Introducing Quantum Computing in Mobile Malware Detection

Giovanni Ciaramella
IIT-CNR
Pisa, Italy
g.ciaramella1@studenti.unimol.it

Giacomo Iadarola
IIT-CNR
Pisa, Italy
fabio.martinelli@iit.cnr.it

Fabio Martinelli
IIT-CNR
Pisa, Italy
fabio.martinelli@iit.cnr.it

Francesco Mercaldo
University of Molise & IIT-CNR
Campobasso, Italy
francesco.mercaldo@unimol.it

Antonella Santone
University of Molise
Campobasso, Italy
antonella.santone@unimol.it

Marco Storto
University of Molise
Campobasso, Italy
m.storto@studenti.unimol.it

ABSTRACT

Mobile malware are increasing their complexity to be able to evade the current detection mechanism by gathering our sensitive and private information. For this reason, an active research field is represented by malware detection, with a great effort in the development of deep learning models starting from a set of malicious and legitimate applications. The recent introduction of quantum computing made possible quantum machine learning i.e., the integration of quantum algorithms within machine learning algorithms. In this paper, we propose a comparison between several deep learning models, by taking into account also a hybrid quantum malware detector. We explore the effectiveness of different architectures for malicious family detection in the Android environment: *LeNet*, *AlexNet*, a Convolutional Neural Network model designed by authors, *VGG16* and a Hybrid Quantum Convolutional Neural Network i.e., a model where the first layer is a quantum convolution that uses transformations in circuits to simulate the behavior of a quantum computer. Experiments performed on a real-world dataset composed of 8446 Android malicious and legitimate applications allow us to compare the various models, with particular regard to the quantum model concerning the other ones.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; • **Security and privacy** → **Software and application security**.

KEYWORDS

malware, deep learning, quantum computing, security, Android

ACM Reference Format:
Giovanni Ciaramella, Giacomo Iadarola, Fabio Martinelli, Francesco Mercaldo, Antonella Santone, and Marco Storto. 2022. Introducing Quantum Computing in Mobile Malware Detection. In *The 17th International Conference on Availability, Reliability and Security (ARES 2022), August 23–26, 2022, Vienna, Austria.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3538969.3543816

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
ARES 2022, August 23–26, 2022, Vienna, Austria
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-9670-7/22/08…$15.00
https://doi.org/10.1145/3538969.3543816

1 INTRODUCTION

Mobile malware writers are constantly looking for ways for their malicious payloads to be downloaded by finding new ways to monetize their efforts. This is the reason why the continued use of banking Trojans, billing fraud, and adware does not come as a major surprise [9]. In this context, the novelty is represented by the different methods that attackers are trying to get apps and malware around or through the security screening of the application stores, for instance, Google Play i.e., the Android store. To circumvent security screening, malicious writers try to distribute their malicious applications through SMS messages or links on popular social media sites. Another attack vector is the so-called drive-by-download, consisting of applications with minimal but legitimate functionality, where the malicious payload is delivered during an update when scrutiny is lessened, and thus obtained additional encrypted packages to obfuscate the real malware. They are also learning to abuse features in mobile device operating systems that give them access beyond the intended purpose, such as accessibility services and messaging controls.

According to the McAfee Mobile Threat Report 2021[1] with most of the world still anxious about COVID-19 and getting vaccinated, cybercriminals are targeting these fears with bogus apps, text messages, and social media invitations. Malware and malicious links hidden inside these fakes display ads and try to steal banking information and credentials. Android is the most widely used mobile operating system in the world and this is the reason why it continues to be a primary target for malware attacks due to its market share and open source architecture [2].

Every year, the number of mobile-oriented malware targeting the Android platform is exponentially increasing: as a matter of fact, Kaspersky security analysts detected 886,105 malicious installation packages in Q2 2021[3].

A third of all the mobile threats detected in the second quarter of 2021 is in the riskware malware category (i.e., application not designed to be harmful, but containing functions that can be used for fraudulent purposes): these applications are dramatically increased by 23.04 p.p., while we ha assisted to a decline of adware malware (i.e., applications with aggressive advertisements intentionally exposed to the user, for commercial purposes).

[1] https://www.mcafee.com/content/dam/global/infographics/McAfeeMobileThreatReport2021.pdf
[2] https://www.networksplusco.com/threats-to-mobile-devices-using-the-android-operating-system/
[3] https://securelist.com/it-threat-evolution-q2-2021-mobile-statistics/103636/

Indeed, these applications are the second threat in the second quarter of 2021 after the riskware (34.10%) with 27.33 p.p. down compared to the previous quarter of the same year. Moreover, several types of Trojans complete the top three (16.48%), whose share increased by 8.21 p.p.

