Nanotechnology Perceptions
ISSN 1660-6795
www.nano-ntp.com

Image Based Malware Detection Using Transfer Learning and Hybrid Quantum Machine Learning Models

Soja Rani S[1], Mouleeswaran[2]

[1]Research Scholar, Dayananda Sagar University, Faculty, New Horizon College of Engineering, Bangalore, India
[2]Associate Professor, Dayananda Sagar University-SOE, Bangalore, India
Email: soja.naveen@gmail.com

Image based malware classification is getting more popular in the field of malware detection as it frees the researcher from the tedious process disassembling and analyzing the code of the malicious software. Deep learning and transfer learning methods are efficient for extracting features from images. The accuracy of prediction will be better with the quality of the features extracted. This paper carries out experiments on a benchmark dataset, Malimg using the deep learning model CNN and transfer learning models VGG16 and ResNet. Within the broader paradigm of Quantum Computing, one of the primary research areas is Quantum Machine Learning (QML). Since applying QML algorithms to solve real-world problems might potentially save time and money when compared to more traditional (or digital) machine learning methods, researchers have become quite interested in QML in recent years. We also developed a HQCCNN which is a classical and quantum hybrid version of convolutional neural network for classifying the images that integrates classical and quantum counterparts. A parameterized quantum circuit is utilized in the design of the quantum convolutional layer. To retrieve hidden information from the quantum state, linear unitary transformation is applied. Additionally, pooling operations are carried out via the quantum pooling unit. Once the quantum system evolves, the measurement of quantum state is performed, and the resulting data is inputted into a fully connected traditional layer for additional processing.

Keywords: Cyber Security, Deep Learning, Malware Detection, Transfer Learning.

1. Introduction

Researchers must be ahead of hackers by creating cutting-edge detection tools to find and counter new malware variants as the quantity and complexity of malware threats keep growing. Zero-day attacks take advantage of previously undiscovered flaws, which makes them particularly difficult to identify. The demand for proactive detection techniques that can locate and combat zero-day malware before it causes substantial damage is growing. To get beyond conventional detection methods, malware developers use a variety of evasion

strategies. Obfuscation, encryption, and polymorphism are only a few of the approaches used, thus it's important to comprehend these evasion tactics and create countermeasures to identify and reduce evasive malware.

Deep learning and machine learning methods have shown promise in enhancing malware detection. However, there is a need for ongoing research to develop hybrid approaches that integrate several detection methods to increase accuracy and efficiency, investigate new algorithms, optimize feature selection to refine and enhance these techniques. Malicious executables modify themselves during adversarial assaults to trick or evade detection software. Researchers must investigate adversarial attack methods and create strong detection models that can fend off such assaults. Collaboration and knowledge exchange between researchers, industry professionals, and security practitioners is necessary for advancements in malware detection.

Traditionally used malware detection techniques use malware executable's signatures. Because there are so many malware samples, manual analysis and signature updates are impractical. Algorithms for machine learning and deep learning can automatically extract pertinent properties from raw data, including file attributes, patterns of network traffic, and system behaviors. Without relying on explicit, human-defined rules, this automated feature extraction enables more effective and efficient detection. These models can extrapolate learnt patterns and behaviors to find previously unknown or zero-day malware. They can understand complicated patterns and recognize new and evolving malware variants that they may not have previously encountered by training on big datasets including a variety of malware samples. To avoid detection, malware programmers frequently use strategies like polymorphism and obfuscation. Strong representations of malware traits that are resistant to such methods can be learnt by deep learning systems. Scalable and automatic malware detection capabilities are made possible by machine learning and deep learning algorithms, which can scan massive datasets and analyze fresh samples in real-time. These models can swiftly categorize and identify malware after being trained, speeding up and improving the effectiveness of detection systems.

Commercial antimalware is falling short of detecting zero-day malicious payloads is the reason why academic and commercial researchers are finding innovative ways for detecting malicious behaviors using AI [1]. The research community has been very interested in Deep Learning (DL) models over the past ten years since they are among the machine learning techniques that are showing the most promise in terms of accuracy in many classification tasks when compared to shallow machine learning techniques [2]. Given that image recognition tasks are one of the jobs where deep learning models outperform shallow machine learning models, malware detection research is utilizing deep learning to process malware binaries that are represented as images. The hardware requirements to perform deep learning algorithms are a limitation; for instance, parallel GPU nodes are essential to the efficient learning of deep neural networks. Recent years have seen the development of machine learning leveraging the hardware support given by quantum computing, specifically to perform classification tasks by using the concepts of quantum theory and quantum random access memory [3].

In this article, we provide malware detection techniques based on quantum computing and deep learning architecture. This paper's goal is to use quantum principles for malware detection

from images and to assess the accuracy of quantum computing versus deep learning for malware detection tasks in the Windows environment. The suggested framework has two steps. In the first phase, malicious and good Windows executables are converted into greyscale images to create an image-based PE dataset. These images are used to create deep learning models, and modern convolutional neural networks and quantum architectures are compared. The standard CNN, ResNet, and VGG16 are the deep learning models utilized for comparison. The Hybrid Quantum Classical Convolutional Neural Network, HQCCNN is the quantum model used to build the model.

