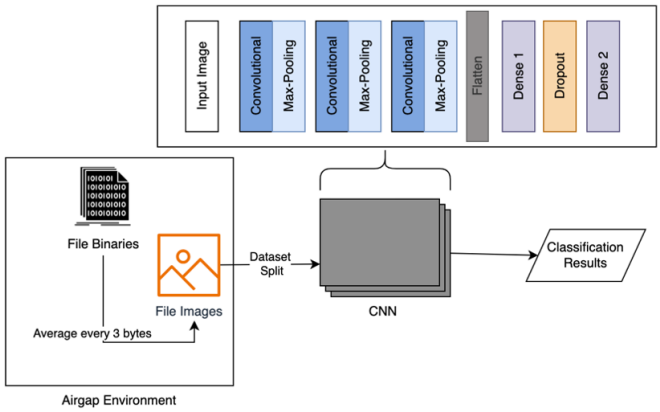

Fig. 3. The overall algorithm of the proposed method.

III. Design and Implementation

Fig. 3 shows the overall process of the proposed method in this project. Ransomware and goodware binaries are taken and transformed into images. This creates the dataset, which is then split into training, validation and test sets. They are subsequently passed through a CNN, with the classification results being outputted where they can then be analysed. Refer to the below sections for an in-depth explanation of each step in this process.

A. Dataset Creation

The dataset for the early detection of ransomware CNN model was comprised of images generated from the opcodes of a collection of ransomware and goodware samples. There were 756 ransomware files and 337 goodware files. By extracting the binaries of the files, each byte can be assigned a value between 0 and 255. These are then transformed into pixel intensities to form an image representation. This process enables the conversion of each ransomware and benignware sample into an image, forming the foundation of the dataset. Originally, the project involved the creation of four types of images that could be used for the model. To craft the images, Python was employed with standard file reading and image creation libraries. Opcodes were read as the file binaries, and each byte converted to a pixel value between 0 and 255. The first type of image generated was a Portable Grey Map (pgm) image where each pixel is created out of the average of every. This process is represented in Fig. 4.

Averaging three values gives decimal points. To fix this issue, the values are rounded to the nearest integer. Fig. 5 contains an image created by averaging every three bytes of a benignware sample to form pixels.

The next three images created from the file samples are multi-channel pgm images based on the equation of binary-to-pixel conversion below.

Assume the binary values are: a1a2a3b1b2b3c1c2c3d1d2d3
Image 1: a1b1c1d1
Image 2: a2b2c2d2
Image 3: a3b3c3d3

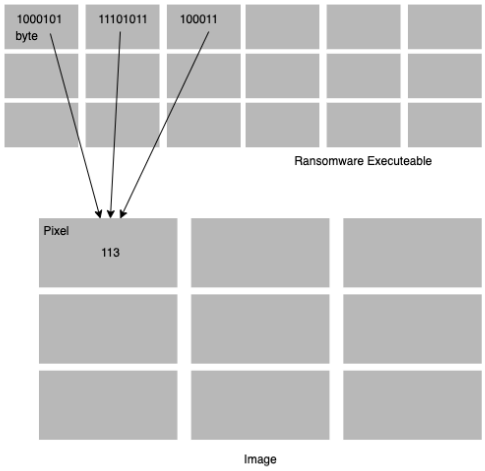

Fig. 4. Binary to image conversion.

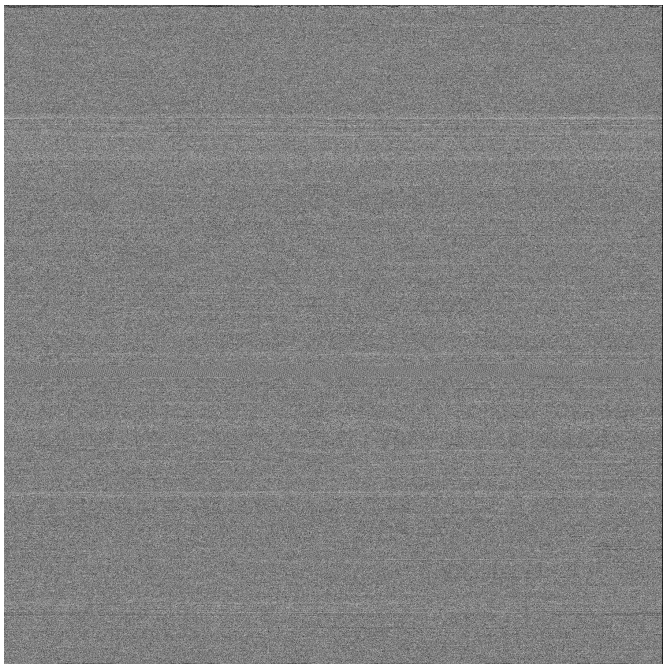

Fig. 5. Average of every three bytes image.