These mobile threats continue to spread because free and commercial mobile antimalware are not adequate to detect new malicious payloads whose signatures are not included within the antimalware repository [6]; this is the reason why researchers from both the industrial and academic world are proposing methods for the detection of malicious behaviors by exploiting artificial intelligence, with particular regard to machine learning. Among the various machine learning techniques, the Deep Learning (DL) models are obtaining more promising results in terms of accuracy than shallow machine learning techniques in many classification tasks, and they have attracted strong interest from the research community in the last decade [24]. Considering that one of the tasks in which DL models overcome shallow machine learning, in terms of performance, is image recognition tasks, malware detection research is adopting DL to process (malware and legitimate) applications represented as images.

The limits of what machines can learn have always been defined by the computer hardware we run deep learning algorithms on—for example, the success of modern-day deep learning with neural networks is enabled by parallel GPU clusters.

Recently, quantum computing machine learning extends the pool of hardware for machine learning by an entirely new type of computing device i.e., the quantum computer [17]. Information processing with quantum computers relies on substantially different laws of physics known as quantum theory [13].

In this paper, we propose several Android malware detectors, based on a deep learning architecture and quantum computing. The idea behind this paper (and, thus, the novelty) is to introduce quantum computing in malware detection but also to compare quantum computing with deep learning in terms of accuracy for the malware detection task in the Android environment. As a matter of fact, to the best of the knowledge of authors, this paper represents the first attempt to introduce quantum computing in image-based malware detection.

The remaining of the paper proceeds as follows: in the next section preliminary notions on the malware detection techniques and the quantum and classical machine learning considered are this paper is provided, in Section 3 the models we considered are presented; the results of the experimental analysis are presented in Section 4 and, finally, in last section conclusions and future research plan are drawn.

2 BACKGROUND

This section reports preliminary information about the technique exploited for malware detection in the Android environment (i.e. the image-based malware detection) and background notions about quantum machine learning. For interested readers, we refer to literature for further information [7, 14].

2.1 Image-based Malware Detection

In recent years, a popular method to classify malware consists of converting malicious software into images, and then applying Image-based DL models for performing the classification [11, 21]. This process starts from the executable file, which is transformed into an array of values by grouping the bits in blocks and casting the bytes to unsigned integers. Then, the array of values is scaled into a 2D matrix, and converted to a grayscale (or RGB) image, by casting each value to a pixel [15].

Most of the malware are variants of previous malware samples, with some differences in the codebase to mislead signature-based antimalware: this is the reason why malware variants of the same original malicious sample are grouped into malware families. Assuming that malware of the same family shares part of the code, also their corresponding images will have patterns in common. Similar to the classical classification of objects within images, a DL trained on a large number of input samples will be able to recognize the pattern that characterizes one malware family rather than another.

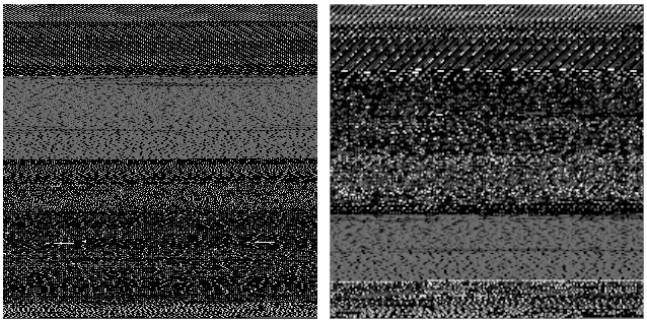

Figure 1: Android malware converted to images, a sample from the *Dowgin* family (left) and *Fusob* family (right).

Figure 1 reports two samples of two Android malware converted to images, belonging to two different families (i.e., *Dowgin* and *Fusob*, respectively). The images may look like random noise, but the information coming from the input executables is preserved.

2.2 Quantum and Classical Machine Learning

The increasing importance of machine learning in recent years has led to many relevant studies that investigate the promise of quantum computers for machine learning [12, 16, 23]. Quantum Machine Learning (QML) is the field that concerns the integration of quantum algorithms into classical machine learning models. The classical machine learning algorithms are used to analyze large amounts of data, and quantum computing comes in help by using qubits and quantum operators to increase the computation and data storage speed of these algorithms. The QML field is still at the front edge of computer science research, and some improvements (the so-called “Quantum Advantage") have not been proved theoretically, yet. With the concept of “Quantum Advantage", the researchers refer to the advantage of Quantum Models over classical approaches, by leveraging quantum effects. While some attainable advantage of the Quantum Models over generic classical computations has been proven, there is no certainty that the “advantage" may bring to a