2. Related Work

To categorize and identify diverse malware families, numerous works have translated malware binaries into pictures and employed a variety of machine learning models. The most recent methods for malware detection that rely on learning- and visualization-based techniques are discussed in this section.

The study [4] suggested employing ResNeXt, a CNN variation, as a method for malware identification. The methodology was tested on the Malimg dataset, converting raw Malimg attributes into grayscale images. On the Malimg and modified Malimg datasets, ResNeXt attained a substantial accuracy of 98.32 percent and 98.86 percent, respectively. Another research [5] suggested a method for malware classification utilizing colored malware graphics along with the same guidelines. The development and optimization of a CNN model is done with colored malware pictures to improve performance. For the binary classification, trials are run on the Malimg malware dataset. In a similar vein, the study [6] developed an ensemble method for malware classification that makes use of many neural networks and a support vector machine (SVM). For their tests on the Malimg dataset, SVM evolved CNN, GRU, and MLP were used to model the data.

A method for identifying malware utilizing a blend of deep learning and machine learning models was suggested by the researchers [7]. The features extracted from CNN were used by XG-Boost model for learning and prediction. On the Maligm dataset, the suggested method CNN-XGBoost displays an accuracy score of 98.7%. Likewise, [8] presents a hybrid model for malware detection that integrates various deep learning methods. To classify the various malware types, two deep transfer learning models, AlexNet and ResNet-152, are used to build the feature set. The Malimg dataset had the best accuracy of 97.78 percent when the suggested method was evaluated on the widely accessible datasets, Microsoft BIG 2015 and Malimg.

[9] provides a hybrid quantum malware detector for mobile malware detection and suggests a comparison between various deep learning models. The study investigates how well these models work in identifying harmful families in the Android ecosystem. LeNet, AlexNet, a convolutional Neural Network (CNN), VGG16, and a Hybrid Quantum CNN—which simulates quantum computer behavior by using quantum convolution—are among the models put to the test. A comparison of the various models may be made using experiments done on a real-world dataset of 8446 Android applications, with an emphasis on how well the quantum model performs in comparison to the others.

The Maling dataset was utilized to create a CNN, which was trained entirely from scratch and then fine-tuned using transfer learning. As a result, the accuracy is 98.61 percent, and the F-score is 0.96. Then, utilizing it for classification on the Microsoft malware classification dataset [10], [11] used hierarchical CNN to build n-gram feature vector of the binaries and utilizing CNN to classify malware with greater accuracy than all previous research.

Research [12] on malware identification in Android apps used a custom Android APK dataset, which was converted to pictures, and were used to train supervised learning models like RF, J48, RT, LMT, and REP Tree. Without employing any benchmark datasets or data augmentation approaches, an accuracy of more than 90% is achieved. Instead of identifying distinct malware classifications, their suggested method determined whether the samples were harmful or not.

BACKGROUND

This section provides background information on deep learning models and quantum machine learning, as well as preliminary data on the method used for detecting malware in the Windows platform.

Image Based Malware Detection

Transforming malicious software to images and using DL models to classify them has been a recent technique for categorizing malware [6, 13]. The technique does not call for the reverse engineering analysis and execution of the malicious code. Additionally, the image characteristics utilized for categorization offer stronger features for encryption and other obfuscation methods.

The bytes that make up a malware binary are treated as the bytes of a greyscale PNG image. The image's length can be varied with the file size, but its width is fixed as 256. Table. 1 lists several suggestions for the image dimensions. A program is developed for the conversion with the following steps: Each byte of the binary file is transformed to a number (0–255), which defines a pixel's color. The final PNG picture also includes a grayscale pixel for each byte. The process is described in Figure 1. We compute the grey-scale histogram of pixel intensity values from the image we got from the executable file. In binary pictures, the pixel value typically corresponds to a 1-bit number that signifies the background or the foreground. A greyscale image's pixel value is a reflects the brightness of the pixel. The byte data in the image keeps this number as an integer of 8 bits ranging from 0 to 255. Normally, 255 is white while zero is considered black. The numerous colours of grey are made up of values in between. Autorun.K, Autorun, Adialer and Alueron, are some of the varieties of Microsoft Windows malware that are seen in Figure 2.

Table1. Malware image width based on malware file sizes

File size of the executable	Width of the visualized Image
>1000 KB	1024
500 – 1000 KB	768
200 – 500 KB	512
100 – 200 KB	384
60 – 100 KB	256
10 – 30 KB	64
<10 KB	32

