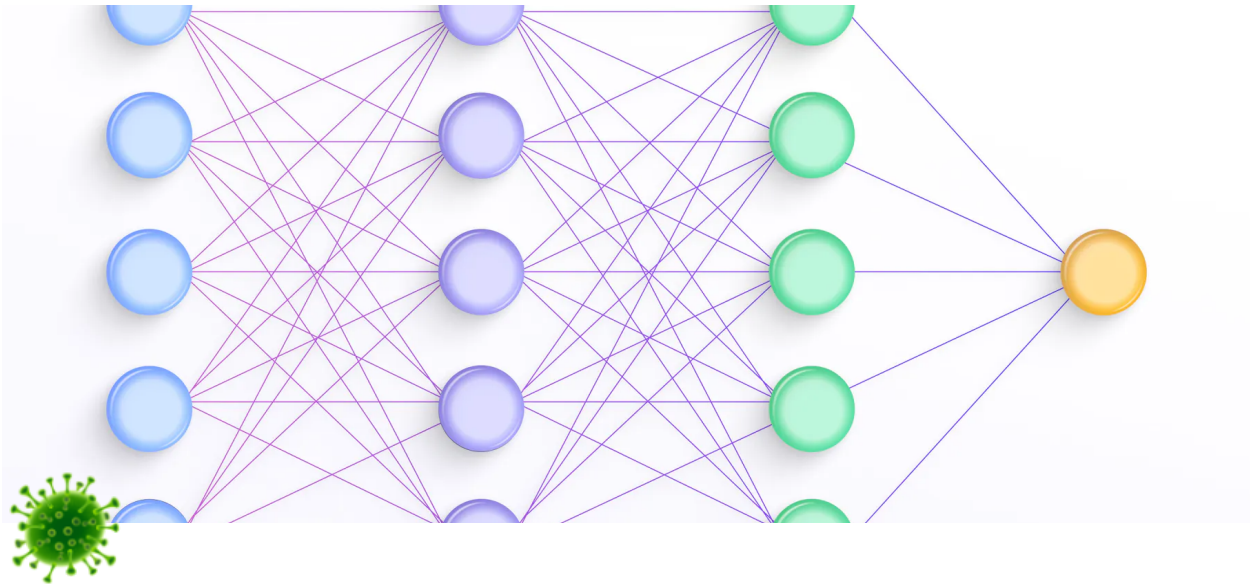


Applying CNNs to Malware Classification

> ⚠ *“Remember, folks, clicking on suspicious links for free stuff is the digital equivalent of taking candy from strangers. Spoiler: It doesn't end with free candy, just tears and tech support.” - ChatGPT, when asked to generate a funny quote about malware (nobody laughed)*

Students - Group 21

Arnob Chowdhury - <A.Chowdhury-2@student.tudelft.nl> - 5977045

Isidoro Tamassia - <I.Tamassia@student.tudelft.nl> - 6054765

Maurya Goyal - <M.Goyal@student.tudelft.nl> - 6003435

Primakov Chungkham - <P.Chungkham@student.tudelft.nl> - 5992532

Work division

Arnob: Code reproduction

Isidoro: Data preprocessing

Maurya: Evaluation

Primakov: Training and Testing

GitHub

https://github.com/arc-arnob/malimg-cnn

Introduction

This article explores the critical importance of classifying malware into their respective families within the domain of cybersecurity, and how Convolutional Neural Networks (CNNs) can be an effective tool to achieve this goal.

Files that exhibit similar harmful activities but are frequently altered or masked to appear as distinct entities pose a challenge. For the analysis and categorization of these numerous files, it's essential to organize them into clusters and determine their specific families based on their behavioral characteristics. By identifying the family to which a piece of malware belongs, cybersecurity tools can utilize known signatures, behaviors, and heuristics more effectively, thereby enabling faster and more accurate threat detection.

Machine learning can be a powerful tool for detecting patterns in the behavior of malicious files, even when they are frequently modified or obfuscated to appear different. Instead of directly dealing with raw binary files, ML approaches to malware detection typically start by extracting features from them, offering an abstract representation of the program. These extracted features are then used for training the model. The ones used for this purpose generally fall into two main categories: static features, which can be analyzed without executing the malware, and dynamic features, which are observed through the behavior of the malware when it's running.