complete “supremacy" in the future; the question is still an open debate. The strong quantum speedup is still debatable if a lower bound for the classical algorithms has not been found yet. Indeed, even if the quantum advantage was demonstrated for some problems (such as the Shor’s factoring algorithm [19]), the demonstration represents a quantum speedup over the best known classical counterpart, but it may be, theoretically, disrupted by improvements in classical calculations techniques and lower bound verifications. The main difference between a quantum and a classical calculator is the basic unit of calculation. In the classic case, a process is based on the bit, which can assume only two states, generally represented as 0 and 1, corresponding to the state of charge of a transistor. On the other hand, in the quantum scenario, the equivalent of the bit is the quantum bit (or qubit) which follows properties deriving from the postulates of quantum mechanics. Superposition and entanglement are two of the key concepts of quantum theory and contribute to the great computational capacity of quantum computers. The combination of these two principles allows you to create computer systems characterized by a great calculation capacity and speed of execution. While in classical computer science, a system consisting of two bits can store only one of the four possible binary combinations (00, 01, 10, 11), a 2-qubit quantum system can store all four combinations.

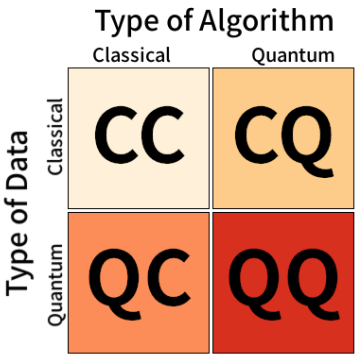

Figure 2: The possible approaches to combine the Machine Learning and the Quantum Computing fields.

In [1] authors conducted research in which they distinguished four ways to combine quantum computing and machine learning. As a matter of fact there are several approaches to applying quantum principles. Those methodologies are reported in Figure 2 and differ on whether the data is generated by a quantum (Q) or classical (C) system and if the information processing device is quantum (Q) or classical (C). Briefly, In summary, case *CC* refers to classical data treated conventionally, case *CQ* employs quantum computing to process classical datasets, and case *QQ* uses quantum data processed by Quantum Computer. In this paper, we focus our attention on quantum data processed by classical algorithms. (i.e., *QC*). The quantum machine learning algorithms can be a quantum version of conventional ML or a completely new algorithm that addresses a classical problem in a quantum scenario. However, also hybrid classical-quantum methods can be used, where parts of the process, usually computationally expensive ones, are assigned to QC. The aim is to combine quantum and classical algorithms to obtain higher performance and decrease the learning cost. One more classification can be done on the type of input data, which can be encoded with classical or quantum representation; thus, also the data type generates more hybrid approaches between the classical and quantum worlds.

3 METHODOLOGY

In this section, we describe the proposed methodology aimed to compare different DL models for the Android malware detection task, with particular regard to the introduction to a quantum machine learning model. As shown in Figure 3 the entire process is composed of four main steps:

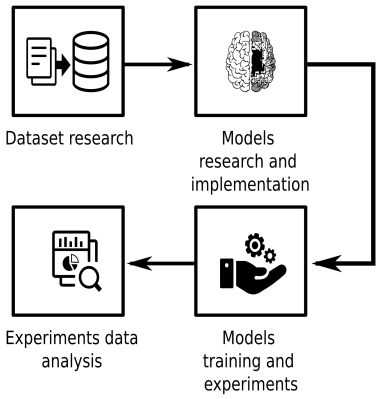

Figure 3: Methodology steps.

3.1 Dataset

The first step necessary for the development of the proposed comparison is to find samples of Android malware and legitimate applications to convert as greyscale images (as introduced in the 2.1 subsection we consider a malware detector based on images). Considering that we apply machine learning, the choice of a solid dataset is essential for the realization of experiments that concern the training of deep learning models, as a matter of fact starting from a poorly constructed dataset could lead to false results and therefore make the experiments useless.

3.2 Model research and implementation

The second phase is focused on the research and implementation of different deep learning models for image classification. In detail, the proposed comparison consists of an evaluation between the efficiency of different models: *LeNet*, *AlexNet*, a Convolutional Neural Network (i.e., called Standard CNN) model designed by authors, *VGG16* and a Hybrid Quantum Convolutional Neural Network (i.e., CNN with a layer that uses transformations in circuits to simulate a quantum convolution). Below we briefly explain the exploited models:

- **Le-Net**: was proposed in the 1990s, but a concrete evaluation was done during the 2010s. The basic configuration of

this model is composed of two convolution layers, two subsampling layers, two fully connected layers, and an output layer with a Gaussian connection;