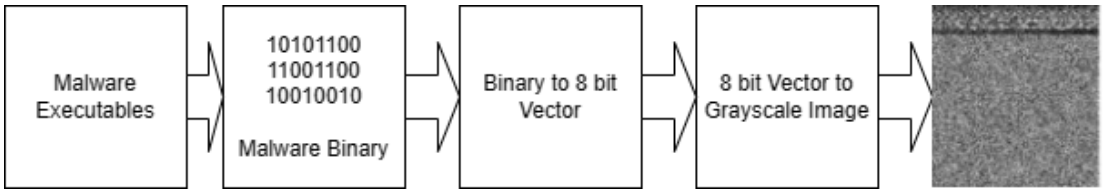

Figure 1: Feature Generation

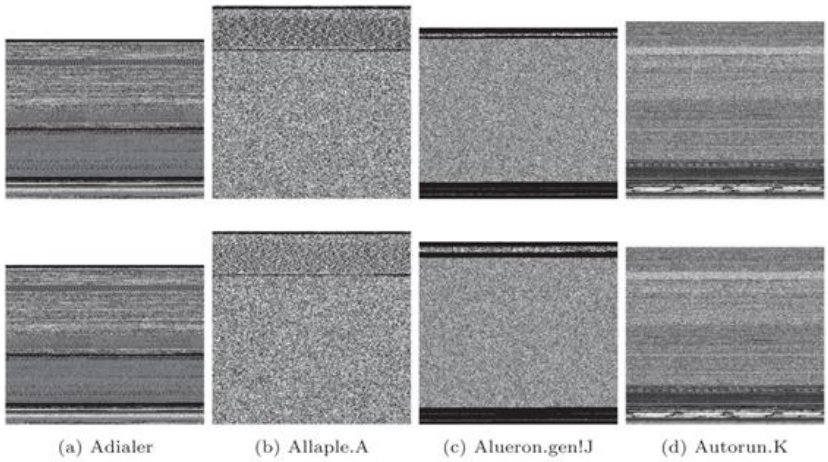

Figure 2: Samples of Windows Malware Families

Deep Learning and Transfer Learning Models

Deep learning and transfer learning are both machine learning paradigms, but they differ in their approaches and applications. Deep learning uses many layers of neurons capable of learning hierarchical representations of data. A model created for one classification problem is used as the starting step for another classification task in transfer learning. DL models are typically trained from scratch on a specific task where the model extracts features and patterns directly from the input data through the training process. In transfer learning pre-trained models trained on one task and that knowledge is either used as a feature extractor or fine-tuned on the new task. Large amounts of labelled data are needed for effectively training a DL model whereas transfer learning can be used even when labelled data for the target task is limited.

CNN

Three types of layers that make up a CNN are - fully connected, pooling, and convolutional. These layers each have unique properties that can be tuned and can carry out various functions on the input data.

1. Convolutional Layer in which finding conjunctions of the features of the lower layer and mapping such appearances to feature maps is the objective. Size n ∗ m feature maps are employed. This map preserves the feature's location and how good it matches with the filter. x filters are involved in each single layer. The depth of the output feature maps is the same as the filters used in one stage. Every filter recognizes a distinct characteristic at each point on

the input. Layer t's output, $Y^{(t)}_i$, consists of $x^{(t)}$ feature maps, each of which is $n^{(t)} * m^{(t)}$ in size. The $Y^{(t)}_i$, or ith feature map, is calculated as follows:

$(Y_i)^{(t)} = B_i{}^{(t)} + \sum_{j=1}^{x(t-1)} K_{i,j}^{(t)} * Y_j^{(t-1)}$ where $K_{i,j}^{(t)}$ is the (t-1) th layer's j^{th} feature map with the i^{th} feature map in the same layer and $B^{(t)}_i$ is a bias matrix.

2. Pooling Layer is in charge of lowering the dimension of the feature vector so as to speed up the process. It is typically applied after many levels of other layers to gradually lower the computational load on the network and reduce the risk of overfitting. The filter's spatial extent, $F^{(t)}$ and the stride $S^{(t)}$ are the two hyperparameters for the pooling layer t. It gives output volume of size $x^{(t)} * n^{(t)} * m^{(t)}$ from an input volume of size $x^{(t-1)} * n^{(t-1)} * m^{(t-1)}$ where,

$$x^{(t)} = x^{(t-1)}\ ;\ n^{(t)} = \frac{n^{(t-1)} - F(t)}{S^t} + 1\ ;\ m^{(t)} = \frac{m^{(t-1)} - F(t)}{S^t} + 1$$

3. Fully Connected dense Layer in which all nodes in one layer are linked to each node in the below layer. Convolutional networks use fully linked layers to achieve the goal of mapping the $x^{(t-1)} * n^{(t-1)} * m^{(t-1)}$ activation functions from the combination of previous several layers. Consequently, the output layer of the neural network will contain x(t−i) outputs, where i is the multilayer perceptron's number of layers. If layer t − 1 is entirely connected,

$$y_i^{(t)} = f\left(z_i^{(t)}\right)\ ; z^{(t)} = \sum_{j=1}^{x(t-1)} w_{i,j}^{(t)} y_i^{(t-1)}$$

A stochastic likelihood representation of each class is obtained by altering the weight parameters through the conjunction of convolutional, non-linearity, rectification, and pooling layers.

VVG16

A CNN model variation called VVG16 was put forth by Oxford University's large-scale image recognition research [14]. More than 14 million photos from 1000 categories are used to train the VVG16 which is aa advanced AlexNet model with larger filters. The first layer with eleven, second layer with five kernel sized filters. 3x3 size kernels follow this convolutional layer. The NVIDIA Titan Black graphical processing unit (GPU) is used to train VGG16 over several weeks [15]. Figure 3 shows the architecture of VVG16, which consists of 4 max-pooling layers, 3 dense layers, and 13 convolutional layers.