Moreover, data visualization techniques can be exploited to help with the analysis of malware, and this is the direction that we will analyze in this work. The translation of the original binary files into a visual representation, e.g. either a grayscale or RGB image, can allow us to employ CNNs, a family of architectures

that have been proven to effectively recognize patterns and yield compelling results. For instance, if we want to map a binary file into a grayscale image, then each byte is to be considered as a single pixel within the image. Following this, bytes need to be arranged into a two-dimensional (2-D) array of values between 0 and 255, with 0 representing black and 255 representing white. This technique allowed for the creation of datasets like the ones that we experiment on in this work.

Despite there exist different neural methods, including the aforementioned CNNs, capable of achieving impressive results in terms of classification accuracy, the majority of these approaches have only been tested on either small datasets or datasets that have been specifically built for the task and are not comparable, in terms of structure and complexity of the malware, to real-word instances. For this reason, we reproduced the work by Gibert et al. in their "**Using convolutional neural networks for classification of malware represented as images** " [1] paper and then applied the same approach to a different dataset that represents a real collection of dangerous malware extracted from the DarkNet. By doing this, we expect to understand how a well-established SOTA model in this field can be applied to a truly challenging task, confirming or denying its effectiveness.

The next sections are structured as follows:

- **Related Work** → An overview of previous works in the research literature that tried to apply different ML methods to perform malware categorization
- **Datasets** → A description of the two datasets that we used for our training and subsequent experiments
- **Methodology and Implementation** → A description of the methodology of the paper we reproduced, the way we adapted it to a new, more challenging dataset, and the way we implemented it in practice
- **Experiments and Results** → An explanation of how we conducted our experiments and which results we were able to obtain
- **Evaluation and Discussion** → A critical analysis of the results achieved and a reflection on potential enhancements for future endeavors.

Related Work

As mentioned in the Introduction, the main work related to our project is the paper "**Using convolutional neural networks for classification of malware represented as images** " [1] by Gibert et al., of which we reproduced the utilized model, code, and main experiments. Here, the authors train a CNN architecture on the MalImg dataset, trying different hyperparameter settings as well as different sizes of the input images.

Regarding the visualization and processing of malware, there have been several methods presented in the recent literature. For instance, Sorokin et al. [2] represented a binary file using its structural entropy, obtained by dividing it into non-overlapping chunks and computing the entropy of each chunk. Moreover, Nataraj et al. [3] applied image processing techniques to classify malware according to its visualization as grayscale images, by extracting GIST features from them and training a k-nearest neighbor model. This last idea of exploiting grayscale images is the one that inspired the work of the paper we reproduced.

Datasets

MalImg

Introduced by Nataraj et al., it comprises 9,342 grayscale images representing 25 distinct families of malware. This collection includes instances of malicious software compressed using UPX, spanning a variety of families, including but not limited to Yuner.A, VB.AT, Malex.gen!J, Autorun.K, and Rbot!gen. Since these images are of different sizes, the authors of the paper resized them to a fixed dimension to use them as input to the model more easily. They experimented with different sizes, including 64×64, 128×128 and 256×256. In our case, we decided to resize them to 128×128 as it seemed to be the best choice, taking into account both the amount of remaining information and the feasibility of the training with our available resources. In this case, the classes are balanced enough, so we did not apply further simplifications and preprocessing.

