Signal-Based Malware Classification Using 1D CNNs

Jack Wilkie[1*], Hanan Hindy[2], Ivan Andonovic[1], Christos Tachtatzis[1], Robert Atkinson[1]

[*]Jack Wilkie, jack.wilkie.2017@uni.strath.ac.uk.
[1]University of Strathclyde.
[2]Ain Shams University.

Abstract

Malware classification is a contemporary and ongoing challenge in cyber-security: modern obfuscation techniques are able to evade traditional static analysis, while dynamic analysis is too resource intensive to be deployed at a large-scale. One prominent line of research addresses these limitations by converting malware binaries into 2D images by heuristically reshaping them into a 2D grid before resizing using Lanczos resampling. These images can then be classified based on their textural information using computer vision approaches. While this approach can detect obfuscated malware more effectively than static analysis; the process of converting files into 2D images results in significant information loss due to both quantisation noise, caused by rounding to integer pixel values, and the introduction of 2D dependencies which do not exist in the original data. This loss of signal limits the classification performance of the downstream model. This work addresses these weaknesses by instead resizing the files into 1D signals which avoids the need for heuristic reshaping, additionally these signals do not suffer from quantisation noise due to being stored in a floating-point format. It is shown that existing 2D CNN architectures can be readily adapted to classify these 1D signals for improved performance. Furthermore, a bespoke 1D convolutional neural network, based on the ResNet architecture and squeeze-and-excitation layers, was developed to classify these signals and evaluated on the MalNet dataset. It was found to achieve state-of-the-art performance on binary, type, and family level classification with F1 scores of *0.874*, *0.503*, and *0.507*, respectively, paving the way for future models to operate on the proposed signal modality.

Keywords: Malware Classification, Malware Detection, Machine Learning, Convolutional Neural Networks, Neural Networks, Computer Vision, MalNet

1 Introduction

Malware detection remains a critical challenge in cyber-security. While signature-based classification is effective for known malware, it relies on handcrafted features which are time-consuming and require expert knowledge to develop (Votipka et al. 2020; Mohaisen et al. 2014). Furthermore, small changes to a malware's source code can greatly alter the compiled binary, allowing signature-based detection to be readily evaded (Sathyanarayan et al. 2008). While dynamic analysis provides a more in-depth inspection, its resource intensive nature limits its applicability at scale, posing significant challenges for cybersecurity defences that require rapid and accurate malware assessments.

Recent advances in machine learning have offered promising avenues for enhancing malware classification. Specifically, the transformation of malware binaries into visual image representations, known as byteplots, has emerged as a novel method that leverages computer vision techniques to classify malware samples based on the byteplot's textural information (Conti et al. 2008; Nataraj et al. 2011a). This approach boasts numerous advantages, removing the need to develop hand-crafted features and malware signatures, moreover the malware images are quick to generate allowing these methods to be deployed at scale. Significantly, this approach has enabled deep learning classification models which have been shown to be robust to both polymorphic malware variants and obfuscation techniques such as section reordering and file packing (Nataraj et al. 2011a,b; A. et al. 2023).

However, this approach is not without its limitations. The process of converting malware files into 2D images introduces significant information loss due to quantisation noise when rounding to integer pixel values and, more significantly, the artificial introduction of 2D spatial dependencies, which can distort the original binary structure and degrade the performance of downstream classification models.

Addressing these limitations, this work proposes an innovative approach that resizes the malware binaries into 1D signals instead of 2D images. This method not only maintains the integrity of the original data structure, but also retains more of the binary's information, improving the signal-to-noise ratio of the resized signal. It is shown that existing 2D models can be adapted to operate on 1D data for improved performance with no additional parameters or computation. Furthermore, a bespoke Convolutional Neural Network (CNN) architecture is proposed to classify the 1D signal data. This architecture is adapted from the pre-activation ResNet architecture (He et al. 2016), however, also incorporates squeeze-and-excitation (SE) layers (Hu et al. 2018) and the GELU activation function (Hendrycks and Gimpel 2023).

While the proposed pipeline can be applied to arbitrary file binaries, this work focuses on the classification of Android DEX files due to the availability of a large-scale dataset for training and evaluation, and the substantial adoption of third-party applications in the Android ecosystem. Despite this, the proposed signal-based classification method is experimentally shown to be more effective than the equivalent image-based approaches on Windows' EXE binaries, making it an attractive replacement to image-based models.

Comprehensive evaluation on the MalNet dataset (Nataraj et al. 2011a) demonstrates that the proposed approach not only surpasses 2D image-based models but also achieves state-of-the-art performance on binary, type, and family level malware classification. This work paves the way for new malware classification systems which rely on 1D signal-based representations.

The core contributions of this work can be summarised as follows:

1. A resizing approach is proposed which keeps the file binaries in a 1D format, improving the signal-to-noise ratio when compared to resizing as a 2D image. Additionally, by storing the signals in a floating-point format quantisation error is avoided.
2. A procedure to adapt arbitrary 2D CNNs to operate on the 1D signal representations is proposed. The 1D models were compared to their equivalent 2D models in a malware classification task on the MalNet and the Microsoft malware classification datasets. It was found that for an equal number of parameters and compute the 1D models outperform their 2D counterparts.
3. A bespoke 1D CNN model is proposed to classify the malware signals. The model is based on the pre-activation ResNet architecture with GELU activation functions and SE layers. The proposed model outperforms state-of-the-art approaches in binary, type, and family classification on the MalNet dataset.

This work is organised as follows: related works are outlined in Section 2, including previous approaches to malware classification, byte plot images, and previous approaches utilising 1D convolutions for malware classification. Section 3 provides overview and explanation of the proposed methods including generating the signal files from binaries, and the novel CNN architecture used. The proposed approach is experimentally evaluated in Section 4, with additional results and ablations being provided in Section 5. Finally, relevant discussion and conclusions are provided in Section 6.

2 Related Works

In this section, a summary of previous works is given. Existing malware classification approaches are described in Section 2.1, where methods such as static analysis, dynamic analysis, and machine learning approaches are detailed. Section 2.2 covers the history and application of byteplot representations, including their uses in manual binary inspection and as an input to machine learning models. Finally, Section 2.3 describes other works which have used 1D convolutions for malware classification.

2.1 Malware Classification

The classification of malware involves the extraction of distinctive features that characterise the contents and functionalities of malicious software. These features are typically gathered through static or dynamic analysis methods. Static analysis extracts features directly from the malware files, such as strings, n-grams, and byte-entropies (Abusitta et al. 2021). Conversely, dynamic analysis, while more time-intensive, provides deeper insights into the malware's functionalities. This method entails running the malware within a controlled virtual environment to observe behaviours such as network activity, permissions, and API calls. This approach is particularly effective against malware that employs obfuscation techniques to evade detection (Abusitta et al. 2021).

Signature-based detection remains a prevalent method in antivirus engines, where human analysts identify common properties among malware samples to define a malware signature. Such signatures may include filenames, byte sequence regular expressions, or printable strings (Schultz et al. 2001). Despite the high precision of signature-based systems, they are susceptible to evasion through anti-analysis techniques including obfuscation, polymorphism, packing, and code reordering. These techniques alter the malware binaries substantially with minimal changes to their functionalities, thereby evading traditional detection methods (Deng and Mirkovic 2022).

In response to the limitations of signature-based detection, researchers have increasingly turned to machine learning classifiers capable of generalisation. Various gradient free methods have been applied such as Support Vector Machines (SVMs) (Rezende et al. 2018), Markov Chains (Anderson et al. 2012), and K-Nearest Neighbours Classifiers (KNNs) (Nataraj et al. 2011a; Rezende et al. 2018), as well as deep models such as CNNs (Freitas et al. 2021) and Vision Transformers (ViTs) (Seneviratne et al. 2022a). One prevalent method involves restructuring and resizing the binary data into a 2D image which can then be classified using computer vision approaches, this has numerous advantages such as being quick to generate and being resistant to obfuscation (Nataraj et al. 2011a).

2.2 Byteplot Visualisation

The visualisation of binaries as byteplots was initially proposed as a tool for security analysts, wherein each byte was represented as a pixel with an intensity corresponding to the integer value of the byte (Conti et al. 2008). These byteplots were displayed in a window of fixed width, allowing users to scroll through the entire binary file. It was qualitatively demonstrated that properties of the binary could be inferred from the textural information of the plots. For instance, network traffic of varying length packets manifested as an irregular texture, and hidden messages could be discerned in MP3 files. This enabled analysts to identify regions of interest without the need for manual examination of the binaries. Subsequent research extracted statistical information from byteplots and utilised it to automatically label primitive data types and sections within a binary file (Conti et al. 2010).