- **AlexNet**: during 2012, Alex Krizhevesky and colleagues, proposed a more complex model compared to Le-Net, which was called AlexNet. This model was a significant breakthrough in the field of machine learning and computer vision for visual recognition and classification tasks and it is the point in history where interest in DL rapidly increased [3]. AlexNet is composed of three convolution layers and two fully connected layers;
- **Standard CNN**: Since 2012, the Convolutional Neural Networks (CNN) have conquered most computer vision fields and are growing at a rapid pace. There are differences in their architecture, but all the CNN models are based on the principle of the convolution filter. This filter, called *kernel*, is applied to the pixel matrix that composes the image on the three RGB levels. We report the structure of our CNN (that we called "Standard" to distinguish between the other CNN architectures) in Listing 1. Its structure was ideated to have a lighter model than the VGG16 but is still capable of achieving high accuracies in multi-class classification tasks. The main layers, common to many more CNNs, are the *Convolutional, Maxpooling, Dropout,* and *Dense* layers;
- **Visual Geometry Group 16**: also known as VGG-16 [20], was introduced in 2014 to address object recognition tasks on the ImageNet, a large dataset with 1000 output classes. The main contribution of VGG16 is that it shows that the depth of a network is a critical component to achieve better recognition or classification accuracy in CNNs. The VGG16 architecture is composed of a series of convolution layers with a 3x3 filter and Maxpool layers with a 2x2 filter. The 16 n VGG16 refers to the 16 layers that have weights in its architecture. The final part of the model is composed of 2 fully connected (Dense) layers and a softmax to perform the classification. One common training approach consists of keeping the convolutional part of the model with the weights coming from training the model on the ImageNet, while the final Dense layers part is trained for the specific classification task required;
- **Hybrid Quantum Convolutional Neural Network** i.e., Hybrid-QCNN: this model is similar to the Standard CNN. Hybrid-QCNNs are meta-networks comprised of quantum and classical neural network-based function blocks composed with one another in the topology of a directed graph. We consider this a rendition of a hybrid quantum-classical computational graph where the inner workings (variables, component functions) of various functions are abstracted into boxes. The edges represent the flow of classical information through the metanetwork of quantum and classical functions[5]. As a matter of fact, it adds to the traditional convolutional network a first layer that simulates calculations performed by a quantum computer, by performing a quantum convolution. This model has as its first layer a quantum convolution that uses transformations in circuits to simulate the behaviour of a quantum computer. In Listing 2, we implemented a model where the first layer is a convolutional layer built to work on quantum circuits.

Listing 1: Structure of a Standard Convolutional Neural Network.

```
model = models.Sequential()
model.add(layers.Conv2D(30, (3, 3), \
          activation='relu',
          input_shape=(self.input_width_height
              ,
                       self.input_width_height
                           ,
                       self.channels)))
model.add(layers.MaxPooling2D(pool_size=(2, 2)
    ))
model.add(layers.Conv2D(15, (3, 3), \
          activation='relu'))
model.add(layers.MaxPooling2D(pool_size=(2, 2)
    ))
model.add(layers.Dropout(0.25))
model.add(layers.Flatten())
model.add(layers.Dense(128, activation='relu')
    )
model.add(layers.Dropout(0.5))
model.add(layers.Dense(50, activation='relu'))
model.add(layers.Dense(self.num_classes, \
          activation='softmax'))
```

Listing 2: Structure of a Hybrid Convolutional Neural Network.

```
NEW_SIZE = 10
[...]
model = models.Sequential()
model.add(QConv(filter_size=2, depth=8, \
  activation='relu', name='qconv1', \
  input_shape=(NEW_SIZE, NEW_SIZE, self.
      channels)))
model.add(layers.Conv2D(16, (2, 2), \
  activation='relu'))
model.add(layers.Flatten())
model.add(layers.Dense(32, activation='relu'))
model.add(layers.Dense(self.num_classes, \
  activation='softmax'))
```

We perform a model assessment phase to find the best hyperparameters for the DL models, with regard to the classification task and the comparison. We look for a fair balance in the models' input in order to have a solid base to compare the model to similar experiment results. For instance, we reduce the image input size for the CNN models to adopt an image size closer to the Hybrid QCNN one,

which was 25x25x1 due to the model complexity restriction and computational resources required. Finally, the experimental results are evaluated and compared one model to each other, in order to provide interesting suggestions for future works and test how a common image classification task (such as the image-based malware classification) performs on new architectures and scientific principles (in a simulated environment).

3.3 Experiments

To implement our experimental analysis, we used Python as a programming language and we considered two different open-source libraries, *Tensorflow* and *Tensorflow Quantum*, to develop the DL models. More details on *Tensorflow Quantum* can be found in [5]. Briefly, Tensorflow Quantum is a framework that offers high-level abstractions for the design and training of both discriminative and generative quantum, compatible with existing TensorFlow APIs, along with quantum circuit simulators. In addition, the authors demonstrate that this library can apply to tackle advanced in many fields, i.e. quantum learning tasks including meta-learning, layerwise learning, Hamiltonian learning, sampling thermal states, variational quantum eigensolvers, classification of quantum phase transitions, generative adversarial networks, and reinforcement learning.

4 EXPERIMENTS AND DISCUSSION

In this section, we describe the results of the experimental analysis we conducted.