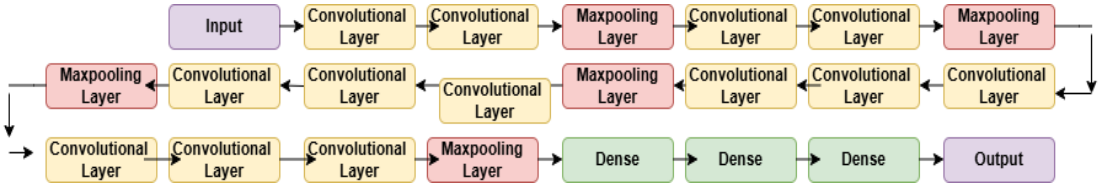

Figure 3: Architecture of VVG16

ResNet-50

Based on the CNN architecture, ResNet-50, also known as Residual net, is a member of the ResNet family. To increase precision, it delves further than VVG16. In addition to addressing the vanishing gradient problem, ResNet-50 outperforms other models in terms of results [16]. In order for the model to perform noticeably better than previous models, all layers are included in the stack design [17]. Figure 4 displays the ResNet-50 architecture.

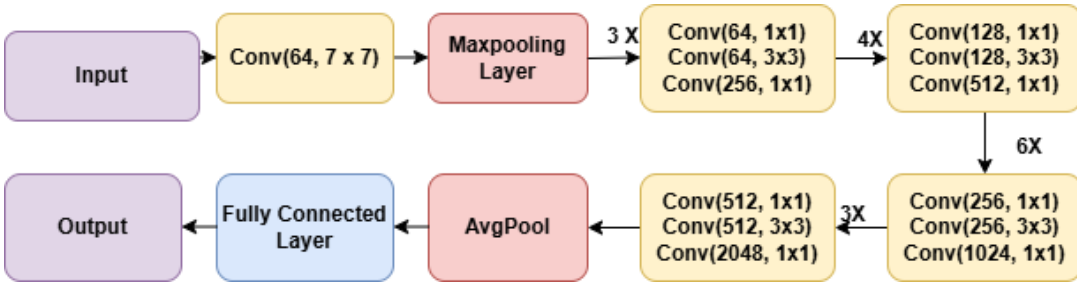

Figure 4: Architecture of ResNet

Quantum Machine Learning

Because machine learning approaches typically demand vast volumes of data, their constraints lie in their calculation time and storage requirements. Furthermore, the training time may increase even further with the use of contemporary deep learning techniques. Because of this, to make computations possible, researchers typically use Graphics Processing Units (GPUs). Using quantum computers is a next-generation approach to decreasing the storage and computing time for certain methods. Large-scale data processing has been addressed by quantum machine learning (QML), which makes use of quantum random access memory (QRAM) and quantum computing principles [19]. Due to these unique features, quantum computing is becoming a growing trend in a variety of technological domains.

In the conventional scenario, a transistor's state of charge is represented by the bit, which has two distinct states that it can adopt. These states are typically represented as 0 and 1. Alternatively, the quantum bit (or qubit) is the bit's counterpart in the quantum scenario, with qualities derived from quantum physics. Two fundamental ideas of quantum theory, superposition and entanglement, are responsible for the enormous computing power of quantum computers. Combining these two ideas enables the development of computer systems with high computational capacity and quick execution times. A two-qubit quantum system may store all four possible binary combinations (00, 01, 10, 11).

In a study published in [20], the authors identified four approaches for fusing ML and QC. These approaches are shown in Figure 5 and vary depending on whether a quantum (Q) or classical (C) system generated the data and whether a quantum (Q) or classical (C) information processing device is used. We address quantum data handled by traditional algorithms in this study. The algorithms for quantum machine learning may be entirely new or they may be a quantum adaptation of traditional machine learning. that solves problems in a quantum setting. Combining quantum and traditional algorithms should improve performance and lower learning costs. The type of input data can also be classified further, as it can be encoded using either a classical or quantum representation. This means that different data types result in

different hybrid techniques between the classical and quantum realms.

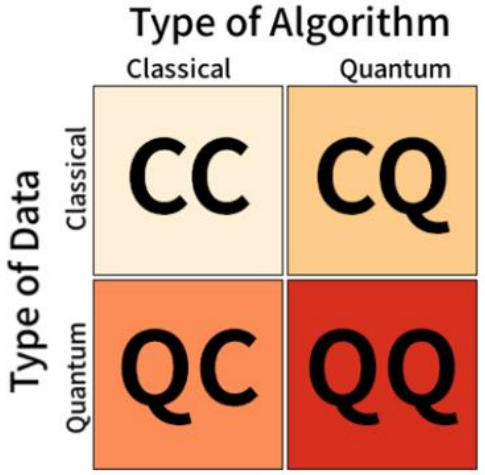

Figure 5: Approaches for combining ML and quantum computing paradigms.[21]

Variational Quantum Circuits

Utilizing the advantages of both quantum and classical computing, the variational quantum circuit, also known as the parameterized quantum circuit, is a hybrid quantum-classical technique. One of the main benefits of VQC-based QML over classical ML is the significant decrease in the number of model parameters, which may help to alleviate the overfitting issue that is often associated with classical ML. Furthermore, research has demonstrated that QML models have the potential to learn more quickly or attain greater testing accuracies than their classical counterparts in some scenarios. A classical and a quantum component are usually present in a contemporary QML design. It is one kind of tunable-parameter quantum circuit that is iteratively optimized by a computing classical system. These parameters form the weights in the ANN. The key components of a VQC include the following.