Byteplots have since found utility in computer vision-based classification approaches, where binary files are reshaped into a 2D grid. The width of this grid varies depending on the size of the binary, according to a heuristic resizing rule. The resulting 2D grid is then resized into a 2D image suitable for classification (Nataraj et al. 2011a). Initially, filters were manually designed to extract feature vectors from these images, which could then be classified using SVMs (Rezende et al. 2018) or KNNs (Nataraj et al. 2011a). However, the success of CNNs on byteplot images has obviated the need for manual feature extraction (Rezende et al. 2018; Kalash et al. 2018; Llauradó 2016).

While it is common to generate greyscale byteplot images using standard procedures, several variations have been proposed. One approach seeks to enhance the images by incorporating colour channels to encode semantic information (Gennissen and Blasco 2017), or by encoding multiple bytes into a single pixel (Huang and Kao 2018). Another variation suggests replacing the commonly used heuristic reshaping rule with a method that shapes the data into a 2D grid with a width determined by the length of the longest section in the binary. However, this results in significant padding in rows corresponding to smaller sections (Chong et al. 2022).

Due to the success of computer vision-based classification, byteplot representations have been widely adopted in research (Priya and Sathya Sofia 2023). Notably, the MalImg dataset contains 9,339 byteplot images from 25 different families (Nataraj et al. 2011a). Another dataset, MalNet, comprises over 1 million byteplot images extracted from the DEX files of Android APKs (Freitas et al. 2021). Additionally, converting binaries into byteplot images is a commonly used preprocessing step (Kalash et al. 2018; Llauradó 2016; Lu et al. 2022) in works utilising the Microsoft Malware Classification dataset (Ronen et al. 2018).

2.3 1D CNNs for Malware Classification

1D CNNs have been utilised for malware classification, though not by operating on a 1D signal representations. A typical approach involves extracting tabular features through static or dynamic analysis. Common features can include n-grams (Liu et al. 2017; Yeboah and Musah 2022), header-based features (Damasevicius et al. 2021), structural entropy (Paik et al. 2022; Kim et al. 2023), and byte/opcode frequencies (Liu et al. 2017; Paik et al. 2022). These features are then used as input for 1D CNNs, either as standalone classifiers (Bovenzi et al. 2022) or within an ensemble of models (Damasevicius et al. 2021).

1D CNNs have shown prominence in processing sequences derived from API calls or opcode features. Initially, data is collected as strings via disassembly (Llauradó 2016; McLaughlin et al. 2017; Yeboah and Musah 2022; Wang et al. 2021; Safa et al. 2019; Dib et al. 2021) or dynamic analysis (Schofield et al. 2021). These strings are then parsed to identify a discriminative subset of opcodes or API calls, creating a short sequence for each sample. The sequences are then encoded using methods such as one-hot encoding (Schofield et al. 2021; Schofield 2021; McLaughlin et al. 2017), look up tables (McLaughlin et al. 2017), GloVe (Wang et al. 2021), word-2-vec (Yeboah and Musah 2022; Llauradó 2016) or frequency based features (Schofield 2021) to obtain a sequence of embeddings which can then be classified using a shallow CNN (Schofield 2021; McLaughlin et al. 2017; Wang et al. 2021; Schofield et al. 2021), or convolutional layers followed by a recurrent model (Alsulami and Mancoridis 2018; Safa et al. 2019; Zyout et al. 2023; Dib et al. 2021). However, these data processing methods are time-consuming, and often these models underperform compared to byteplot or handcrafted feature-based approaches (McLaughlin et al. 2017).

Despite the prevailing preference of 2D byteplot images in machine learning based approaches on raw binaries, some works have shown evidence of 1D convolutions providing performance benefits. One work has shown that the convolutional filters in 2D CNNs trained on byteplot images show a preference towards row-based features with column based features being less impactful on the binaries predicted label (Chong et al. 2022). Additionally, it was found that by ensembling a 2D CNN with features extracted by a single 1D convolutional layer operating on flattened byteplots provided performance benefits over using a 2D model alone (Chong et al. 2022). Other works have also performed 1D convolutions on flattened byteplot images, finding performance

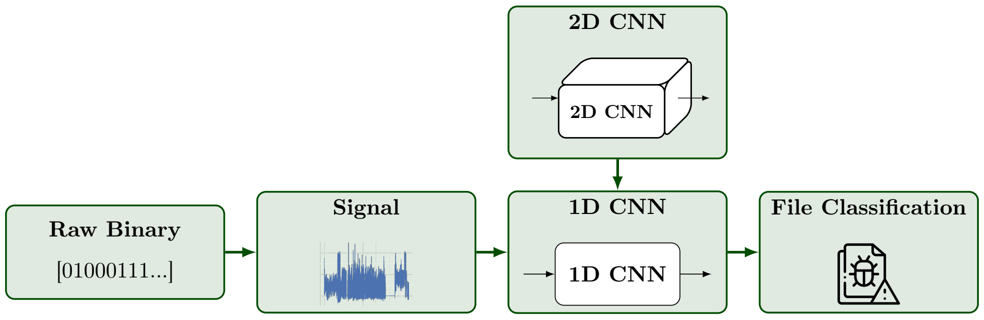


Fig. 1 High-level overview of the proposed system. File binaries are initially converted into 1D signals, which preserve more of the file information, before being classified using a 1D CNN. Additionally, existing 2D CNNs can be adapted to operate on the 1D signals by modifying the convolution kernels.

benefits in eliminating the vertical dimension in the convolutions (Kim et al. 2023). Although these works have used relatively shallow networks, it has been attempted to train a Vision Transformer on entire 1D binaries, however, the authors found this impractical due to the file sizes involved and instead switch to a 2D byteplot based approach (Lu et al. 2022). Other attempts have dealt with this file size constraint by operating only on the first or last N bytes in a file (Hasegawa and Iyatomi 2018) or by identifying discriminate regions within a binary and extracting sub-sequences for classification (Kim et al. 2022). This paper presents the first work to train deep 1D CNNs on entire binaries resized into 1D signals, marking a novel approach in machine learning based malware classification.

3 Proposed Approach

A high-level schematic of the proposed approach is provided in Figure 1. This represents an approach to perform malware classification based on 1D signal representations of binary files. These representations are produced by resizing binaries into 1D signals using Lanczos resampling as described in Section 3.1. To classify the signal representations a methodology is presented in Section 3.2 to convert existing 2D CNNs to operate on 1D data by flattening the convolution kernels and squaring the stride values. This technique is applied to the ResNet model along with several modifications to develop a bespoke architecture adept at classifying the generated signals in Section 3.3.

3.1 Data Processing and Resizing

Instead of processing malware binaries into byteplot visualisations, this work converts them into signal representations. For the purposes of this work the term “signal” is used to refer to 1D inputs to machine learning models, in order to differentiate from 2D images. Initially, the binary file is transformed into an integer representation where each byte is represented by an integer in the range $[0, 255]$, corresponding to the byte’s numerical value when interpreted as an integer.

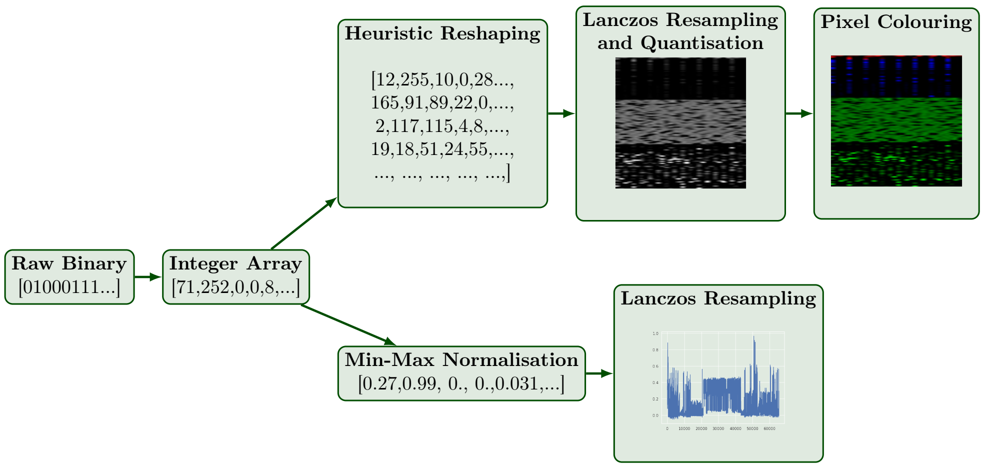


Fig. 2 The signal processing procedure use to generate both byteplot images (top path) and signal representations (bottom path). The signals are generated in a similar manner to bytplot images but do not require the reshaping or quantisation steps which introduce unnecessary noise when generating byteplot images.