These three images serve as three channels of an RGB image. An example of these images created from a benignware sample can be found in Fig. 6.

We chose to use the average of every three bytes to construct the dataset for the CNN as this can be a good representation of opcodes. Opcodes have different number of bytes per operation, so ensuring proper representation of each opcode is impossible.

An interesting issue encountered when creating the images occurred when the code was initially not reading enough data, and seemed to predominantly pick up on resource sections of the file. This made the images pick up on the file icons stored in the file's resource section and display the icons. An example of this is provided in Fig. 7, where a Compact Disk (CD) logo can be clearly seen across the bottom of the image.

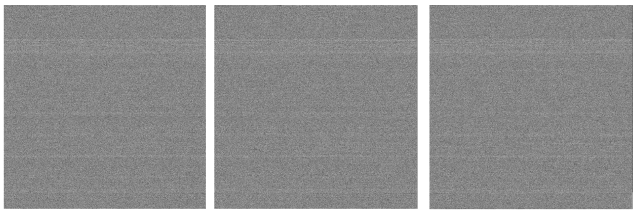

Fig. 6. Image types 1, 2 & 3 of multi-channel images.

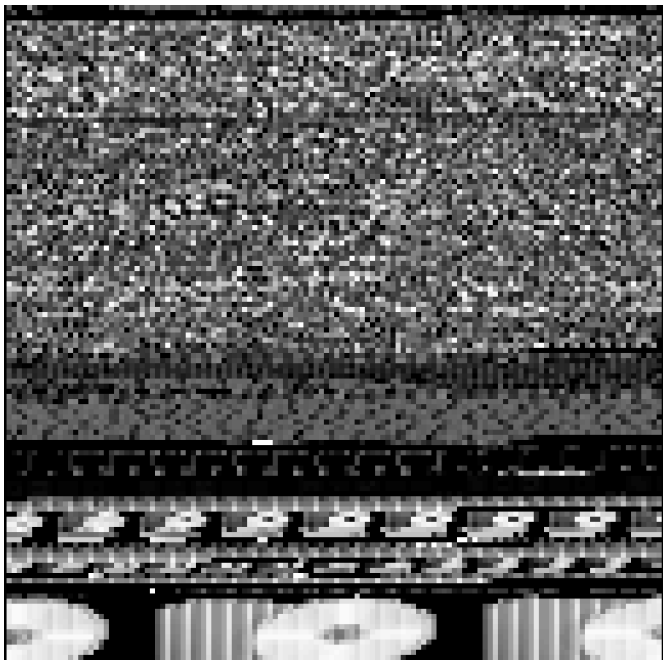

Fig. 7. Image showing CD icon.

This highlights the importance of reading a large enough amount of data so that the header is not the only part of the file included in the image. An obfuscation technique of malware is displaying itself as a common file type such as MP3. Other standard benign executable files will also store icons within the resource section of a file so this is not a relevant feature for the distinction between ransomware and benign files.

The script for this file-to-image conversion process was constructed using Python. Python was chosen because of its built-in file-reading capabilities and ease of use.

1) Environment: Because ransomware is a malicious program, safety measures had to be considered during the creation of the images. A safe environment was constructed by transferring the zipped and password-protected ransomware samples onto a Windows 7 Virtual Machine (VM). The VM was then disconnected from the network, air gapping it as per the project's safety plan. This precautionary measure guaranteed that the ransomware was unable to cause any damage once unzipped and can be seen in the system overview in Fig. 3. Windows Defender had to be disabled on the VM as it would quarantine the ransomware files during the conversion to images. Consequently, the dataset of images was then deemed safe for use on the ECS machines. The

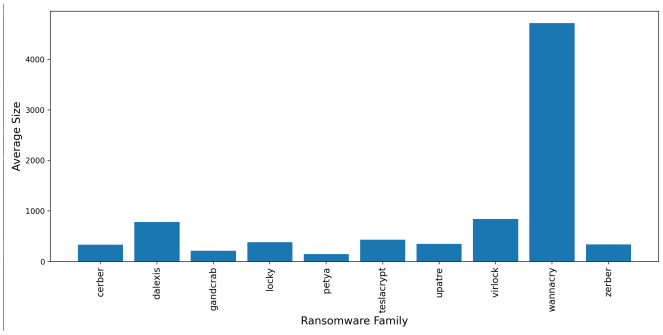

Fig. 8. Ransomware file sizes.

goodware images did not need to be created in an air gapped environment as they were not malicious.