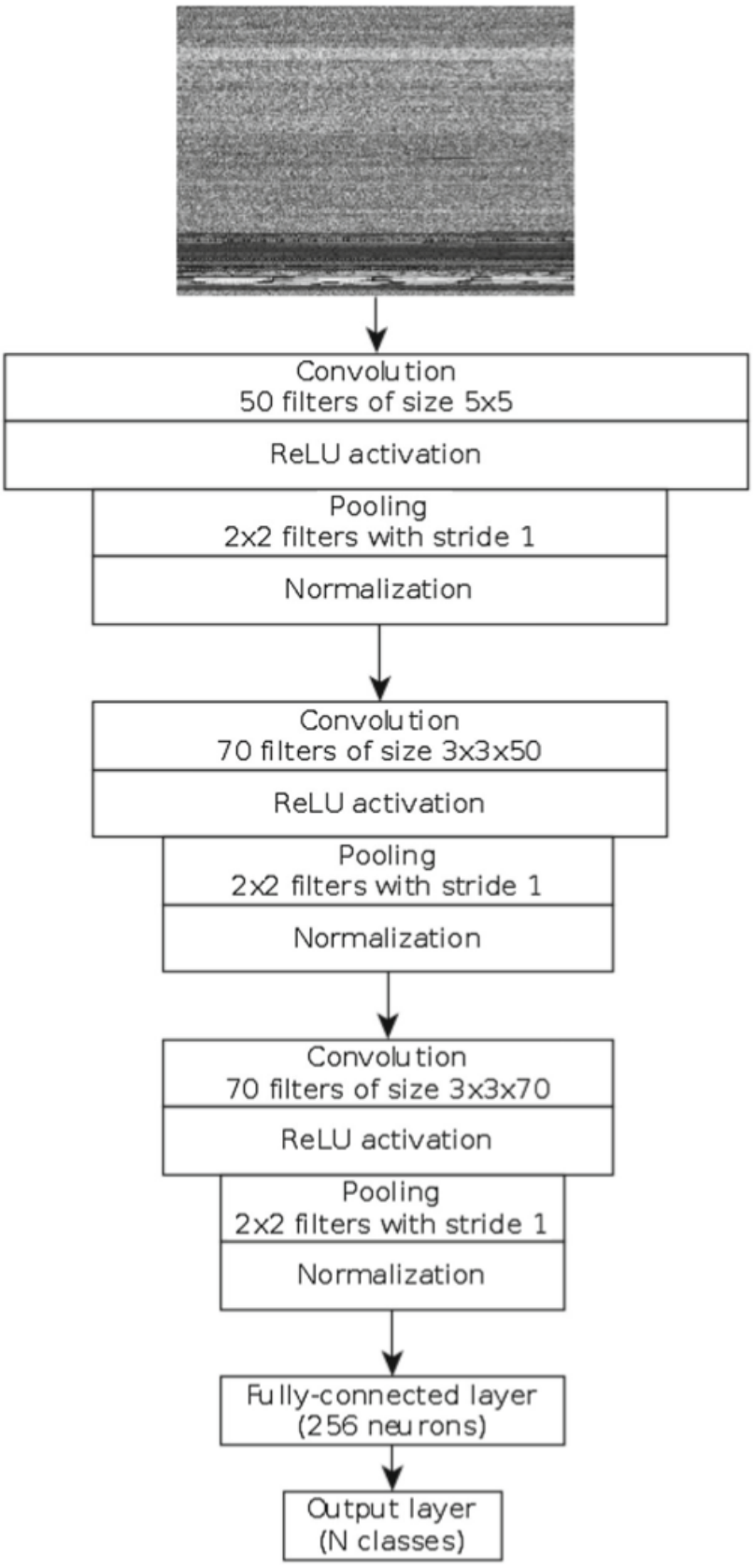
DarkNet

A complex, highly imbalanced dataset of malware extracted from the DarkNet, divided into 548 classes with a variable number of samples, weighting approximately 150GB in total. The malware binaries are translated into RGB images, therefore, we decided to convert them to grayscale images in order to have a more fair comparison with our experiments on MalImg. Again, we need to apply a resizing, and we opt for the same dimensions that we used in the previous case, 128×128. This allows us to train the model using the same resources as before since the weight drops significantly. Furthermore, since the cardinality of the classes varies from a single sample to thousands, we decided to drop the classes that present fewer than three samples, since we considered such few samples to be not enough representative of the whole malware category.

Methodology and Implementation

The main advantages of using CNNs and converting the byte code to images are as follows:

- Different sections of the binary codes can be well differentiated and extracting discriminative local features becomes easier.
- For malware, small parts of the code are reused which makes it easier to identify them and classify them into different families.
- It allows the detection of small changes while retaining the global structure of samples belonging to the same family.
- CNNs and max pooling are invariant to translation.