We gathered a dataset of malware and benign Android applications (8446). To compose the dataset, we used six different malware families: *Airpush* (1170), *Dowgin* (763), *FakeInst* (2129), *Fusob* (1275), *Jisut* (550), and *Mecor* (1499). In addition, we also reserved a class for the *Trusted* (1060) family, which contains legitimate Android applications. The samples of the *Trusted* applications were tested with the commercial antimalware (for instance the ones provided by the VirusTotal service), to avoid possible mistakes in the dataset labeling. Also, the malware applications were checked with the VirusTotal service in order to confirm the maliciousness and the malicious family. The dataset was divided into training and test set, with a percentage of 80-20. Also has been balanced the number of family samples in each set. The executable code of the input applications was converted to an image in PNG format, thus, the preprocessing step generates the image from all the 8446 samples.

We executed several experiments and tested different hyperparameters. Table 1 reports some hyperparameters settings (input image size, epochs number, batch dimension) with regard to the DL model, while Table 2 exhibits the experimental analysis results.

As shown from settings in Table 1, it emerges that the smallest image dimension is the one we considered for the Hybrid-QCNN model: this is due to computational resources and thus the time required to run the Hybrid-QCNN model, greater if compared to the time for the other models. The training time reported in Table 1 should not be considered for a proper time complexity analysis, but only for comparison purposes between the proposed models. To evaluate the models, several metrics are considered: *Loss*, *Accuracy*, *Precision*, *Recall*, *F-Measure* and *Area Under the Curve* (AUC). Below we provide the metrics definition.

Table 1: Model Settings

Model	Image	Batch	Epoch	Training time (HH:MM:SS)
LeNet	100x1	32	50	00:02:47
AlexNet	100x1	32	50	00:03:59
Standard CNN	100x1	32	25	00:02:49
VGG16	100x3	32	25	00:05:56
Hybrid-QCNN	25x1	32	20	07:43:21

- **Loss**, is the penalty for a bad prediction and in other words, it indicates how bad the model's prediction was on a single sample.
- **Accuracy** is used to determine which model is best at identifying relationships and patterns between variables in a dataset based on the input, or training, data.

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

- **Precision** indicates the quality of a positive prediction made by the model. It can be computed using the following formula:

$$Precision = \frac{TP}{TP + FP}$$

- **Recall** represents the fraction of relevant instances that were retrieved, and can be computed as follows:

$$Recall = \frac{TP}{TP + FN}$$

- **F-Measure** is computed as the harmonic mean of precision and recall, giving each the same weighting. It allows a model to be evaluated taking both the precision and recall into account using a single score, which is helpful when describing the performance of the model and in comparing models.

$$F\text{-}measure = 2 * \frac{Precision * Recall}{Precision + Recall}$$

- **Area Under the Curve** also known as AUC, measures the entire two-dimensional area underneath the entire ROC curve (think integral calculus) from (0,0) to (1,1). In other words, it helps us visualize how well our machine learning classifier is performing.

In the formulae previously shown, we used the terms: *true positive, true negative, false positive,* and *false negative.* True positive is an outcome where the model correctly predicts the positive class. Similarly, a true negative is an outcome where the model correctly predicts the negative class. About false positive is an outcome where the model incorrectly predicts the positive class, while a false negative is an outcome where the model incorrectly predicts

the negative class. In the case of multi-class classification, where there is no binary choice between positive/negative class, the analysis proceeds by targeting one class at a time: the true positive is the correct class (the cell cross between the predicted and true label in the confusion matrix representation), the false positive is the sum of the target class samples misclassified to another class (the Y-axis sum), the false negative is the sum of the target class samples of other classes classified in the target class (the X-axis sum), and the true negative is the sum of all the other remaining cells. Table 3 shows the experimental results on the test set with regards to the output classes (the malware families and the Trusted category).

The results in Table 4 shows that the model obtaining the best performances is the VGG16, with a precision equal to 0.96 and a recall equal to 0.95.

The Hybrid-QCNN model exhibits a precision equal to 0.92 and a recall of 0.91: considering that we trained this model with an image of 25x1 (while the VGG16 is trained with an image of 100x3 dimension) the quantum model obtain really interesting performances.

It is worth noting also the good performance of the Standard CNN model, able to obtain a precision of 0.95 and a recall of 0.84. These results are almost comparable with the Hybrid-QCNN and also promising, considering the lighter architecture than the VGG16 (21 layers of the VGG16 and 12 layers for the Standard CNN).

On the other hand, the LeNet and the AlexNet were unable to correctly distinguish the Android application from the different malicious families: they both obtained a precision and a recall equal to 0. Their structure was unable to extract sufficient features and information from the input samples to train the model correctly, probably, due to the low number of trainable layers and parameters.

In Figures 4 and 5 we respectively show the confusion matrix of VGG16 and Hybrid-QCNN models to directly compare the best deep learning model (in terms of performances) with the quantum one, in order to directly compare the best deep learning architecture and the quantum model.