2) Preprocessing: To preprocess the images, the main challenged encountered was the difference in ransomware and goodware file sizes. The CNN model, which will be discussed in the next section, requires images of the same sizes. To achieve this, the lowest file size of 70KB was considered. Therefore we read that amount of data for every file to create the pgm images. This meant that every image was constructed out of the same amount of pixels (and therefore bytes). The average file sizes can be seen documented in Fig. 8.

We also chose to scale the pixel values to a range between [0–63] for the images. While scaling pixel values to a range of [0–1] is a common practice, in this specific context, it was not preferred. This was driven by the assumption that using a [0–1] range may not effectively capture the variation in the data. The idea of the pixel scaling was to help in reducing the computational requirements, as well as memory usage, during training and inference. Some neural networks may converge faster or perform better when dealing with data in a smaller range. Scaling the values to 0-63 helps the network's weights and biases adjust more efficiently during training [16], [17]. This is advantageous when working with limited computational resources or memory constraints. A smaller range can also lead to a simpler model, which can be beneficial in situations where model complexity needs to be managed. It can also help reduce overfitting, especially in cases where the original data may contain more details than necessary (e.g. standard operational codes like MOV and JMP that are very frequent in most executable files) [16], [17].

B. CNN Model

For the model, the dataset is partitioned into three sets for training, validation, and testing purposes. The training and validation datasets are employed during model training, with supervised learning utilising the labelled images (ransomware and benignware) to evaluate the model's performance. The validation dataset enables the selection of the final model and provides an assessment of when to stop early if the model is overfitting. The test dataset is used to evaluate the trained model's performance. The 60:20:20 split is a commonly used ratio because it strikes a balance between having enough data for training (which is essential for building a robust model) and having sufficient data for validation and testing (which is necessary for reliable model evaluation). This ratio is especially suitable for datasets of moderate size [18].

Python along with the TensorFlow and Keras libraries are used to construct the CNN. TensorFlow, as an open-source machine learning framework, provides a robust backend for efficient computation and network optimization [19]. Keras, a high-level neural network API, offers a user-friendly interface to define and train the CNN model. Leveraging TensorFlow's extensive operations and Keras' abstraction layers, the CNN architecture can be customised to include convolutional layers, pooling layers, and fully connected layers for feature extraction [20]. The libraries also provide diverse pre-processing utilities, such as image augmentation and normalisation, to enhance the network's robustness and generalisation.

The CNN model was comprised of a chosen amount of layers, as depicted in Fig. 3. The layers in a CNN are responsible for feature extraction through a process called convolution and pooling. The convolutional layers apply a set of learnable filters (also called kernels) to the input data. These filters are small, spatially localised grids. As the filters slide (convolve) across the input data, they perform element-wise multiplications and product-sum the results. This operation captures local patterns or features within the data. Each filter produces a feature map that highlights a specific pattern detected in the input. Multiple filters are applied simultaneously, producing multiple feature maps. These feature maps capture different patterns and gradually become more complex. After the convolution operation, the activation function ReLU is applied to the feature maps. This introduces non-linearity into the network, enabling it to learn complex relationships between features by replacing negative values with zero while leaving positive values unchanged. Next, a max-pooling layer is applied. Pooling reduces the spatial dimensions of the feature maps while retaining essential information. Max-pooling involves selecting the maximum value within a small window (e.g. 2×2 or 3×3) and discarding the rest. This down-sampling helps make the network more robust to variations in the input. After three convolutional and pooling layers, the feature maps are flattened into a 1D vector. This prepares the data for input into the fully connected layers. The choice of three layers here stems from the observation made in related work of keeping the model as simple as possible to explore efficient training times [5]. In the fully connected "Dense" layers, the network learns global patterns and relationships from the features extracted in the earlier layers. Each neuron in these layers is connected to every neuron in the previous layer. These layers perform complex transformations to map the representation between the input and output (classification). The CNN model constructed in this project has a "Dropout" layer between the final two "Dense" layers. The Dropout layer helps prevent overfitting by randomly deactivating units during training, making the network more robust [21]. The final Dense layer employs the Sigmoid activation is used to squash the output value between 0 and 1, representing the probability of the input being in the positive "ransomware" class for binary classification [22].