Convolution
50 filters of size 5x5
ReLU activation
Pooling
2x2 filters with stride 1
Normalization
Convolution
70 filters of size 3x3x50
ReLU activation
Pooling
2x2 filters with stride 1
Normalization
Convolution
70 filters of size 3x3x70
ReLU activation
Pooling
2x2 filters with stride 1
Normalization
Fully-connected layer
(256 neurons)
Output layer
(N classes)

Model Architecture: For the training data, we convert all the images to 128×128 size and to grayscale for both datasets to allow for a fair comparison. The model architecture comprises 3 convolutional layers followed by one fully-connected layer (for classification). The input of the network is a malicious program represented as a gray-scale image. The output of the network is the predicted class of the malware sample, for which we apply a softmax, and the output depends on the families (classes) present in the dataset.

To prevent overfitting, we apply dropouts as a regulariser, which randomly drops a proportion of p units during forward propagation and prevents the co-adaptation between neurons. The author of the paper tried using dropouts in every layer and found that they gave the best results when applied only to the last fully connected and output layers. Hence, we followed the same approach and applied dropouts to the fully connected layer and the output layer.

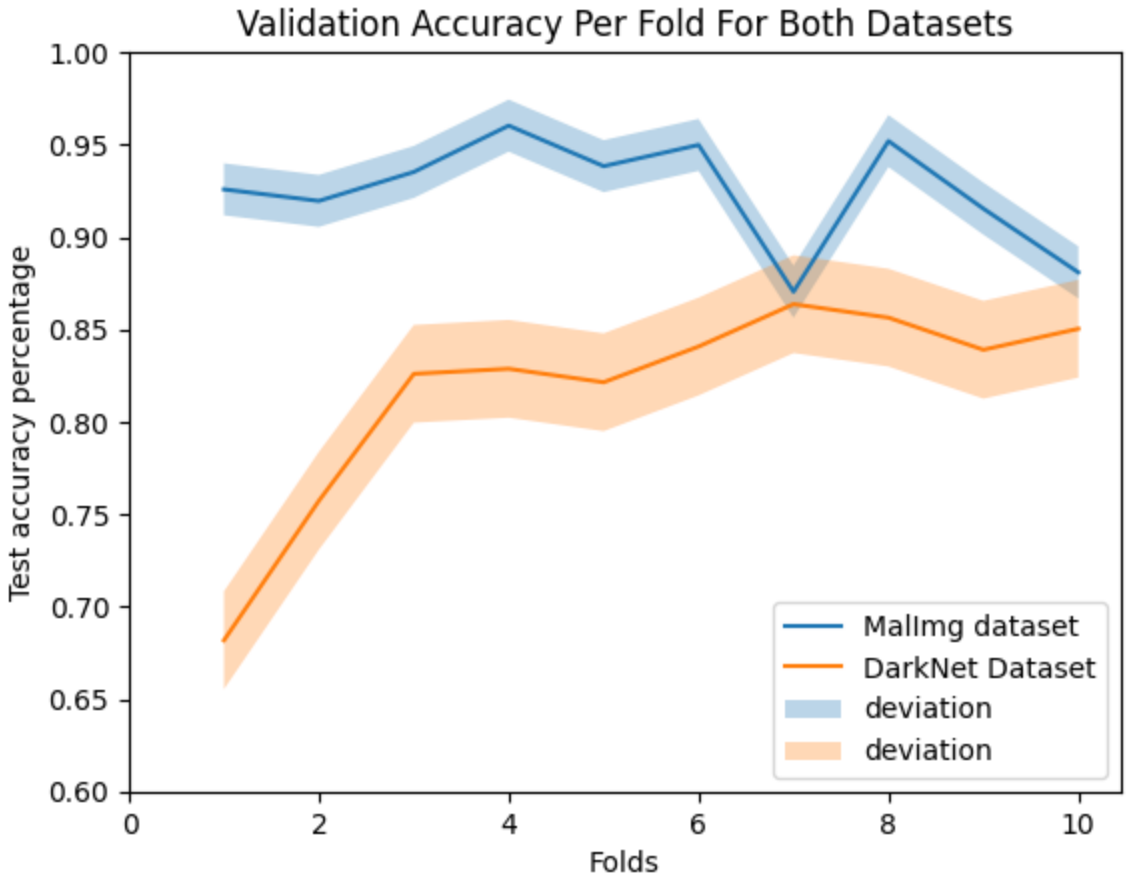
Since the classes were not uniformly distributed we used stratified k folds to split the dataset into test and train, which preserves the distribution.