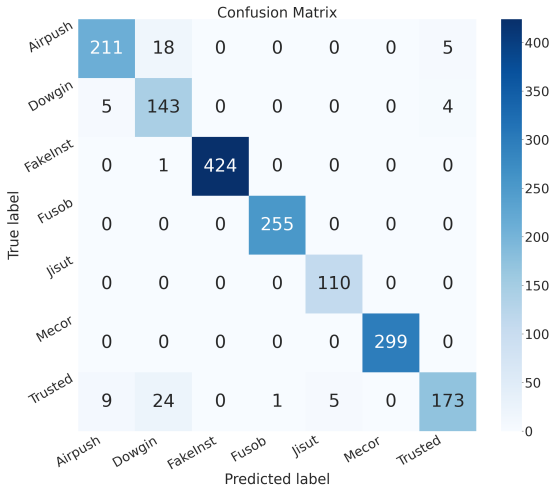

Figure 4: Confusion Matrix obtained with the VGG16 model.

As shown from the *VGG16* confusion matrix in Figure 4, this model produces a really small number of misclassified samples. As a matter of fact, the greater number of incorrectly classified samples is 24 (samples belonging to the Dogwin family marked as Trusted by the VGG16 model). Other 9 malware belonging to the Airpush family are marked as trusted.

The confusion matrix of the Hybrid-QCNN model is shown in Figure 5.

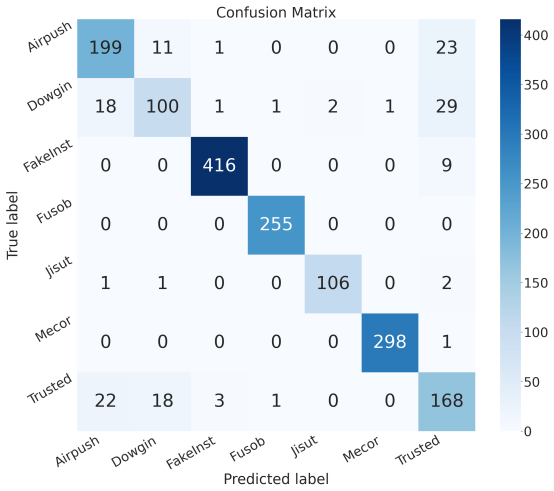

Figure 5: Confusion Matrix obtained with the Hybrid-QCNN model.

The number of misclassified samples of the Hybrid-QCNN model is slightly higher is compared to the VGG16 model. Similar to what happened for the VGG16 model, also for the Hybrid-QCNN one, the greater number of misclassified samples are belonging to the Airpush and Dowgin malware families and are marked from the Hybrid-QCNN model as Trusted. This is symptomatic that a reduced set of images obtained from the Airpush and Dowgin malware are similar to images obtained from Trusted applications. Relating to the Hybrid-QCNN model also a set of Trusted applications is misclassified into the Airpush and Dowgwin families, as shown from the confusion matrix in Figure **??**. Furthermore, 18 applications belonging to the Airpush family are classified as the Dowgin one. This is the reason the accuracy of the Hybrid-QCNN model is (slightly) lower if compared to the VGG16 (i.e., 0.95 for the VGG16 model and 0.91 for the Hybrid-QCNN one). We remark that the Hybrid-QCNN model is trained with 25x25x1 images while the VGG16 one with 100x100x3 images and that the Hybrid-QCNN model training required approximately 8 hours while the VGG16 less than 6 minutes, as highlighted from Table 1.

5 RELATED WORK

The current state-of-the-art in the adoption of deep learning for mobile malware detection is discussed in this section.

Quantum computing has evolved rapidly in a short time and is showing significant benefits in different fields. In medicine, for

Table 2: Experimental Results on the test set.

Model	Loss	Accuracy	Precision	Recall	F-Measure	AUC
LeNet	1.86	0.25	0	0	0	0.62
AlexNet	1.87	0.25	0	0	0	0.62
Standard CNN	0.28	0.90	0.95	0.84	0.89	0.99
VGG16	0.15	0.95	0.96	0.95	0.95	0.99
Hybrid-QCNN	0.81	0.91	0.92	0.91	0.91	0.97

Table 3: Experimental Results for each output class on the test set, with regard of the two best DL models.