Increasing the number of convolutional and pooling layers

TABLE I
HYPERPARAMETERS [5]

Parameter	Value
Epoch	50
Batch Size	24
Learning Rate	1e-4
Rescaling Factor	1/255

allows the CNN to develop a deeper hierarchy of features. Lower layers tend to capture low-level image details such as edges or textures, while higher layers learn more abstract and semantic representations. As the images are complex and indistinguishable to the human eye, more than two layers was preferable. Three layers is a shallow network, but suitable to the amount of training samples we have. A larger network would require more training samples. This feature extraction process enables the CNN network to progressively learn and comprehend complex concepts. This contributes to its ability to generalise and make accurate predictions. The hyperparameters used can be found in Table I.

These hyperparameters are taken from a state-of-the-art research paper [5], which applied machine learning to entropy features of ransomware and goodware. Due to their success, and the fact that the datasets used are the same, we decided to incorporate their hyperparameters into this project to keep them consistent.

IV. EXPERIMENTAL SETTINGS

As mentioned in Section I-B, we used the same dataset as the authors used in [5]. This dataset was comprised of 785 ransomware samples and 328 goodware samples converted into images. The images for the experiment were constructed from the average of every 3 bytes, as mentioned in Section III-A. In [5], they performed multi-class classification on entropy features taken from the file samples. They used a siamese neural network, two CNNs that had shared weights.

We evaluate our method's performance based on classification accuracy, balanced accuracy and training times. The experiment consisted of 30 runs of our CNN where it was trained, validated and tested on the same datasets. The machine used for this experiment was the same each time, model Dell Optiplex 7060 PC.

V. RESULTS AND PERFORMANCE METRICS

The proposed model is evaluated using the balanced accuracy and training time analysis.

Balanced accuracy provides a balanced assessment of the model's performance because it considers the ability of the model to correctly classify instances of both the positive and negative classes. It is calculated as below:

$$Balanced\ \ Accuracy = \frac{1}{N}\sum_{i=0}^{N-1}\frac{correct_i}{total_i}, \tag{1}$$

where N is the number of classes, $correct_i$ is the correctly classified instances and $total_i$ is the total number of instances, for the ith class, respectively.

TABLE II
COMPARISON OF OUR MODEL AND STATE-OF-THE-ART SOLUTION.

Metric	Solution [5]	Our Model
Average Balanced Accuracy	95.92	84.70
Highest Balanced Accuracy	100.0	92.10

When the class distribution is imbalanced, a high overall accuracy can be misleading, as the model may be primarily predicting the majority class. Balanced accuracy helps account for this issue, providing a more equitable evaluation of the model's effectiveness. As illustrated in Table II, the CNN model achieved an average balanced accuracy of 84.7%. The highest balanced accuracy reached was a very promising 92.1%. These results underscore the model's ability to differentiate effectively between ransomware and goodware, even in scenarios where the class distribution is imbalanced thereby making reliable predictions for both classes. The benchmark solution method [5] achieved a higher average and best balanced accuracy, as seen in Table II, but out model remains competitive.

One notable advantage of the proposed approach is the significantly reduced training time in comparison to a state-of-the-art solution [5]. The state-of-the-art proposed solution is significantly more complex and therefore has a much higher training time compared to that of the proposed method in this project, as depicted in Fig. 9. Training time is a critical factor in the practical implementation of machine learning models, especially in scenarios where time-sensitive decision-making or real-time analysis is required. This scenario is especially applicable to ransomware detection due to the high stakes. As ransomware attacks continue to evolve, the ability to quickly adapt and retrain models with updated threat data is crucial. The efficient training process facilitates this adaptability, allowing for the incorporation of the latest threat intelligence and ability to stay ahead of the threat landscape.

The efficiency of our training process stems from a combination of factors, including the simple model architecture and data preprocessing. By streamlining these elements, the model has achieved a huge reduction in training time while maintaining competitive performance.