- Quantum Gates: Core of quantum circuit is the quantum gates. They manipulate the state of qubits, the basic units of quantum information. Gates can perform operations like rotations, entanglement, and conditional operations. Common gates include Pauli-X (X), Pauli-Y (Y), Hadamard (H), Controlled-NOT (CNOT), and Pauli-Z (Z). The parametric gates are defined as follows:

$$R_x(\theta) = \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) & -i\sin\left(\frac{\theta}{2}\right) \\ -i\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{bmatrix} R_y(\theta) = \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) & -\sin\left(\frac{\theta}{2}\right) \\ \sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{bmatrix} R_z(\theta) = \begin{bmatrix} e^{-i\left(\frac{\theta}{2}\right)} & 0 \\ 0 & e^{i\left(\frac{\theta}{2}\right)} \end{bmatrix}$$

- Qubits: The quantum counterparts of classical bits are called qubits. They can represent both 0 and 1 states simultaneously due to the principle of superposition. Qubits are at the core of quantum computation, enabling the encoding and manipulation of information in quantum states.

- Adjustable Parameters: The key feature of a VQC are its adjustable parameters. These parameters determine how quantum gates are applied to qubits. These parameters are varied

during the optimization process to guide the quantum circuit toward the desired quantum state or solution.

- Layers: A VQC often consists of multiple layers of quantum gates. Each layer applies a set of gates to the qubits. The choice of the quantity of layers and the type of gates in every layer can impact the circuit's expressiveness and its ability to solve specific problems.

- Objective Function: This quantifies the quality of the solution achieved by the quantum circuit. It's specific to the problem being solved. For example, in machine learning tasks, the objective function could be a measure of prediction accuracy. In quantum chemistry simulations, it might be the energy of a molecular system.

- Classical Optimizer: The classical optimizer is responsible for adjusting the parameters of the quantum circuit in an iterative manner to optimize the objective function. It uses classical optimization techniques, like gradient descent, to find the parameter values that yield the best solution.

- Measurement: Finally, qubits are measured to extract information. The measurement outcomes are probabilistic and provide information about the quantum state the circuit has prepared.

- Quantum Hardware/Backend: The VQC needs access to quantum hardware or simulator to perform the actual quantum operations. The choice of hardware impacts the performance and accuracy of the circuit.

Variational quantum circuit approach includes the following steps 1) Prepare initial state 2) Perform some unitary transformations with parameters. U(w) using the quantum circuit, a series of gates 3) Measure some observable quantity B The conversion of the input quantum state $|x\rangle$ to the output quantum state $|y\rangle$ is carried out by the parameterized circuit $|y\rangle = U(w)|x\rangle$ in accordance with the quantum circuit U(w), where w is the circuit parameter, such as the qubit rotation gate's angle. PQC offers a large deal of flexibility in the design of hybrid algorithms with classical and quantum counterparts by mixing various quantum gates [22][23]. The quantum-classical training approach is used to update the PQC parameters, as shown in Figure 6. Once the output state is generated by the PQC, the state is measured. These measured values are taken by the traditional computer and the loss value is computed. Classical computer iterates repeatedly to optimize the quantum circuit parameters until either the maximum number of iterations is reached, or the expectation value is satisfied [24].

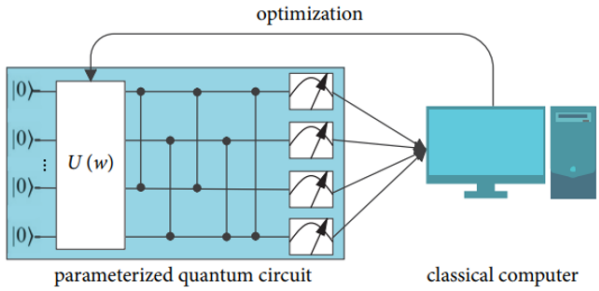

Figure 6: Schematic diagram of Parametrized Quantum Circuits

Quantum Convolutional Neural Network

Preparing the Quantum State: Since there is a limitation on the number of qubits, there is a necessity to reduce the dimension of the input data. Images must first be flattened and resized to m x m dimensions with pixel values of [0,1]. These operations result in 1 X m2 vector x = $[x_1, x_2, ..., x_{m2}]$ which is converted to angle information α, so $\alpha = \pi x$, $\alpha = [\alpha_1, \alpha_2, ... \alpha_{m2}]$. α is the rotation angle of gate R_y and the initial state is encoded for a m2 input.

$$|\varphi_{img}\rangle = \bigotimes_{i=1}^{m^2} R_y(\alpha_i)|0\rangle_1 \dots |0\rangle_{m^2}$$

Therefore, m x m qubits are required for a m x m image, and the quantum state is the encoding of every pixel in the m x m image.