As depicted in Figure 2, generating a byteplot image involves arranging integer values into a 2D grid using a heuristic resizing rule. However, the proposed data processing method diverges at this point by employing min-max normalisation. Here, each integer value is divided by 255, resulting in a floating-point representation in the range $[0, 1]$. By maintaining the data's original 1D structure the signal representations avoid the introduction of 2D spatial dependencies which result in lower quality resizing and diminished downstream classification performance.

Similar to byteplot visualisations, the Lanczos filter is used to resample the data. This is due to its ability to preserve textural information in malware binaries useful for classification. Although signals can be resampled to arbitrary size, this work resamples to a length of 65536, equal to the number of pixels in a 256×256 image. This allows for fair comparison between the proposed approach and existing image-based classifiers. Post-resizing, the signal remains in floating-point format, which avoids the quantisation loss suffered by byteplot representations.

In contrast to byteplot visualisations, this work does not encode binary section information as multiple channels; instead, a single channel is used. While encoding information as colour channels is beneficial for visualisation purposes, it does not enhance the performance of the downstream classifier despite resulting in additional model complexity and data storage requirement (Freitas et al. 2021). The byteplot visualisation of samples from the MalNet dataset and their corresponding signal representations are shown in Figure 3. Correlation can be seen between the byteplot image's pixel intensity and the corresponding signal values. Notably regions of padding appear as black regions in byteplot images and long 0 regions in the signal representation. Before model training, the mean and standard deviation of the signal are calculated on the training set, enabling z-normalisation of the signal.

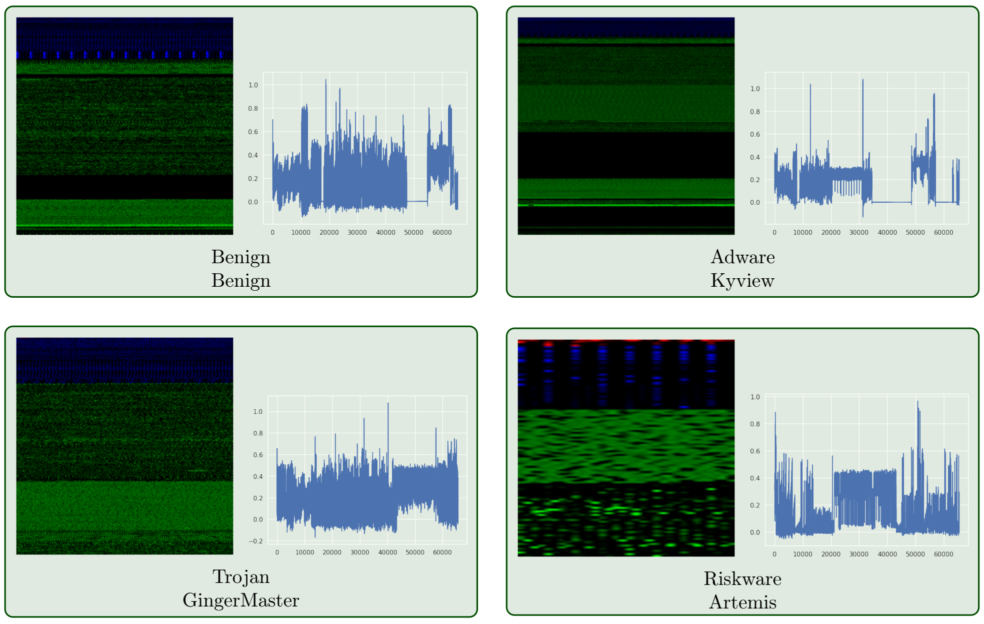
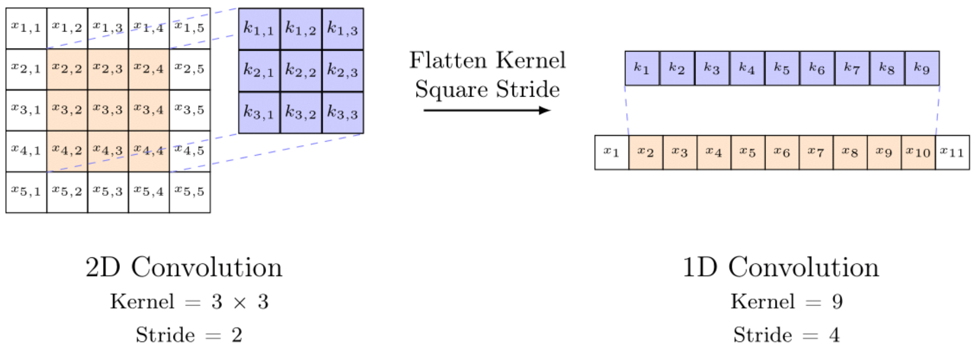


Fig. 3 Comparison of byteplot images with the corresponding signal representation. Visual similarity can be seen between the representations, most notably on areas of padding which appears as black regions in the image and 0 regions in the signals. The captions show the type of malware (top) and its family (bottom).

3.2 Converting Convolutions to One Dimension

Typical 2D CNN classifiers are unable to process the proposed 1D signal representations of file binaries, necessitating an architecture able to operate on these signals. Existing approaches in other domains have attempted to adapt CNNs to 1D data by either preserving the convolution size along a single dimension—for instance, converting a 2D convolutional layer with a 3×3 kernel into a 1D convolutional layer with a kernel size of 3 (Daly et al. 2023)—or by employing custom architectures with kernel sizes tailored to the specific characteristics of the input data (Mekruksavanich and Jitpattanakul 2022). However, these adaptations do not result in 1D architectures equivalent to their 2D counterparts, with discrepancies in model structure, parameter count, and computational requirements when compared to their 2D equivalents.

This work instead proposes a novel approach to convert 2D CNNs into 1D CNNs that preserves the network architecture and key properties such as parameter count and computational cost. As illustrated in Figure 4, instead of modifying kernel sizes arbitrarily, the kernels are flattened, maintaining the same number of parameters, except across a single dimension. Additionally, the stride values are squared preserving the downsampling ratio in convolutional layers (a stride of 2 across 2 dimensions results in a total downsampling ratio of 4). By performing this transformation on all 2D convolutions in a model, existing CNN architectures can readily be adapted to the



$x_{1,1}$ $x_{1,2}$ $x_{1,3}$ $x_{1,4}$ $x_{1,5}$
$x_{2,1}$ $x_{2,2}$ $x_{2,3}$ $x_{2,4}$ $x_{2,5}$
$x_{3,1}$ $x_{3,2}$ $x_{3,3}$ $x_{3,4}$ $x_{3,5}$
$x_{4,1}$ $x_{4,2}$ $x_{4,3}$ $x_{4,4}$ $x_{4,5}$
$x_{5,1}$ $x_{5,2}$ $x_{5,3}$ $x_{5,4}$ $x_{5,5}$
$k_{1,1}$ $k_{1,2}$ $k_{1,3}$
$k_{2,1}$ $k_{2,2}$ $k_{2,3}$
$k_{3,1}$ $k_{3,2}$ $k_{3,3}$
Flatten Kernel
Square Stride
k_1 k_2 k_3 k_4 k_5 k_6 k_7 k_8 k_9
x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 x_9 x_{10} x_{11}
2D Convolution
Kernel = 3 × 3
Stride = 2
1D Convolution
Kernel = 9
Stride = 4

Fig. 4 Conversion of a 2D CNN Kernel into a 1D CNN kernel. By flattening the kernels and squaring the stride values across a 2D CNN, an equivalent 1D CNN can be found with the same number of parameters and compute requirement. Here the kernel is represented by k, and the input is given as x.