VI. FURTHER ANALYSIS

TensorFlow's CNN capability outputs the classification accuracy which can be calculated as:

$$\text{Accuracy} = \frac{\text{Number of Correct Classifications}}{\text{Total Number of Classifications}}. \tag{2}$$

The classification accuracy was used during training to monitor the training accuracy over the epochs, forming a convergence graph. Fig. 10 shows the convergence graph of the average classification accuracies over 30 different experiments at each epoch. The whiskers show the range of accuracies at each epoch. Smaller whiskers imply a more consistent model, one that tends to converge to the same local minimum of the search space across different runs or experiments. This consistency suggests that the model's training is stable, and it reliably finds the same or very similar solutions when

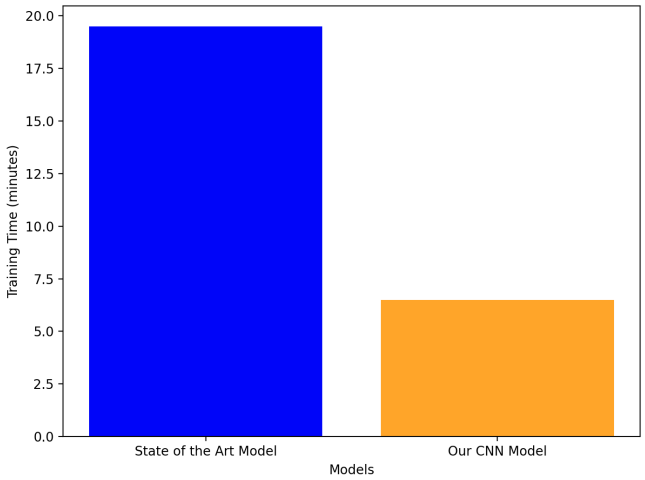

Fig. 9. Training time comparison of the proposed method and the benchmark method.

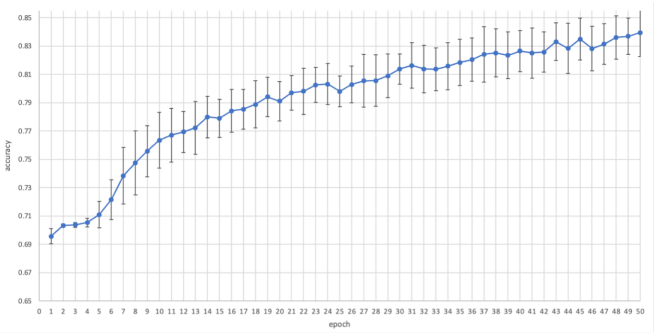

Fig. 10. The convergence graph for the proposed method.

trained multiple times. Conversely, larger whiskers indicate that the model is exploring different local minima within the search space during different runs or experiments. Fig. 9 demonstrates that the training accuracy reaches approximately 84% at epoch 50. The whiskers do not decrease in size as training progresses. This observation suggests that the model may be less consistent. However, as the whiskers are not changing in size, this reflects the model's consistency across all runs. While larger whiskers may indicate some variability, it also shows that the model is exploring various areas of the problem space in each experiment. Also in Fig. 9, it is apparent where the model is in the exploration phase and exploitation phase. In the context of reinforcement learning, a model must first explore new actions to learn about its environment (exploration) and then exploit known actions to maximise its rewards (exploitation). Exploration is shown by the model making large jumps in terms of the steeper increase in accuracy as shown in Fig. 9. Then the model makes smaller movements to exploit the information it has gained. This is picture in the smaller increases in accuracy towards the later epochs on the graph, ultimately leading to convergence.

VII. Sustainability Considerations

This project, by focusing on early detection of ransomware through a machine learning model, presents notable implications for sustainability considerations, namely encompassing environmental, social, economic and technical aspects.

A. Environmental Sustainability

Despite this project not impacting the environment directly, the indirect contributions due to the nature of cybersecurity are considerable. Swiftly detecting ransomware with CNN-based analysis can potentially reduce the environmental footprint associated with recovery efforts of ransomware attacks. With diligent cybersecurity, individuals and organisations alike can significantly minimize resource wastage.

B. Social Sustainability

Privacy is one of the potentially most pressing social sustainability issues, valued unequivocally by both individuals and organisations. In terms of privacy, this project directly addresses ransomware attacks that compromise sensitive information. By developing a proactive defense mechanism, the project absolutely aids in protecting the privacy, and thus security of not just individuals but potentially communities. This ultimately enhances social well-being and fosters a sense of much needed security in the digital realm, ultimately contributing to social sustainability.