Quantum Convolution Layer: Once $|\varphi_{img}\rangle$ is obtained, the quantum convolution kernel U(θ) performs unitary transformation on $|\varphi_{img}\rangle$.

Quantum Pooling layer: The output of a convolution layer is transferred to a qubit after pooling. The qubit can be measures to obtain the expected value. Nonlinearity is introduced by classical CNN via nonlinear functions whereas in a quantum-based system it's achieved through measurement. Ultimate quantum state $|\varphi_{out}\rangle$ can be attained as the quantum system's progression to the quantum state.

Z-based measurements is conducted on the output state $|\varphi_{out}\rangle$ to compute the expected value. Vector of Z-operators ($Z_1, \dots, Z_N$) applied on different qubits; V, unitary gate in the pooling layer; $U(\theta) = u_1(\theta)u_2(\theta) \dots u_l(\theta)$, l being the number of convolutions; for image of m × m size, l = (m – 1)2 the pooling unit conducts (m – 1)2 operations. Thus E, a vector with dimension 1 × (m – 1)2 is obtained by measuring the output of the quantum pooling layer. Since E is a vector made up of the Z expected values of various qubits and is unrelated to the label of the image, it should be entered into the fully connected classical layer for additional operations [26][27].

Generating Hybrid Network: Hybrid Quantum Classical CNN consists of a quantum convolutional layer, a quantum pooling layer, and a classical fully connected layer. Several convolution kernels make up the quantum convolutional layer to produce feature map, as seen in Figure 7. The quantum pooling layer reduces the convolution results, and the quantum pooling structure is the quantum pooling unit. The class of the image is then determined by measuring the qubits and entering the measurement data into the fully connected layer.

3. Experiments and Results

This part of the paper describes the dataset and discusses implementation and obtained results.

Dataset

The proposed system was evaluated using a Malimg, publicly available benchmark dataset containing greyscale images of various malware families having a varied distribution in the quantity of samples. The labels that are encoded span through 0 to 24 where the target class corresponds to a malware variant. It contains 9939 image samples generated from malware binaries of 25 malicious families. Pictures were of varying dimensions from 64 pixels x 200

pixels to 800 pixels x 800 pixels. For experimentation 80% of the images was used for training and 20% were used for testing.

Implementation

TensorFlow-quantum, Qiskit, PennyLane, and other development libraries for researchers have surfaced as a result of the active advancement of research in quantum computing and quantum machine learning. Among these, PennyLane was developed to facilitate research on quantum machine learning and makes it simple for anybody to use quantum simulators to test the functionality of quantum circuits. The PennyLane-supported quantum simulator enables the CPU to simulate QPU operations, with support for gradient operation and PyTorch tensor parameters [28]. These features make it simple for anyone who has used PyTorch for machine learning research in the past to begin studying quantum machine learning with PennyLane. In this research, a quantum convolutional neural model was developed with PyTorch and PennyLane based on this foundation.

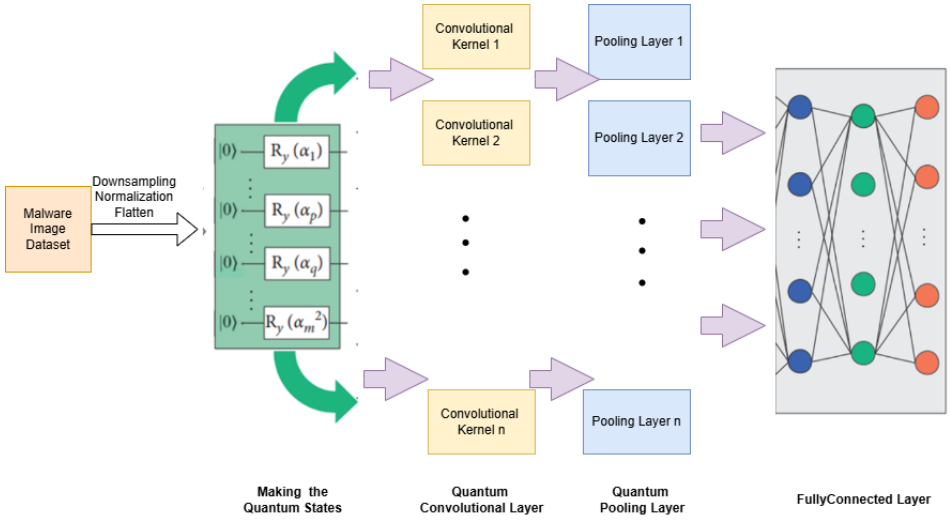

Figure 7: Hybrid Quantum Classical Convolutional Neural Network Architecture

On the default, the complete variational circuit is implemented. PennyLane provides a qubit simulator. Via the IBM QXplatform's Qiskit [28] quantum computing framework, IBM offers access to actual quantum hardware. Using the Pennylane-Qiskit [28] plugin, IBM QX platform is used for unitary processes related to state preparation. The packages required to combine Qiskit with PennyLane's quantum machine learning capabilities are provided by the Pennylane-Qiskit plugin. As a result, we can use the PennyLane package to write a Python program that uses quantum operations and run it on real-time quantum hardware. The real quantum hardware can be accessed using the sample code snippet that follows.

```
import pennylane as qml
dev = qml.device('qiskit.ibmq',
                 wires = 4,
                 backend = 'ibmq_16_melbourne',
                 ibmqx_token = "XXX")
```