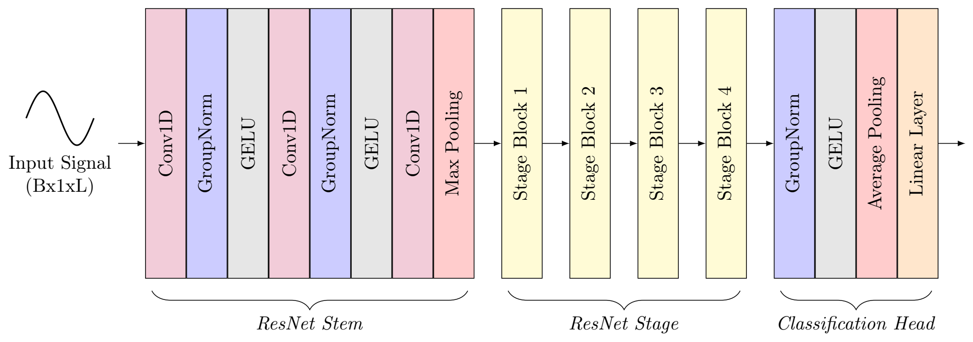


Fig. 5 Architecture of the proposed 1D ResNet. The model is based on the ResNet architecture with a deep stem, however, 2D convolutions are replaced with 1D Convolutions. Additionally, the kernel is flattened and the stride is squared for each convolution.

proposed signal representations. The 1D models show improved performance for no extra parameter or compute cost over their 2D counterparts.

3.3 Model Architecture

To demonstrate the efficacy of malware classification using signal representations, the bespoke 1D CNN architecture shown in Figure 5 was developed by converting the 2D ResNet architecture (He et al. 2015) to operate on 1D signals. To modernise the architecture, the ReLU activation functions were replaced with GELU and squeeze-and-excitation (SE) layers were introduced in the PRE-SE configuration (Hu et al. 2018). The network has three stages: the stem, which performs initial feature extraction whilst spatially downsampling the input signal; the stage, which applies residual learning to progressively learn a feature map; and the classification head, which predicts the binary's class label based on the extracted features.

The model's stem utilises the deep architecture introduced by the ResNet-D models (He et al. 2018), employing a hierarchical sequence of convolutional, normalisation, and downsampling operations to extract features from the input signal whilst spatially downsampling it to improve downstream efficiency. It consists of three convolutions: two initial convolutions, each followed by Group Normalisation (GroupNorm) and a GELU activation function; and a third convolution which is followed by max pooling. The first convolution employs a stride of 4, reducing the input signal's length by a factor of four whilst projecting it to 32 channels. The second convolution maintains the spatial dimensions and channel count, refining feature representations through additional non-linear transformations. A final convolutional layer then expands the channel depth to 64, followed by a max pooling operation with a stride of 4, further downsampling the signal length by a factor of four. A kernel size of nine is used across the max pooling and convolutions in the stem.

The ResNet stage consists of 4 stage blocks containing a series of the residual blocks shown in Figure 6 (left). These blocks are 1D adaptations of the pre-activation ResNetV2 architecture (He et al. 2016) with the aforementioned modernisations of the GELU activation and SE layers. Each residual block begins with GroupNorm, followed by a GELU activation and SE layer, after which the signal is processed through two parallel branches. The main branch employs separable convolutions, beginning with a pointwise convolution (followed by GroupNorm and GELU) which reduces the dimensionality of the feature map; before a depthwise convolution (also followed by GroupNorm and GELU) processes the signal in the lower dimensionality feature space. A second pointwise convolution then restores original number of channels. Meanwhile, the residual branch resamples the signal to match the shape of the main branch using average pooling (to resize the spatial dimension) and a pointwise convolution (to match the number of channels). The two branches are then reconciled using a pointwise addition. Importantly, the first residual block in each stage block downsamples the length of the signal by a factor of 4, using the stride parameter of the depthwise convolution, and doubles the number of channels using the pointwise convolutions. This progressive downsampling allows the model to learn semantic, hierarchical features as the signal propagates through the network.

The SE layers are used to provide channel-wise attention to the residual block. As illustrated in Figure 6 (right), global average pooling is used to get a compressed representation of the feature map in the current layer of the network. This pooled representation is then passed through an MLP, utilising the bottleneck architecture, to compute a scaling vector used gate the feature channels. By scaling the input to the convolutions, the model is able to exploit input dependant filters.

Finally, the classification head is used to predict the class of the input binary using the feature map learned by the ResNet stage. The extracted features are first normalised using GroupNorm. Following this, a GELU activation is used to allow the classification head to make predictions based on a non-linear combination of features. Finally, average pooling is used to eliminate the length dimension of the signal, with the remaining features being classified using a linear layer.

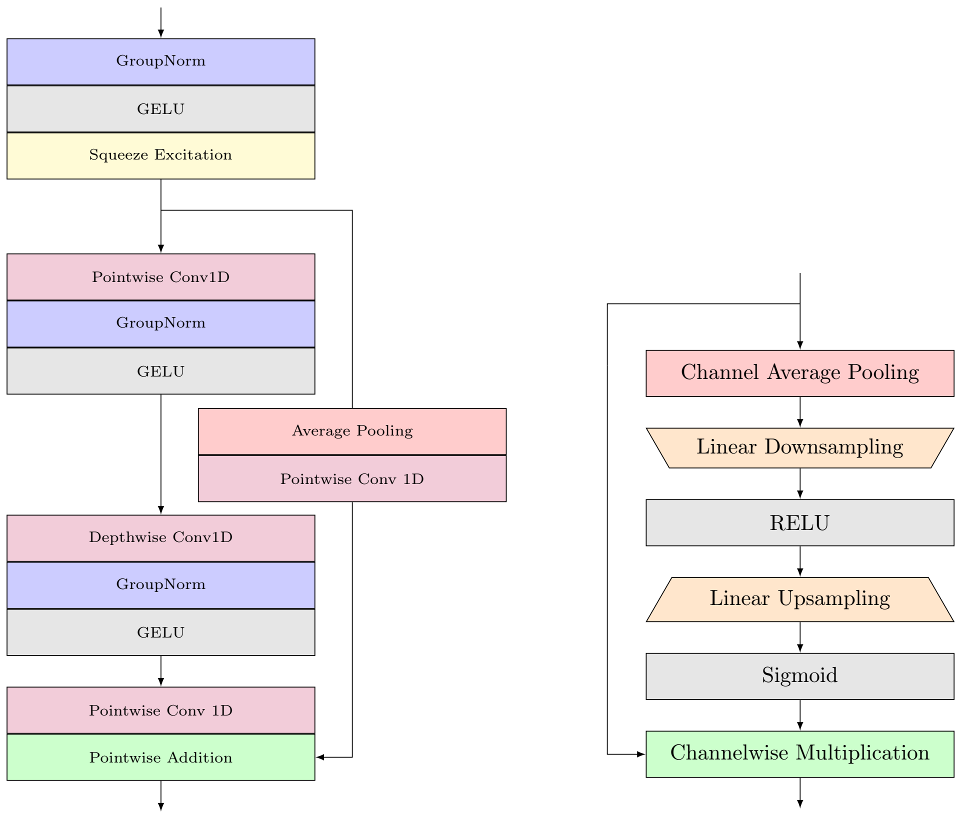


Fig. 6 Architecture of the residual block. **Left:** The blocks use the pre-activation ResNet architecture with the addition of the GELU activation function and a squeeze-and-excitation layer. **Right:** architecture of a squeeze-and-excitation block used to provide input dependant convolution filters.

4 Evaluation

This section experimentally evaluates this works primary contributions. Initially, the evaluation datasets and experimental procedures are detailed in Section 4.1 and Section 4.2, respectively. Next the performance impact of transitioning from a byteplot representation of a binary to a signal-based one is measured in terms of both signal loss and performance impact on downstream models in Section 4.3. Section 4.4 compares various approaches to convert 2D convolutional filters to operate on 1D data. Finally, Section 4.5 compares the proposed model to current SOTA approaches on the MalNet dataset in binary, type, and family level classification.

4.1 Datasets

This work primarily uses the MalNet Image dataset (Freitas et al. 2021). This is the largest open source malware datasets containing 1,262,024 Android malware samples across 696 families and 47 types of malware, including benign samples. While the

MalNet dataset is provided as byteplot images, the file hashes can be used to download the original APK files from Androzoo (Allix et al. 2016). After downloading the corresponding APKs the binary sequences from the DEX files were extracted and converted to a signal representation using the methodology described in Section 3.1.

To demonstrate the applicability of the proposed methods to other file formats the Microsoft malware classification dataset (Ronen et al. 2018) was used. This dataset contains the raw binaries and dissassembled opcode of EXE files across nine malware families. While this dataset is used to show the applicability of the proposed approach to windows EXE files it is a much smaller scale than the MalNet dataset, containing only 10,868 samples, and thus has less sample variety. Consequently, the dataset has been largely solved with existing models achieving near perfect performance (Gibert et al. 2020). For this reason, the proposed 1D CNNs are compared to their equivalent 2D models on this dataset but are only compared to SOTA approaches on the Malnet dataset.