For the optimizer, we followed the authors' choice and used the Adam optimizer, while as a loss function, we employed the Keras sparse categorical cross-entropy which just applies a cross-entropy loss on integer labels, since differently from the authors, we did not convert the labels to a one-hot representation, but still wanted to apply the same loss function to accurately reproduce their results on MalImg. Another way to do so would have been turning the labels into a one-hot representation and using the alternative categorical cross-entropy, which instead handles one-hot encoding.

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 124, 124, 50)      1300

 max_pooling2d (MaxPooling2  (None, 62, 62, 50)        0
 D)

 batch_normalization (Batch  (None, 62, 62, 50)        200
 Normalization)

 conv2d_1 (Conv2D)           (None, 60, 60, 70)        31570

 max_pooling2d_1 (MaxPoolin  (None, 30, 30, 70)        0
 g2D)

 batch_normalization_1 (Bat  (None, 30, 30, 70)        280
 chNormalization)

 conv2d_2 (Conv2D)           (None, 28, 28, 70)        44170

 max_pooling2d_2 (MaxPoolin  (None, 14, 14, 70)        0
 g2D)

 batch_normalization_2 (Bat  (None, 14, 14, 70)        280
 chNormalization)

 flatten (Flatten)           (None, 13720)             0

 dropout (Dropout)           (None, 13720)             0

 dense (Dense)               (None, 256)               3512576

 dropout_1 (Dropout)         (None, 256)               0

 dense_1 (Dense)             (None, 25)                6425

=================================================================
Total params: 3596801 (13.72 MB)
Trainable params: 3596421 (13.72 MB)
Non-trainable params: 380 (1.48 KB)
_________________________________________________________________
```

Experiments and Results

The main aim of the project was to see how well the state-of-the-art models perform on an actual dataset containing malware images. We used the same SOTA architecture to train both the models and then we compared the validation accuracies for each fold of the datasets and also the MAE (Mean Absolute Error).



Plot 1: Validation Accuracy v/s Fold for both the datasets

Our results are summarized in the reported plots above. In Plot 1, we can see that the SOTA model performs way better on the MalImg dataset whereas the

performance on the DarkNet dataset is not as good. For the MalImg dataset, it reaches an average accuracy of 92.4% with a standard deviation of 0.02%, while the average accuracy for the DarkNet dataset is 81.6% with a standard deviation of 0.05%. For the MalImg dataset, the mean MAE was 0.465 and for the DarkNet dataset, it was 27.6.

These numbers underscore a significant disparity in the performance of state-of-the-art (SOTA) models when applied to newly introduced real-life datasets. It highlights that while these models excel on meticulously curated, tailored datasets, their performance significantly diminishes when tested with real-world observations.

Evaluation and Discussion

The results discussed here suggest many paths for future study. It is important to focus on creating models that work well not just on carefully chosen data, but also in real-life situations. This necessitates a concerted exploration of novel architectures, training methodologies, and regularization techniques tailored to increase the robustness and generalization capabilities of machine learning models.

Moreover, there is a great chance for more research on the recently introduced dataset. This could involve thoroughly examining it to find any biases, unusual patterns, or mistakes in the data that might affect how well the model works. Furthermore, given the absence of hyperparameter tuning in the initial analysis, future research endeavors can focus on optimizing model performance through meticulous parameter tuning.

Additionally, removing the final layer and keeping the model's layers until the second last one can help identify the important features in malware images (also referred to as the 'DNA' of the malware). Further research can be done to identify the DNAs for all these different families of malware.

By pursuing these avenues of inquiry, researchers could advance our understanding of model behavior in real-world contexts, enhance model

adaptability, and contribute to the ongoing evolution of machine learning methodologies for malware detection/classification.

References

1. Gibert, D., Mateu, C., Planes, J. *et al.* Using convolutional neural networks for classification of malware represented as images. *J Comput Virol Hack Tech* **15**, 15–28 (2019). https://doi.org/10.1007/s11416-018-0323-0
2. Sorokin, I.: Comparing files using structural entropy. J. Comput. Virol. **7**(4), 259 (2011)
2. Nataraj,L.,Karthikeyan,S.,Jacob,G.,Manjunath,B.S.:Malware images: visualization and automatic classification. In: Proceedings of the 8th International Symposium on Visualization for Cyber Security, VizSec '11, pp. 4:1–4:7. ACM, New York, NY, USA (2011)