A quantum device or quantum simulator can be initialized using the method qml.device to carry out quantum processes using the quantity of qubits fed into wires as input. The IBM QX platform's quantum hardware is utilized to carry out quantum operations, as indicated by the input parameter qiskit.ibmq. The backend receives the name of a particular quantum device to be used as input. By setting up an account on the IBM QX, one can get the IBM QX API token, which is required to undertake quantum operations. The token is then entered into the ibmqx_token function. On the chosen quantum hardware, all of the qubits are initially set to $|0\rangle|0\rangle$ by default. Following IBM's quantum device choices, qml.Hadamard and qml.QubitUnitary operations are employed in creating state. The interfaces required to conduct hybrid classical-quantum operations on the quantum simulator and the classical computing facility are provided by PennyLane.

Performance Evaluation

This stage presents a thorough performance analysis of the studied models using five different assessment indicators, accuracy, precision, recall, F1 score and AUC. Multi-class confusion matrix, differs from the conventional matrix of 2-way classification problems, can be used to estimate the TP, TN, FP, and FN values. This study presents a classification confusion matrix with N malware families (classes). For example, four different classification outcomes (actual and projected) can be obtained for a given malware family Fx.

4. Results

On the families of the Malimg dataset shown in Table 2, we tested the dataset using standard CNN, two CNN transfer learning models (VGG16, ResNet-50), and the hybrid Quantum CNN. We split the dataset, as we previously stated, into 80% for training and 20% for testing. The training and classification procedures made use of an Intel Core i7 processor with 8 GB of RAM. Using the Malimg dataset, we computed the F1-Score, AUC, accuracy (Acc.), precision (Prec.), recall (Rec.), and precision (Prec.) for every malware family. Next, we calculated these assessment metrics' average values.

We ran multiple experiments and experimented with various hyperparameters. Table 4 displays the outcomes of the experimental investigation, whereas Table 3 gives certain hyperparameter settings of the used models. The settings in Table 1 demonstrate that the smallest picture dimension that we took into consideration for the Hybrid-QCNN model. This is because the Hybrid-QCNN model requires more computational resources, which results in a longer run time than the other models. Table 3's results indicate that the VGG16 model is achieving the best performances. Given that the Hybrid-QCNN model was trained using a 25 x 25 image (whereas the VGG16 model was trained using an image with 110 × 110 dimensions), it is not surprising that the hybrid model performs comparably to the VGG16 model in terms of precision. Given that the Standard CNN model has a lighter architecture than the VGG16, its findings are encouraging and nearly identical to those of the Hybrid-QCNN. We note that, as shown in Table 1, the Hybrid-QCNN model is trained using 25x25 photos, but the VGG16 model uses 110x110 images.

Table 3. Model Settings

Model	Image size	Batch	Epoch	Training Time
Standard CNN	100 * 100	32	25	00:02:49
VGG 16	100 * 100	32	25	00:05:56
ResNet	100 * 100	32	50	00:03:59
Hybrid QCNN	25 * 25	32	20	07:43:21

Table 4: Model Performance Metrics

Models	Accuracy	Precision	Recall	F1Score	AUC
Standard CNN	96.89	96.99	96.88	96.99	98.3
VGG 16	97.32	97.01	96.85	97.49	98.3
ResNet	96.99	96.36	95.99	96.32	97.99
Hybrid QCNN	97.11	97.01	97.12	97.56	98

5. Conclusion

In this paper, the state-of-the-art for using deep learning to detect mobile malware is covered. In a short period of time, quantum computing has advanced quickly and is proving to be quite beneficial in a variety of disciplines. The scientific and industrial research communities are proposing new approaches for malware detection, specifically for the Windows platform and by utilizing deep learning techniques, in light of the insufficiency of current antimalware to detect new threats. The advent of quantum computing has made it feasible for models to learn by taking advantage of hybrid quantum and classical learning algorithms. We try to contrast the performance of deep learning models and a hybrid quantum model for classifying malicious payloads from image representation of the executables. We obtained the better result for this task from VGG16 transfer model. From the accuracy point of view, is the Hybrid QCNN follows. Owing to computational difficulties, we note that the VGG16 model was trained with an image of 100x100 dimensions, whereas the Hybrid-QCNN was trained with images of 25x25 dimensions. This difference in training results indicates that the HybridQCNN is a potential candidate for the malware detection challenge. Better quantum models may be explored in other studies.

Acknowledgement: We thank Soja Rani S and Dr Mouleeswaran for their contributions to this work.