While other malware classification datasets exist, they cannot be used in this work as they do not provide the original file binaries. Although commonly used the MalImg (Nataraj et al. 2011a) and the VirusMNIST (Noever and Noever 2021) datasets provide only the image representation of the files, making it impossible to convert the samples into a signal representation. Other datasets such as the EMBER dataset (Anderson and Roth 2018) provide the data as a set of tabular features. All of these datasets are also significantly smaller than the MalNet dataset.

4.2 Experimental Details

Models trained on the MalNet dataset utilise the predefined 70/10/20 train, test, and validation sets for each task granularity. During training, models were evaluated at each epoch on the validation set, with the checkpoint exhibiting the best validation performance being selected for final evaluation on the test set. For the Microsoft Malware Classification dataset, due to the unavailability of public test set labels, the training set was split into an 80/20 train-validation split, with validation results being reported.

Two distinct training recipes were employed in this work. The standard recipe was derived from previous studies (Freitas et al. 2021) and is used for a direct comparison with existing literature when evaluating the efficacy of transforming the data into a 1D signal-based representation. It trains the model for 100 epochs using the Adam optimiser with a learning rate of 0.001 and a batch size of 64. Additionally, an improved training recipe was introduced to compare the proposed model's performance against current state-of-the-art (SOTA) approaches. The improved recipe incorporates several enhancements, including linearly scaling class balancing strength by a factor of 0.5, applying weight decay with strength 0.005, as well as label smoothing with an alpha value of 0.1, and using a warm-up cosine schedule for the learning rate. In the improved training recipe, models are initially trained using the stated hyperparameters for 50 epochs. Subsequently, the class balancing weights are removed, and the model is fine-tuned for a further 10 epochs with a learning rate which is reduced by a factor of 10. This improved recipe allows models to achieve improved performance despite training for fewer epochs.

In the remainder of this work ResNet models trained on signal data are referred to as ResNet1D models to distinguish them from traditional image-based ResNets. The standard notation is retained to denote the ResNet's size; for example, ResNet1D152 refers to a ResNet152 model adapted for signal data. The original ResNet block is denoted as the V1 architecture (He et al. 2015), while the bottleneck architecture is referred to as V1.5 (He et al. 2015), and the pre-activation architecture is labelled as V2 (He et al. 2016). SE denotes the inclusion of squeeze-and-excitation modules, and D indicates the use of a deep ResNet stem. For instance, a ResNet1DV2-152D-SE model refers to a ResNet152 model utilizing the pre-activation architecture, trained on signal data, and incorporating both a deep stem and squeeze-and-excitation modules.

4.3 Loss of Information in 2D Images

In this section, the effect of resizing binaries into 1D signals as opposed to 2D images is explored. To provide a quantitative comparison of the resizing methods, the mean squared error (MSE) and signal-to-noise ratio (SNR) was measured at each stage of the resizing process.

To assess resizing noise, the binaries were resized to a length of 65536 for signals or reshaped into a grid and resized to 256×256 for images. The downsampled binary was then upsampled back to its original size, with the MSE and SNR being calculated between the original and the resized binary. Quantisation noise was measured by first resizing the binary in either 1D or 2D and then quantising to the nearest integer, comparing quantised signal to the resized signal. The quantised signals were subsequently upsampled back to their original sizes and compared to the original binaries to measure the total noise incurred from the combination of resizing and quantisation.

The mean resizing and quantisation noise for both images and signals is presented in Table 1 for the MalNet dataset and in Table 2 for the Microsoft Malware Classification dataset. The results indicate that resizing noise is more significant when the data is resized into 2D images. Although on the Microsoft Malware dataset quantising signals introduces more noise than quantising image representations, it should be noted that the signal representations are not quantised when used to train a model, thus only resizing noise is suffered. Consequently, the total noise in the image representations is the cumulation of resizing and quantisation noise, whereas for signals, it is only resizing noise. The results demonstrate that for both Android APKs and Windows' EXE files, the proposed signal representations introduce significantly less noise during preprocessing.

Reduced preprocessing noise does not necessarily guarantee improved performance when training a classification model. To evaluate this, the performance of various 2D CNNs, trained on images, were compared to their 1D equivalent models, trained on signal representations. The performance of both sets of models on the MalNet type dataset is shown in Table 3 with the comparison being extended to the Microsoft Malware Classification dataset in Table 4. It is evident that despite having the same computational requirements and number of parameters, the 1D models outperform their 2D equivalent models, with each model achieving a higher F1 score when trained on 1D signals across both Android APK and Windows' EXE data.

Table 1 Sources of noise in the binary resizing process on the MalNet dataset.

Signal Loss Type	SNR (dB)	MSE
Image Resizing	-17.6461	3757.7163
Image Quantisation	35.9104	1.2992
Image Resizing + Quantisation	-17.6439	3755.8519
Signal Resizing	-16.9099	3238.4049
Signal Quantisation	40.2085	1.1551
Signal Resizing + Quantisation	-16.8438	3240.5584

Table 2 Sources of noise in the binary resizing process on the Microsoft Malware Dataset.

Signal Loss Type	SNR (dB)	MSE
Image Resizing	-19.3484	4937.4840
Image Quantisation	44.6832	0.7755
Image Resizing + Quantisation	-19.3483	4937.3816
Signal Resizing	-18.5801	4133.1473
Signal Quantisation	42.2795	3.2579
Signal Resizing + Quantisation	-18.5871	4138.1813

Table 3 Comparison between 1D and 2D CNNs on type level classification when trained with identical hyperparameters.

Model	F1 Score	Precision	Recall	Parameters (M)	Compute (GFLOPs)
ResNet18	.467	.556	.424	11.2	2.3
ResNet50	.479	.566	.441	23.6	9.9
DenseNet121	.471	.558	.428	7.0	3.6
Densenet169	.477	.573	.433	12.6	4.3
MobileNetV2(x.5)	.460	.547	.424	0.7	0.1
MobileNetV2(x1)	.453	.527	.419	2.28	0.4
ResNet1D18	.482	.598	.432	11.2	2.3
ResNet1D50	.486	.585	.443	23.6	9.9
DenseNet1D121	.499	.574	.466	7.0	3.6
Densenet1D169	.488	.562	.452	12.6	4.3
MobileNet1DV2(x.5)	.482	.548	.458	0.7	0.1
MobileNet1DV2(x1)	.481	.552	.444	2.28	0.4

4.4 Adapting Convolutions to 1D Data

This section investigates methodologies for converting 2D CNNs to 1D models. Two primary techniques were employed: the first preserves the original stride and kernel sizes, while the second squares these values to account for the dimensionality reduction when transitioning to 1D data. Alternatively, either the kernel size or the stride can be squared independently, maintaining the original size of the other parameter.

When the kernel size is squared, this is equivalent to flattening the 2D kernel into a 1D form, thereby preserving the same number of parameters. However, this

Table 4 Comparison of CNN architectures on the Microsoft Malware dataset.

Model	F1 Score	Precision	Recall
ResNet18	.976	.971	.982
ResNet50	.972	.968	.977
DenseNet121	.978	.976	.980
Densenet169	.981	.983	.979
MobileNetV2(x.5)	.956	.960	.953
MobileNetV2(x1)	.958	.946	.982
ResNet1D18	.992	.993	.990
ResNet1D50	.991	.990	.991
DenseNet1D121	.980	.971	.992
Densenet1D169	.985	.980	.991
MobileNet1DV2(x.5)	.984	.985	.982
MobileNet1DV2(x1)	.989	.975	.988

Table 5 Comparison of various approaches to convert 2D CNNs to 1D when training a ResNet18 on MalNet type classification.

Squared Stride	Squared Kernel	F1 Score	Precision	Recall	Parameters (M)	Compute (GFLOPs)
False	False	.406	.598	.406	3.9	11.1
False	True	.471	.561	.434	11.2	32.6
True	False	.466	.573	.420	3.9	0.8
True	True	.482	.598	.432	11.2	2.3

transformation also extends the kernel's receptive field, making it sensitive to longer-range dependencies along the single dimension. Squaring the stride preserves the overall amount of downsampling after each ResNet stage block, thereby maintaining consistent VRAM requirements and computational load per forward pass. In contrast, preserving the original stride reduces the amount of downsampling, consequently increasing the VRAM and computational demands of the model.

In Table 5 the model performance for each combination of 2D to 1D conversion methodology is shown for a ResNet18 model trained on the MalNet type classification dataset. Best performance was achieved when both the kernel sizes and stride values were squared, this also produced a model with the same number of parameters and compute requirements as the original 2D model. Flattening the kernel sizes appeared to have more of an impact on model performance than squaring the stride values. Additionally, squaring the stride improved performance while maintaining the same degree of downsampling as 2D models, meaning that training the model required the same amount of VRAM and compute per forward pass.