Output Classes	DL Model	Accuracies	Precision	Recall	F-Measure	AUC
Airpush	*Hybrid-QCNN*	0.95	0.83	0.85	0.84	0.91
	VGG16	0.98	0.94	0.9	0.92	0.95
Dowgin	*Hybrid-QCNN*	0.95	0.77	0.66	0.71	0.82
	VGG16	0.97	0.77	0.94	0.85	0.96
FakeInst	*Hybrid-QCNN*	0.99	0.99	0.98	0.98	0.99
	VGG16	1.0	1.0	1.0	1.0	1.0
Fusob	*Hybrid-QCNN*	1.0	0.99	1.0	1.0	1.0
	VGG16	1.0	1.0	1.0	1.0	1.0
Jisut	*Hybrid-QCNN*	1.0	0.98	0.96	0.97	0.98
	VGG16	1.0	0.95	1.0	0.97	1.0
Mecor	*Hybrid-QCNN*	1.0	1.0	1.0	1.0	1.0
	VGG16	1.0	1.0	1.0	1.0	1.0
Trusted	*Hybrid-QCNN*	0.94	0.72	0.79	0.76	0.87
	VGG16	0.97	0.95	0.81	0.88	0.90

example, following the global pandemic in 2020, deep learning is used frequently to assist in analyzing potentially large numbers of thoracic CT exams. To gain better performances to the analysis cited before, in [4] authors used *QML*. The results obtained from the researchers have demonstrated that *quantum algorithms* have better performances to spot the infection in its early stages.

In [16] authors, showed that the Support Vector Machine (SVM) can be implemented into a quantum computer, with complexity logarithmic in the size of the vectors and the number of training examples. In cases where classical sampling algorithms require polynomial time, an exponential speedup obtaining. At the core of this quantum, the big data algorithm is a non-sparse matrix exponentiation technique for efficiently performing a matrix inversion of the training data inner-product (kernel) matrix. Different from them, we exploit a Hybrid QCNN model with a good level of accuracy, i.e. 0.91.

Researchers in [23], showed that quantum computing not only reduces the time required to train a deep restricted Boltzmann machine but also provides a richer and more comprehensive framework for deep learning than classical computing. This leads to significant improvements in the optimization of the underlying objective function. It came out under the assumption algorithms for quantum computers have been shown to efficiently solve some problems that are intractable on conventional, classical computers. In addition, the quantum method shown is also able to allow an efficient training of full Boltzmann machines and multilayer, fully connected models, for which there are no efficient classical counterparts.

Seymour and his colleague [18] show that a minimalist malware classifier, with cross-validation accuracy comparable to standard machine learning algorithms, can be created. However, even such a minimalist classifier incurs a surprising level of overhead.

Authors in [12] consider supervised and unsupervised quantum machine learning algorithms for cluster assignment: they show that quantum machine learning algorithms can take logarithmic time in both the number of vectors and their dimension, an exponential speed-up over classical algorithms.

Researchers in [2] propose a combination of software used to locate the most frequent hashes and n-grams between benign and malicious software and a quantum search algorithm. The latter will be used to search among every permutation of the entangled key and value pairs to find the desired hash value. This prevents one from having to re-compute hashes for a set of n-grams.

6 CONCLUSION AND FUTURE WORKS

Considering the inadequacy of current antimalware to detect new threats, the industrial and scientific research communities are proposing new methods for malware detection, with particular regard to the Android platform and by exploiting deep learning techniques. Recently, due to the introduction of quantum computing, it is possible to train models by exploiting the integration of quantum algorithms within machine learning algorithms. In this paper, we proposed a comparison between several deep learning models and a hybrid quantum model for the Android malware detection task. We represent each application as a grayscale image: in detail, in the experimental analysis, we consider 7386 applications belonging

to six different malicious families and 1060 legitimate applications composing one class reserved for the Trusted application.

We obtain that the best architecture for the Android malware detection task is the VGG16 model, with an accuracy equal to 0.95. The second model, from the accuracy point of view, is the Hybrid-QCNN one with an accuracy equal to 0.91. Due to computational issues, we remark that the Hybrid-QCNN was trained with images of 25x25x1 size, while the VGG16 model was trained with an image of 100x100x3 dimension: this result is symptomatic that the Hybrid-QCNN is promising for the malware detection task.

In future works we will consider a fully quantum model, as a matter of fact, the Hybrid-QCNN model exploited in this paper basically adds to the convolutional network a first layer aimed to simulate the computations performed by a quantum computer.

Moreover, we plan to apply the Hybrid-QCNN model to images directly obtained from the source code (with a reverse engineering process) [8] in order to consider the possibility to exploit activation maps mechanisms[10, 22] with the aim to highlight the area in the image responsible for a certain prediction.

ACKNOWLEDGMENTS

This work has been partially supported by MIUR SecureOpenNets, EU SPARTA, CyberSANE and E-CORRIDOR projects.

REFERENCES

[1] Esma Aïmeur, Gilles Brassard, and Sébastien Gambs. 2006. Machine learning in a quantum world. In *Conference of the Canadian Society for Computational Studies of Intelligence*. Springer, 431–442.

[2] Nichola R Allgood. 2020. *A quantum algorithm to locate unknown hashes for known n-grams within a large malware corpus.* Ph. D. Dissertation. University of Maryland, Baltimore County.