C. Economic Sustainability

Economic implications of ransomware are obvious and substantial, and often why solutions are first considered. Ranging from directly paying ransoms to other economic effects such as recovery and reputation damage. A focus on cybersecurity can secure digital assets for individuals and organisations, as well as increase financial trust in existing systems and services alike. Therefore, the early detection system that my project delivers offers potential direct cost savings, as swift identification can reduce economic burden of recovering from a ransomware attack as well as direct ransom attempts, thereby supporting economic sustainability in both short and long-term solutions.

D. Technical Sustainability

The project aligns directly with technical sustainability by utilising appropriate technologies, specifically using a simple CNN for efficient and effective detection of ransomware. The choice of my project to use a CNN demonstrates clear application of cutting-edge technology that supports functional and maintenance requirements of technology over a sustained time period. The use of this specific technology ensures that the project stays relevant and effective, therefore technically sustainable, in the ever-evolving landscape of technology, especially cybersecurity.

VIII. Limitations and Future Work

A constraint identified during the development of the project's system is the absence of a preemptive check for packed ransomware files before the transformation into images. In future applications, a careful consideration should

be made to ensure the dataset's validity and integrity by unpacking the packed ransomware samples before they are converted into images. The purpose of packing is to obfuscate the code and make it more challenging to analyse or detect by security analysts and antivirus software. By employing packing techniques, ransomware authors aim to evade detection, prolong their malware's lifespan, and increase its chances of successfully infecting systems. Unpacking the ransomware before the image creation would show the true instructions or opcodes it gives the system, adding to the data's validity and integrity.

Another constraint identified was the loss of information during converting the binaries to images. Representing files as images inherently leads to the loss of data because this process took the first 70KB of all files, missing the rest. This data could contain crucial characteristics of the files, potentially impacting the model's ability to discern certain ransomware patterns. In future, the header of the files could be skipped as that data is not as highly correlated to ransomware classification.

Another limitation identified was the dataset size. The dataset is relatively small and imbalanced compared to that of other studies. A larger dataset would provide the model with more training data so the model can learn to differentiate between subtle variations in ransomware and goodware behaviour. Expanding the dataset can help address the issue of class imbalance, creating a more equal representation of ransomware and goodware instances. This, in turn, could lead to a model that is less biased toward the majority class, ultimately reducing the likelihood of false negatives.

To further evaluate this model's performance, we aimed to perform a statistical significance "T" test against the compared solution, [5]. However, this required performance metrics that were not available. Future work could focus on determining whether any observed differences between the model's performance and the solution presented in [5] are statistically significant. This would offer valuable insights for researchers and practitioners looking to build upon or utilise the model in real-world applications, as well as contribute to the ongoing development of ransomware detection.

Another direction this research could take, is to combine this static analysis technique with behaviour-based analysis. This aims to compete with real-world state-of-the-art existing defenses against ransomware attacks, as mentioned in [6], where Crowdstrike uses static and dynamic analysis techniques to detect ransomware.

IX. Conclusion

In conclusion, this project presents a novel approach to address the challenging task of early ransomware detection. By creating a comprehensive dataset of images generated from the operational codes (opcodes) of both ransomware and benignware, this project aimed to provide an early detection mechanism for ransomware attacks. The CNN model demonstrated its ability to differentiate between these types of files, achieving an average balanced accuracy of 84.17%, competing with existing state-of-the-art solutions. This promising result suggests that the CNN model can be an effective tool for detecting ransomware, potentially minimising the devastating consequences of such attacks. While various approaches have been proposed for this problem, the image analysis technique offers a unique perspective that holds promising potential.

Acknowledgments

I would like to thank Harith Al-Sahaf and Shabbir Abbasi for their continuous help and support on this project.

References

[1] J. Fruhlinger, "Ransomware explained: How it works and how to remove it." https://www.csoonline.com/article/3236183/what-is-ransomware-how-it-works-and-how-to-remove-it.html, 2023.

[2] CertNZ, "Quarter four cyber security insights 2022." https://www.cert.govt.nz/about/quarterly-report/quarter-four-cyber-security-insights-2022/, 2023. Accessed: 2023-05.

[3] G. Olano, "Loot from nz ransomware attack being sold on dark web." https://www.insurancebusinessmag.com/nz/news/cyber/loot-from-nz-ransomware-attack-being-sold-on-dark-web-431229.aspx, 2022. Accessed: 2023-05.