4.5 Comparison to Existing Models

The performance of the proposed approach was evaluated using precision, recall, and macro-averaged F1 score across binary, type, and family level classification tasks on the MalNet dataset. A ResNet1DV2-152D-SE model, trained with the improved recipe, was compared to three popular convolutional architectures: ResNet, DenseNet, and EfficientNet, as reported in the MalNet paper (Freitas et al. 2021). Additionally, it

Table 6 Performance comparison of the proposed approach with existing approaches on the MalNet dataset across task granularities.

Model	Binary			Type			Family		
	F1 Score	Precision	Recall	F1 Score	Precision	Recall	F1 Score	Precision	Recall
ResNet1DV2-152D-SE	**.874**	.907	**.846**	**.503**	**.643**	**.453**	**.507**	**.580**	**.480**
SHERLOCK	.854	**.920**	.810	.497	.628	.447	.491	.568	.461
ResNet18	.862	.893	.837	.467	.556	.424	.454	.538	.423
ResNet50	.854	.907	.814	.479	.566	.441	.468	.541	.443
DenseNet121	.864	.900	.834	.471	.558	.428	.461	.529	.438
Densenet169	.864	.890	.841	.477	.573	.433	.462	.545	.434
MobileNetV2(x.5)	.857	.894	.827	.460	.547	.424	.451	.528	.423
MobileNetV2(x1)	.854	.889	.825	.452	.527	.419	.438	.532	.405

was compared to the current SOTA model, SHERLOCK, which employs a ViT-B pretrained using masked autoencoding (Seneviratne et al. 2022b). Due to the recency of the MalNet dataset these baselines encompass all models currently benchmarked.

The performance comparison results, presented in Table 6, demonstrate the strong performance of the proposed model. The results indicate that the proposed model outperforms all baselines across all three classification granularities in terms of F1 score, thus achieving SOTA performance. Furthermore, the proposed model surpasses all baseline models in terms of recall and precision for both type and family classification tasks. While the proposed approach is outperformed by SHERLOCK in precision for binary classification, this can be attributed to the precision-recall trade-off, where a model's recall can be sacrificed to improve its precision and vice versa. While SHERLOCK favours precision, ResNet18 favours recall, making it feasible to individually outperform each model but challenging to surpass both models on both metrics simultaneously. However, the overall performance, as quantified by the F1 score, indicates the proposed model's superior overall performance compared to the baselines in binary classification. Adjusting the model's decision boundaries to make decisions closer to the marginal label distribution could potentially enhance its precision at the expense of recall.

5 Additional Results and Ablations

The proposed ResNet1DV2-152D-SE model was shown to achieve state-of-the-art performance on the MalNet dataset. This section provides additional analysis and examines the contribution of various design decisions to the proposed model's performance. Section 5.1 begins by comparing the performance of various residual block architectures. Next, architectural design choices are evaluated with various ResNet stems and activation functions being compared in Section 5.2 and Section 5.3, respectively. The input representation is analysed in the following sections, with Section 5.4 examining the choice of resampling filter and Section 5.5 investigating the length of the input signals. Performance at different model sizes is compared in Section 5.5. Finally, Section 5.7 examines the trade-off between precision and recall by analysing the precision-recall (PR) curves of ResNet1DV2-152D-SE and SHERLOCK.

Table 7 Performance comparison of different residual blocks when used in a ResNet1D18 trained on MalNet type classification.

Model Architecture	F1 Score	Precision	Recall	Parameters (M)	Compute (GFLOPs)
ResNetV1	.518	.645	.464	11.2	2.3
ResNetV1.5	.520	.669	.457	14.0	5.4
ResNetV2	.510	.619	.461	14.4	5.4
ResNetV2SE	.512	.632	.459	21.0	5.4

Table 8 Performance comparison of stems when used in a ResNet1DV2-152-SE trained on MalNet type classification.

Stem Architecture	F1 Score	Precision	Recall	Parameters (M)	Compute (GFLOPs)
Standard	.506	.591	.465	106.8	29.4
Deep	.503	.643	.453	106.8	29.8

5.1 Choice of ResNet Architecture

The performance of various residual block architectures is compared in Table 7. Specifically, the original ResNet block is evaluated alongside the bottleneck block, as well as the pre-activation block, both with and without squeeze-and-excitation modules, when incorporated into a ResNet1D18 model. While the ResNetV1.5 architecture achieved the best performance; adding SE layers to the ResNetV2 architecture resulted in an improvement in the model's F1-score and precision. This enhancement resulted in a marginal increase in computational overhead, but a large increase in model parameter count.

5.2 Effect of Model Stem

The effect of replacing the initial convolution at the start of the ResNet with the deep stem containing multiple convolutions is shown in Table 8, where a ResNet1DV2-152-SE model was benchmarked on MalNet type classification with both a deep stem and with the original convolution. The deep stem improved the precision of the model, although this came at the expense of recall and F1 score. Additionally, the deep stem introduced a marginal increase in the number of parameters and a slight computational overhead.

5.3 Choice of Activation Function

The proposed model replaces the ReLU function, which is typically used with ResNet architecture, with the GELU activation function. The impact of this on model performance is shown in Table 9, where a ResNet1DV2-152D-SE model is trained using each activation function with the improved training recipe on MalNet type classification. The GELU activation function produced a high precision model, however, this came at the expense of recall, F1 score, and a marginal increase in computational overhead. The reason for this performance difference may be due to the GELU activation function learning smoother decision boundaries resulting in more reliable predictions.

Table 9 Performance comparison of ReLU and GELU activations when used in a ResNet1DV2-152D-SE for MalNet type classification.

Activation	F1 Score	Precision	Recall	Parameters (M)	Compute (GFLOPs)
ReLU	.509	.580	.473	106.8	29.8
GELU	.503	.643	.453	106.8	29.8

Table 10 Performance comparison of Resnet1D18 models when trained on signals generated using various resizing filters.

Activation	F1 Score	Precision	Recall
Lanczos	.518	.645	.464
Nearest Neighbour	.450	618	391
Cubic	.449	.623	.387
Linear	.287	.517	.227

5.4 Choice of Resampling Filter

The Lanczos filter employed by this work has been widely adopted in prior works utilsing byteplot images; however, it was originally developed for natural images and thus may not be the ideal resampling filter for byte data. To evaluate its effectiveness, a ResNet1D18 model was trained MalNet type level classification using signal representations resampled using various filters. The results, given in Table 10, show that Lanczos resampling outperforms nearest neighbour, linear, and cubic resampling.

The superior performance of Lanczos resampling suggests that discriminative features in byteplot signals may reside in higher frequency components. Since the Lanczos filter preserves more high-frequency content due to its sinc-based formulation, it is better able to retain fine-grained variations in byte transitions that may be informative for classification. This implies that abrupt changes in byte values, which manifest as high-frequency components in the signal domain, are likely important for distinguishing malware types.

5.5 Choice of Signal Length

In this work's main evaluation, signal representations were generated by resizing file binaries to a length of 65536. This was chosen to allow for comparison with existing models trained on the MalNet dataset which consists of byteplot images resized to 256×256 (65536) pixels. In Table 11 the performance of a ResNet1D18 model on MalNet type classification is shown when trained on malware signals resized to various lengths. It can be seen that the F1 score, precision, and recall monotonically increase with signal length. This trend can be attributed to reduced information loss in longer signals, which retain more of the original binary's structure. However, this benefit comes at the cost of increased storage requirements as well as higher computational demands during both training and inference.

Table 11 Performance comparison ResNet1D18 models when trained on signal representations of varying length on MalNet type classification.

Signal Length	F1 Score	Precision	Recall	Compute (GFLOPs)
256	.018	.017	.021	0.01
1024	.034	.169	.034	0.03
4096	.106	.308	.087	0.1
16384	.174	.387	.135	0.6
65536	.518	.645	.464	2.3

Table 12 Performance comparison of varying sizes of ResNet1D when trained on MalNet type classification.