[3] Md Zahangir Alom, Tarek M Taha, Christopher Yakopcic, Stefan Westberg, Paheding Sidike, Mst Shamima Nasrin, Brian C Van Esesn, Abdul A S Awwal, and Vijayan K Asari. 2018. The history began from alexnet: A comprehensive survey on deep learning approaches. *arXiv preprint arXiv:1803.01164* (2018).

[4] Javaria Amin, Muhammad Sharif, Nadia Gul, Seifedine Kadry, and Chinmay Chakraborty. 2021. Quantum machine learning architecture for COVID-19 classification based on synthetic data generation using conditional adversarial neural network. *Cognitive Computation* (2021), 1–12.

[5] Michael Broughton, Guillaume Verdon, Trevor McCourt, Antonio J Martinez, Jae Hyeon Yoo, Sergei V Isakov, Philip Massey, Ramin Halavati, Murphy Yuezhen Niu, Alexander Zlokapa, et al. 2020. Tensorflow quantum: A software framework for quantum machine learning. *arXiv preprint arXiv:2003.02989* (2020).

[6] Alberto Ferrante, Eric Medvet, Francesco Mercaldo, Jelena Milosevic, and Corrado Aaron Visaggio. 2016. Spotting the malicious moment: Characterizing malware behavior using dynamic features. In *2016 11th International Conference on Availability, Reliability and Security (ARES)*. IEEE, 372–381.

[7] Ekta Gandotra, Divya Bansal, and Sanjeev Sofat. 2014. Malware analysis and classification: A survey. *Journal of Information Security* 2014 (2014).

[8] Giacomo Iadarola, Rosangela Casolare, Fabio Martinelli, Francesco Mercaldo, Christian Peluso, and Antonella Santone. 2021. A Semi-Automated Explainability-Driven Approach for Malware Analysis through Deep Learning. In *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–8.

[9] Giacomo Iadarola, Fabio Martinelli, Francesco Mercaldo, and Antonella Santone. 2019. Formal methods for android banking malware analysis and detection. In *2019 Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS)*. IEEE, 331–336.

[10] Giacomo Iadarola, Fabio Martinelli, Francesco Mercaldo, and Antonella Santone. 2020. Evaluating deep learning classification reliability in android malware family detection. In *2020 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*. IEEE, 255–260.

[11] Giacomo Iadarola, Fabio Martinelli, Francesco Mercaldo, and Antonella Santone. 2021. Towards an interpretable deep learning model for mobile malware detection and family identification. *Computers & Security* 105 (2021), 102198.

[12] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. 2013. Quantum algorithms for supervised and unsupervised machine learning. *arXiv preprint arXiv:1307.0411* (2013).

[13] José D Martín-Guerrero and Lucas Lamata. 2022. Quantum Machine Learning: A tutorial. *Neurocomputing* 470 (2022), 457–461.

[14] Fabio Valerio Massoli, Lucia Vadicamo, Giuseppe Amato, and Fabrizio Falchi. 2021. A Leap among Entanglement and Neural Networks: A Quantum Survey. *arXiv preprint arXiv:2107.03313* (2021).

[15] Francesco Mercaldo and Antonella Santone. 2020. Deep learning for image-based mobile malware detection. *Journal of Computer Virology and Hacking Techniques* 16, 2 (2020), 157–171.

[16] Patrick Rebentrost, Masoud Mohseni, and Seth Lloyd. 2014. Quantum support vector machine for big data classification. *Physical review letters* 113, 13 (2014), 130503.

[17] Maria Schuld, Ilya Sinayskiy, and Francesco Petruccione. 2015. An introduction to quantum machine learning. *Contemporary Physics* 56, 2 (2015), 172–185.

[18] John J Seymour. 2014. *Quantum classification of malware.* University of Maryland, Baltimore County.

[19] Peter W Shor. 1999. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review* 41, 2 (1999), 303–332.

[20] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[21] Danish Vasan, Mamoun Alazab, Sobia Wassan, Babak Safaei, and Qin Zheng. 2020. Image-Based malware classification using ensemble of CNN architectures (IMCEC). *Computers & Security* 92 (2020), 101748.

[22] Haofan Wang, Zifan Wang, Mengnan Du, Fan Yang, Zijian Zhang, Sirui Ding, Piotr Mardziel, and Xia Hu. 2020. Score-CAM: Score-weighted visual explanations for convolutional neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*. 24–25.

[23] Nathan Wiebe, Ashish Kapoor, and Krysta M Svore. 2014. Quantum deep learning. *arXiv preprint arXiv:1412.3489* (2014).

[24] Ding Yuxin and Zhu Siyi. 2019. Malware detection based on deep learning algorithm. *Neural Computing and Applications* 31, 2 (2019), 461–472.