References

1. Schuld, M.; Sinayskiy, I.; Petruccione, F. An introduction to quantum Machine Learning. Contemp. Phys. 2015, 56, 172–185. [CrossRef]
2. Ding Yuxin and Zhu Siyi. 2019. Malware detection based on deep learning algorithm. Neural Computing and Applications 31, 2 (2019), 461–472.
3. Martín-Guerrero, J.D.; Lamata, L. Quantum Machine Learning: A tutorial. Neurocomputing 2022, 470, 457–461. [CrossRef]
4. J.H. Go, T. Jan, M. Mohanty, O.P. Patel, D. Puthal, M. Prasad, Visualization approach for malware classification with ResNeXt, 2020 IEEE Congress on Evolutionary Computation (CEC), pp. 1–7.
5. D. Vasan, M. Alazab, S. Wassan, H. Naeem, B. Safaei, Q. Zheng, IMCFN: imagebased

malware classification using fine-tuned convolutional neural network architecture, Comput. Netw. 171 (2020)
6. A. Bensaoud, N. Abudawaood, J. Kalita, Classifying malware images with convolutional neural network models, Int. J. Netw. Secur. 22 (6) (2020) 1022–1031.
7. S. Saadat, V.J. Raymond, Malware classification using CNN-XGBoost model, in: Artificial Intelligence Techniques for Advanced Computing Applications, Springer, 2021, pp. 191–202.
8. Ö. Aslan, A.A. YILMAZ, A new malware classification framework based on deep learning algorithms, IEEE Access
9. Ciaramella, G.; Iadarola, G.; Mercaldo, F.; Storto, M.; Santone, A.; Martinelli, F. Introducing Quantum Computing in Mobile Malware Detection. In Proceedings of the 17th International Conference on Availability, Reliability and Security, Vienna, Austria, 23–26 August 2022; pp. 1–8
10. Nataraj, L., Karthikeyan, S., Jacob, G. and Manjunath, B., 2011. [online] dropbox.com.
11. Gibert, D., Mateu, C., & Planes, J. A hierarchical convolutional neural network for malware classification. In 2019 International Joint Conference on Neural Networks (IJCNN) (pp. 1-8). IEEE
12. Casolare, R.; Ciaramella, G.; Iadarola, G.; Martinelli, F.; Mercaldo, F.; Santone, A.; Tommasone, M. On the Resilience of Shallow Machine Learning Classification in Image-based Malware Detection. Procedia Comput. Sci. 2022, 207, 145–157. [CrossRef]
13. 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.
14. K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint, arXiv:1409.1556.
15. VGG16 - convolutional network for classification and detection, https:// neurohive.io/en/popular-networks/vgg16/.
16. Asam, M., Khan, S. H., Jamal, T., Zahoora, U., & Khan, A. (2021). Malware Classification Using Deep Boosted Learning. arXiv preprint arXiv:2107.04008
17. ResNet50-pre-trained models for image classification, https:// www.analyticsvidhya.com / blog / 2020 / 08 /top-4-pre-trained-models-for-imageclassification-with-python-code/.
18. M.N.S. Jahromi, P. Buch-Cardona, E. Avots, K. Nasrollahi, S. Escalera, T.B. Moeslund, G. Anbarjafari, Privacy-constrained biometric system for non-cooperative users, Entropy 21 (11) (2019) 1033.
19. J. Choi and J. Kim, "A tutorial on quantum approximate optimization algorithm (QAOA): Fundamentals and applications," in Proceedings of the IEEE International Conference on Information and Communication Technology Convergence (ICTC), 2019, pp. 138–142.
20. 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.
21. José D Martín-Guerrero and Lucas Lamata. 2022. Quantum Machine Learning: A tutorial. Neurocomputing 470 (2022), 457–461.
22. 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).
23. J. Choi, S. Oh, and J. Kim, "Quantum approximation for wireless scheduling," Applied Sciences, vol. 10, no. 20, 2020.
24. J. Choi, S. Oh, and J. Kim, "The useful quantum computing techniques for artificial intelligence engineers," in Proceedings of the IEEE International Conference on Information Networking (ICOIN), 2020, pp. 1–3.
25. S. Oh, J. Choi, and J. Kim, "A tutorial on quantum convolutional neural networks (QCNN),"

in Proceedings of the IEEE International Conference on Information and Communication Technology Convergence (ICTC), 2020, pp. 236–239.

26. J. Choi, S. Oh, and J. Kim, "A tutorial on quantum graph recurrent neural network (QGRNN)," in Proceedings of the IEEE International Conference on Information Networking (ICOIN), 2021, pp. 46–49.
27. S. Oh, J. Choi, J.-K. Kim, and J. Kim, "Quantum convolutional neural network for resource-efficient image classification: A quantum random access memory (QRAM) approach," in Proceedings of the IEEE International Conference on Information Networking (ICOIN), 2021, pp. 50–52.
28. Wei Li, Peng-Cheng Chu, Guang-Zhe Liu, Yan-Bing Tian, Tian-Hui Qiu, Shu-Mei Wang, "An Image Classification Algorithm Based on Hybrid Quantum Classical Convolutional Neural Network", Quantum Engineering, vol. 2022, Article ID 5701479, 9 pages, 2022. https://doi.org/10.1155/2022/5701479