SAPIENZA
UNIVERSITÀ DI ROMA

DEPARTMENT OF COMPUTER SCIENCE

Malware Detection with malware images using CNN techinques

AI LAB: COMPUTER VISION AND NLP

Professors:
Daniele Pannone

Students:
Edoardo Allegrini
Samuele Bella

Academic Year 2022/2023

Contents

1 Introduction

According to [1] malware detection is a crucial factor in cybersecurity, not only organizations, but whoever uses a technologic appliance, should strive to detect and validate malware incidents rapidly to minimize the damage. Our goal is to develop a Malware Detection System (MDS) able to provide a significant contribution of defense against malicious attacks. Instead of traditional MDS that uses feature extractions ML algorithms, we thought that could be interesting and challeging to develop a Convolutional Neural Network (CNN) trained to classify an executable file as malware or benign. This approach, as shown in the next pages, solves the time-consuming problem of traditional ML MDS.

2 Development

2.1 Approach

The project includes a training phase in which the CNN learns to detect malware and benign executable by their image representation. The image approach is a focal point of this project, in fact analyzing a potential virus executable is not that easy as could be thought. Converting the exe in image prevents the time and resource consuming taken by the analysis of it's behaviour running in a SandBox, which is one of the phases of a traditional malware detection. Our CNN is so trained to learn the features/textures of the image and associate them to the corresponding labels.

2.2 Executable convertion to image

To convert executable in image we developed a function called "exe_to_png" which takes in input the path of file α to transform. For first instantiates a numpy array of shape [256, 256], then loops through every byte of the file α and converts that byte into a pixel using big-endian approach. At the end of the computation is returned, and saved, always an image 256x256; if the executable doesn't fit the dimensions 256x256 the image is padded with zeros, conversely if the bytes of α exceeds then the exceeding part is discarded. We decided to discard the exceeding bytes because the loss of informations is not comparable to the needed time that needs an eventual CNN trained by very big images of different sizes.

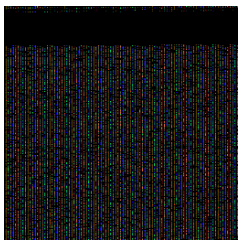


Figure 1: Convertion malware to image

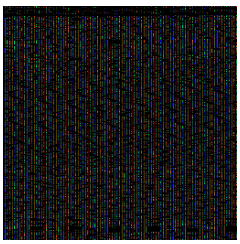


Figure 2: Convertion benign_file to image

2.3 Building CNN

The CNN has two convolutional layers with a kernel of size (5, 5), every convolutional layer is followed by the non-linear layer Rectified Linear Unit (ReLU) combined with a pooling layer. The details about our CNN's structure are:

```
self.cnn = nn.Sequential(
    nn.Conv2d(3, 50, (5, 5)),
    nn.ReLU(),
    nn.MaxPool2d(kernel_size=(2, 2), stride=(2, 2)),
    nn.Conv2d(50, 60, (5, 5)),
    nn.ReLU(),
    nn.MaxPool2d(kernel_size=(2, 2), stride=(2, 2)),
)
```

At the end of CNN layers comes into play a multilayer perceptron (MLP) designed as:

```
self.mlp = nn.Sequential(
    nn.Linear(223260, 64),
    nn.ReLU(),
    nn.Linear(64, 2),
    # nn.LogSoftmax(1)
)
```

The last line of MLP is commented because the performace/accuracy tests reported that the CNN performs better without the last function `LogSoftmax`.

2.4 Training

The batch size of the dataset 3.1 for training the CNN is 10, that phase is splitted into 3 epochs with a learning rate of 0.001. The function `nn.CrossEntropyLoss()` is used to compute the loss and as optimizer we used the stochastic gradient descent.

2.5 Testing

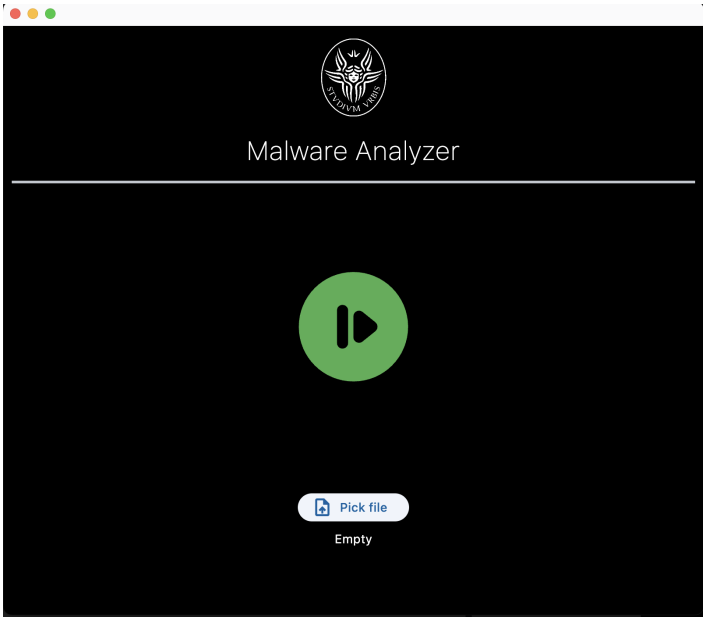
As same as 2.4, in testing phase, the number of epochs is 3 and for each prediction done by the model during testing, if the prediction is correct, the algorithm rises the correct counter for computing the accuracy at the end. Executing the test function on the model pre-trained we get:

- Epoch: 0 Accuracy: 83.69565217391305
- Epoch: 1 Accuracy: 83.69565217391305
- Epoch: 2 Accuracy: 83.69565217391305

2.6 GUI window

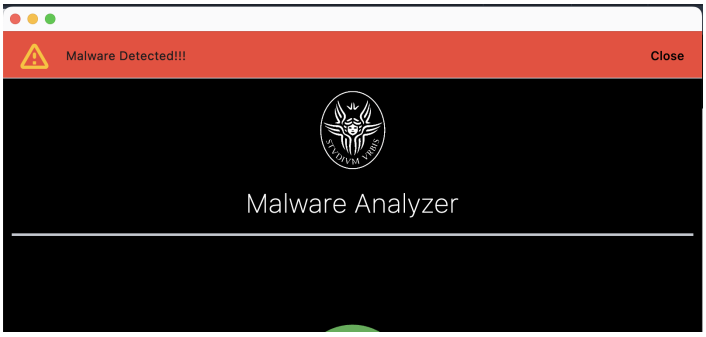
We designed and implemented a GUI for simplifing the interaction between user and CNN model. The GUI lets user upload file and analyze it clicking the run button 3.

Figure 3: GUI



When the detection is finished and the cnn computed if the file selected is a malware or not the result is shown by a popup 4.

Figure 4: GUI detection result



3 Conclusions

3.1 Dataset

We created the dataset used for this project collecting executables of malware and benign files and processing the relative images using 2.2. The dataset is accessible on kaggle.
The dataset is structured as follows:

```
data
├── train This directory holds
│   │    data for training.
│   ├── malware 4481 files.
│   └── benign 491 files.
└── test This directory holds
    │   data for testing.
    ├── malware 4489 files.
    └── benign 491 files.
```

3.2 Results

References

[1] Karen Scarfone (Scarfone Cybersecurity) Murugiah Souppaya (NIST). Guide to malware incident prevention and handling for desktops and laptops. Technical Report NIST Special Publication (SP) 800-83, Rev. 1, National Institute of Standards and Technology, Gaithersburg, MD, 2013.