Model	Parameters (M)	Compute (GFLOPs)	F1 Score	Precision	Recall
ResNet1D18	11.2	2.3	.518	.645	.464
ResNet1D34	21.3	4.7	.520	.643	.463
ResNet1D50	23.6	9.9	.508	.635	.452
ResNet1D101	42.6	19.6	.504	.655	.444
ResNet1D152	58.2	29.3	.501	.629	.450

5.6 Parameter Sensitivity Analysis

The impact of the number of parameters on model performance is shown in Table 12, where the performance of each size of ResNet model is evaluated on MalNet type classification. Interestingly, small models appear to outperform their larger counterparts. This is hypothesised to be due to resizing file binaries significantly reducing the information in the input signal. As a consequence, the model performance becomes heavily dependent on signal length, with more parameters only resulting in increased overfitting.

5.7 Precision Recall Trade-off

In Section 4.5, it was noted that while ResNet1DV2-152D-SE outperforms SHERLOCK in terms of recall, it exhibits lower precision. This was attributed to the trade-off between precision and recall, whereby altering a model's decision threshold can improve precision at the expense of recall and vice-versa. This trade-off can be analysed using the PR curve, which visualises model performance across different decision thresholds. To focus on practical deployment scenarios—where maintaining a reasonable level of threat coverage is essential—the analysis was restricted to the region where recall is greater than or equal to 0.5. The PR curves of the proposed model and SHERLOCK are shown in Figure 7.

It can be seen that ResNet1DV2-152D-SE has a greater area under the PR curve than SHERLOCK indicating superior overall performance across varying decision thresholds. In particular, the decision threshold of ResNet1DV2-152D-SE can be adjusted to give a recall value greater than 0.9 without a significant decrease in precision, whereas operating SHERLOCK in this region would result in significant precision degradation.

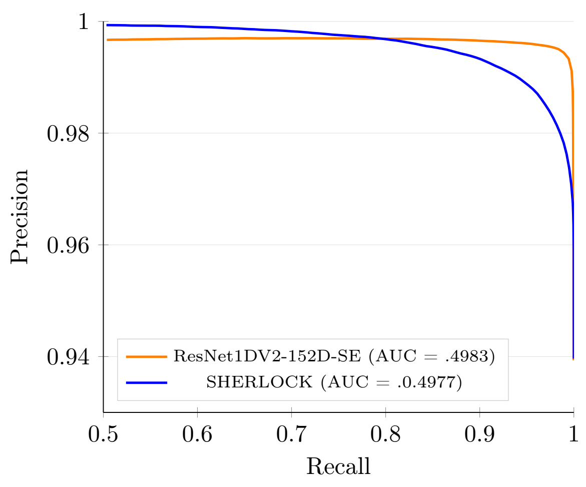


Fig. 7 Precision-recall curves of ResNet1DV2-152D-SE and SHERLOCK on MalNet binary classification.

In practical scenarios, achieving high recall ensures that more malware samples are successfully detected, reducing the likelihood of missed threats. However, this often comes at the cost of lower precision, leading to a higher false positive rate and less reliable predictions. To balance these competing objectives, the model's decision threshold would typically be tuned on a held-out validation set to maximise recall while maintaining an acceptable false positive rate. This approach ensures the model remains effective in detecting malware without overwhelming analysts with excessive false alarms.

6 Discussion and Conclusions

Previous applications of deep neural networks in malware classification overwhelmingly preprocess the raw binaries into image representations known as byteplots. Computer vision approaches such as CNNs and vision transformers trained on these byteplots have shown competitive performance on malware classification tasks while also being resilient to obfuscation techniques such as packing and polymorphism. However, the byteplot representations used to train these models were initially designed to assist human analysts in quickly identifying regions of bytecode of interest for manual inspection and are not the ideal representation for training machine learning models. This work instead proposes a change in paradigm where malware binaries are resized in 1D into signal representations instead of images. These signal representations do not require heuristic reshaping rules to convert them into a 2D structure, furthermore, once resized they do not require quantisation into integer pixel values like images.

This work quantitatively shows that by excluding the heuristic reshaping and quantisation steps, the proposed signal representations contain more information than their image-based counterparts. Furthermore, by converting 2D computer vision models to operate on 1D data, an equivalent model can be trained which has the same number of parameters and compute requirement as its 2D counterpart but boasts superior performance. This is shown to be true for both binaries obtained from android DEX files and from Windows' EXE files through a series of experiments on the MalNet dataset and the Microsoft Malware classification dataset.

To demonstrate the proposed signal representation's effectiveness, a novel 1D CNN architecture was proposed. The CNN was based on the ResNetV2 architecture, however, the 2D convolutions were replaced with 1D convolutions by the squaring the kernel sizes and stride values. Additionally, squeeze-and-excitation blocks were added to the model and the ReLU activation function was replaced with GELU. This model was compared to state-of-the-art approaches in binary, type and family level classification on the large-scale MalNet dataset. The proposed model achieved state-of-the-art performance with F1 scores of 0.974, 0.503, and 0.503 on binary, type, and family level classification respectively.

These results pave the way for future works to use signal-based models instead of the current image-based models. By providing a method to adapt existing 2D CNNs to operate on 1D data it is hoped that researchers favour using 1D equivalent models for improved performance. In the future domain specific resampling methods could be used to generate the signal representations instead of Lanczos resampling, which was primarily intended for images. By preserving more relevant information from the original binaries in the signal representations the performance of the 1D models could be further improved. Additionally, it is hoped that future datasets will provide the raw binaries of samples. This would encourage research into alternative file representations, and allow the proposed approach to be evaluated on a greater variety of malware: for example cross-platform and mixed-format malware.

Declarations

Conflicts of Interest/Competing Interests

The authors declare that they have no competing interests.

Availability of Data and Materials

The datasets used for this research are publically available.

References

A. DK, P. V, Yerima SY, et al (2023) Obfuscated malware detection in iot android applications using markov images and cnn. IEEE Systems Journal 17(2):2756–2766. https://doi.org/10.1109/JSYST.2023.3238678

Abusitta A, Li MQ, Fung BC (2021) Malware classification and composition analysis: A survey of recent developments. Journal of Information Security and Applications 59:102828. https://doi.org/https://doi.org/10.1016/j.jisa.2021.102828, URL https://www.sciencedirect.com/science/article/pii/S2214212621000648

Allix K, Bissyandé TF, Klein J, et al (2016) Androzoo: Collecting millions of android apps for the research community. In: Proceedings of the 13th International Conference on Mining Software Repositories. ACM, New York, NY, USA, MSR '16, pp 468–471, https://doi.org/10.1145/2901739.2903508, URL http://doi.acm.org/10.1145/2901739.2903508

Alsulami B, Mancoridis S (2018) Behavioral malware classification using convolutional recurrent neural networks. In: 2018 13th International Conference on Malicious and Unwanted Software (MALWARE), pp 103–111, https://doi.org/10.1109/MALWARE.2018.8659358

Anderson B, Storlie C, Lane T (2012) Improving malware classification: bridging the static/dynamic gap. In: Proceedings of the 5th ACM Workshop on Security and Artificial Intelligence. Association for Computing Machinery, New York, NY, USA, AISec '12, p 3–14, https://doi.org/10.1145/2381896.2381900, URL https://doi.org/10.1145/2381896.2381900

Anderson HS, Roth P (2018) EMBER: An Open Dataset for Training Static PE Malware Machine Learning Models. ArXiv e-prints arXiv:1804.04637 [cs.CR]

Bovenzi G, Cerasuolo F, Montieri A, et al (2022) A comparison of machine and deep learning models for detection and classification of android malware traffic. In: 2022 IEEE Symposium on Computers and Communications (ISCC), pp 1–6, https://doi.org/10.1109/ISCC55528.2022.9912986

Chong X, Gao Y, Zhang R, et al (2022) Classification of malware families based on efficient-net and 1d-cnn fusion. Electronics 11(19):3064

Conti G, Dean E, Sinda M, et al (2008) Visual reverse engineering of binary and data files. In: Goodall JR, Conti G, Ma KL (eds) Visualization for Computer Security. Springer Berlin Heidelberg, Berlin, Heidelberg, pp 1–17

Conti G, Bratus S, Shubina A, et al (2010) Automated mapping of large binary objects using primitive fragment type classification. Digital Investigation 7:S3–S12. https://doi.org/https://doi.org/10.1016/j.diin.2010.05.002, URL https://www.sciencedirect.com/science/article/pii/S1742287610000290, the Proceedings of the Tenth Annual DFRWS Conference

Daly G, Fieldsend J, Hassall G, et al (2023) Data-driven plasma modelling: Surrogate collisional radiative models of fluorocarbon plasmas from deep generative autoencoders. Machine Learning: Science and Technology 4. https://doi.org/10.1088/2632-2153/aced7f