[4] R. Stock, "Loot from nz ransomware attack being sold on dark web." https://www.stuff.co.nz/business/money/300840473/kiwibank-counting-customers-whose-id-data-was-stolen-in-massive-latitude-privacy-breach, 2023. Accessed: 2023-05.

[5] J. Zhu, J. Jang-Jaccard, A. Singh, I. Welch, H. AL-Sahaf, and S. Camtepe, "A few-shot meta-learning based siamese neural network using entropy features for ransomware classification," *Computers & Security*, vol. 117, p. 102691, 2022.

[6] CrowdStrike, "Crowdstrike." https://www.crowdstrike.com/, 2023. Accessed: 2023-05.

[7] D. Rozimovschii, "Leveraging the dark side: How crowdstrike boosts machine learning efficacy against adversaries." https://www.crowdstrike.com/blog/how-crowdstrike-boosts-machine-learning-efficacy-against-adversarial-samples/, 2023. Accessed: 2023-05.

[8] CrowdStrike, "Crowdstrike wins tenth consecutive av-comparatives award, highlighting falcon's proven efficacy and market-leading technology." https://www.crowdstrike.com/press-releases/crowdstrike-wins-tenth-consecutive-av-comparatives-award/, 2023. Accessed: 2023-05.

[9] F. Khan, C. Ncube, L. K. Ramasamy, S. Kadry, and Y. Nam, "A digital dna sequencing engine for ransomware detection using machine learning," *IEEE Access*, pp. 1–1., 2020. doi:10.1109/ACCESS.2020.3003785.

[10] J. Baldwin and A. Dehghantanha, "Leveraging support vector machine for opcode density based detection of crypto-ransomware," *Cyber Threat Intelligence*, pp. 107–136, 2018.

[11] B. M. Khammas, "Ransomware detection using random forest technique," *ICT Express*, vol. 6, no. 4, pp. 325–331, 2020.

[12] H. Zhang, X. Xiao, F. Mercaldo, S. Ni, F. Martinelli, and A. K. Sangaiah, "Classification of ransomware families with machine learning based onn-gram of opcodes," *Future Generation Computer Systems*, vol. 90, pp. 211–221, 2019.

[13] B. Zhang, W. Xiao, X. Xiao, A. K. Sangaiah, W. Zhang, and J. Zhang, "Ransomware classification using patch-based cnn and self-attention network on embedded n-grams of opcodes," *Future Generation Computer Systems*, vol. 110, pp. 708–720, 2020.

[14] R. Gate, "Illustration of max pooling and average pooling." https://www.researchgate.net/figure/Illustration-of-Max-Pooling-and-Average-Pooling-Figure-2-above-sho/ws-an-example-of-max_fig2_333593451, 2023. Accessed: 2023-05.

[15] R. Gate, "Schematic diagram of a basic convolutional neural network (cnn) architecture." https://www.researchgate.net/figure/Schematic-diagram-of-a-basic-convolutional-neural-network-CNN-architecture-26_fig1_336805909, 2023. Accessed: 2023-05.

[16] P. J. Delorme, "Image preprocessing." https://medium.com/unpackai/image-preprocessing-6654d1bb4daa#:~:text=There%20can%20be%20great%20benefit,and%20even%20standardising%20the%20values., 2021. Accessed: 2023-10.

[17] Tanya, "Data preprocessing and network building in cnn." https://towardsdatascience.com/data-preprocessing-and-network-building-in-cnn-15624ef3a28b, 2020. Accessed: 2023-10.

[18] P. Baheti, "Train test validation split: How to & best practices." https://www.v7labs.com/blog/train-validation-test-set, 2021. Accessed: 2023-08.

[19] TensorFlow, “Tensorflow.” https://www.tensorflow.org/, 2023. Accessed: 2023-05.
[20] Keras, “Keras.” https://keras.io/, 2023. Accessed: 2023-05.
[21] baeldung, “How relu and dropout layers work in cnns.” https://www.baeldung.com/cs/ml-relu-dropout-layers#:~:text=4.-,The%20Dropout%20Layer,and%20leaves%20unmodified%20all%20others., 2023. Accessed: 2023-10.
[22] M. Mishra, “Convolutional neural networks, explained.” https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939, 2020. Accessed: 2023-10.