Damasevicius R, Venčkauskas A, Toldinas J, et al (2021) Ensemble-based classification using neural networks and machine learning models for windows pe malware detection. Electronics 10(4). https://doi.org/10.3390/electronics10040485, URL https://www.mdpi.com/2079-9292/10/4/485

Deng X, Mirkovic J (2022) Polymorphic malware behavior through network trace analysis. pp 138–146, https://doi.org/10.1109/COMSNETS53615.2022.9668396

Dib M, Torabi S, Bou-Harb E, et al (2021) A multi-dimensional deep learning framework for iot malware classification and family attribution. IEEE Transactions on Network and Service Management 18(2):1165–1177. https://doi.org/10.1109/TNSM.2021.3075315

Freitas S, Dong Y, Neil J, et al (2021) A large-scale database for graph representation learning. arXiv:2011.07682

Gennissen J, Blasco J (2017) Gamut : Sifting through images to detect android malware. URL https://api.semanticscholar.org/CorpusID:44430018

Gibert D, Mateu C, Planes J (2020) Hydra: A multimodal deep learning framework for malware classification. Computers & Security 95:101873. https://doi.org/https://doi.org/10.1016/j.cose.2020.101873, URL http://www.sciencedirect.com/science/article/pii/S0167404820301462

Hasegawa C, Iyatomi H (2018) One-dimensional convolutional neural networks for android malware detection. In: 2018 IEEE 14th International Colloquium on Signal Processing & Its Applications (CSPA), pp 99–102, https://doi.org/10.1109/CSPA.2018.8368693

He K, Zhang X, Ren S, et al (2015) Deep residual learning for image recognition. arXiv:1512.03385

He K, Zhang X, Ren S, et al (2016) Identity mappings in deep residual networks. arXiv:1603.05027

He T, Zhang Z, Zhang H, et al (2018) Bag of tricks for image classification with convolutional neural networks. arXiv:1812.01187

Hendrycks D, Gimpel K (2023) Gaussian error linear units (gelus). arXiv:1606.08415

Hu J, Shen L, Sun G (2018) Squeeze-and-excitation networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)

Huang THD, Kao HY (2018) R2-d2: Color-inspired convolutional neural network (cnn)-based android malware detections. arXiv:1705.04448

Kalash M, Rochan M, Mohammed N, et al (2018) Malware classification with deep convolutional neural networks. In: 2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS), pp 1–5, https://doi.org/10.1109/NTMS.2018.8328749

Kim HI, Kang M, Cho SJ, et al (2022) Efficient deep learning network with multi-streams for android malware family classification. IEEE Access 10:5518–5532. https:

//doi.org/10.1109/ACCESS.2021.3139334

Kim J, Paik JY, Cho ES (2023) Attention-based cross-modal cnn using non-disassembled files for malware classification. IEEE Access 11:22889–22903. https://doi.org/10.1109/ACCESS.2023.3253770

Liu L, Wang Bs, Yu B, et al (2017) Automatic malware classification and new malware detection using machine learning. Frontiers of Information Technology & Electronic Engineering 18(9):1336–1347. https://doi.org/10.1631/FITEE.1601325, URL https://doi.org/10.1631/FITEE.1601325

Llauradó DG (2016) Convolutional neural networks for malware classification. URL https://api.semanticscholar.org/CorpusID:22879106

Lu Q, Zhang H, Kinawi H, et al (2022) Self-attentive models for real-time malware classification. IEEE Access 10:95970–95985. https://doi.org/10.1109/ACCESS.2022.3202952

McLaughlin N, Martinez del Rincon J, Kang B, et al (2017) Deep android malware detection. In: Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy. Association for Computing Machinery, New York, NY, USA, CODASPY '17, p 301–308, https://doi.org/10.1145/3029806.3029823, URL https://doi.org/10.1145/3029806.3029823

Mekruksavanich S, Jitpattanakul A (2022) Deep residual network for smartwatch-based user identification through complex hand movements. Sensors 22:3094. https://doi.org/10.3390/s22083094

Mohaisen A, West AG, Mankin A, et al (2014) Chatter: Classifying malware families using system event ordering. In: 2014 IEEE Conference on Communications and Network Security, pp 283–291, https://doi.org/10.1109/CNS.2014.6997496

Nataraj L, Karthikeyan S, Jacob G, et al (2011a) Malware images: visualization and automatic classification. In: Proceedings of the 8th International Symposium on Visualization for Cyber Security. Association for Computing Machinery, New York, NY, USA, VizSec '11, https://doi.org/10.1145/2016904.2016908, URL https://doi.org/10.1145/2016904.2016908

Nataraj L, Yegneswaran V, Porras P, et al (2011b) A comparative assessment of malware classification using binary texture analysis and dynamic analysis. In: Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence. Association for Computing Machinery, New York, NY, USA, AISec '11, p 21–30, https://doi.org/10.1145/2046684.2046689, URL https://doi.org/10.1145/2046684.2046689

Noever D, Noever SEM (2021) Virus-mnist: A benchmark malware dataset. arXiv:2103.00602

Paik JY, Jin R, Cho ES (2022) Malware classification using a byte-granularity feature based on structural entropy. Computational Intelligence 38(4):1536–1558. https://doi.org/https://doi.org/10.1111/coin.12521, URL https://onlinelibrary.wiley.com/doi/abs/10.1111/coin.12521, https://onlinelibrary.wiley.com/doi/pdf/10.1111/coin.12521

Priya V, Sathya Sofia A (2023) Review on malware classification and malware detection using transfer learning approach. In: 2023 5th International Conference on Smart Systems and Inventive Technology (ICSSIT), pp 1042–1049, https://doi.org/10.1109/ICSSIT55814.2023.10061076

Rezende ERS, Ruppert GCS, de Carvalho TJ, et al (2018) Malicious software classification using vgg16 deep neural network's bottleneck features. URL https://api.semanticscholar.org/CorpusID:21388434

Ronen R, Radu M, Feuerstein C, et al (2018) Microsoft malware classification challenge. arXiv:1802.10135

Safa H, Nassar M, Rahal Al Orabi WA (2019) Benchmarking convolutional and recurrent neural networks for malware classification. In: 2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC), pp 561–566, https://doi.org/10.1109/IWCMC.2019.8766515

Sathyanarayan V, Kohli P, Bezawada B (2008) Signature generation and detection of malware families. pp 336–349, https://doi.org/10.1007/978-3-540-70500-0_25

Schofield M (2021) Comparison of malware classification methods using convolutional neural network based on api call stream. International Journal of Network Security & Its Applications (IJNSA) 13(2). Available at SSRN: https://ssrn.com/abstract=3822934

Schofield M, Alicioglu G, Binaco R, et al (2021) Convolutional neural network for malware classification based on api call sequence. pp 85–98, https://doi.org/10.5121/csit.2021.110106

Schultz MG, Eskin E, Zadok E, et al (2001) Data mining methods for detection of new malicious executables. Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy p 38 – 49. URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-0034838197&partnerID=40&md5=388ce70cc320d3c766cada92e1b12ccf, cited by: 840

Seneviratne S, Shariffdeen R, Rasnayaka S, et al (2022a) Self-supervised vision transformers for malware detection. IEEE Access 10:103121–103135. https://doi.org/10.1109/access.2022.3206445, URL http://dx.doi.org/10.1109/ACCESS.2022.3206445

Seneviratne S, Shariffdeen R, Rasnayaka S, et al (2022b) Self-supervised vision transformers for malware detection. IEEE Access 10:103121–103135. https://doi.org/10.

1109/access.2022.3206445, URL http://dx.doi.org/10.1109/ACCESS.2022.3206445

Votipka D, Rabin S, Micinski K, et al (2020) An observational investigation of reverse Engineers' processes. In: 29th USENIX Security Symposium (USENIX Security 20). USENIX Association, pp 1875–1892, URL https://www.usenix.org/conference/usenixsecurity20/presentation/votipka-observational

Wang L, Sun J, Luo X, et al (2021) Transferable features from 1d-convolutional network for industrial malware classification. Computer Modeling in Engineering & Sciences 130:1003–1016. https://doi.org/10.32604/cmes.2022.018492

Yeboah PN, Musah HBB (2022) Nlp technique for malware detection using 1d cnn fusion model. Security and Communication Networks 2022:2957203. https://doi.org/10.1155/2022/2957203, URL https://doi.org/10.1155/2022/2957203

Zyout M, Shatnawi R, Najadat H (2023) Malware classification approaches utilizing binary and text encoding of permissions. International Journal of Information Security 22(6):1687–1712. https://doi.org/10.1007/s10207-023-00712-z, URL https://doi.org/10.1007/s10207-023-00712-z