Handwritten Numeric Image Classification with Quantum Neural Network using Quantum Computer Circuit Simulator

Achmad Benny Mutiara[1], Muhammad Amir Slamet[2], Rina Refianti[3], Yusuf Sutanto[4]
Faculty of Computer Science and Information Technology, Gunadarma University[1, 3, 4]
Department of Informatic Engineering, Gunadarma University[2]
Jl. Margonda Raya No. 100, Depok 16424, Jawa Barat[1, 2, 3, 4]

Abstract*—Quantum Computer is a computer machine using principles of quantum mechanics in doing its computation. The Quantum Computer Machine itself is still in the development stage and has not been deployed yet, however TensorFlow provides a library for hybrid quantum-classical machine learning called TensorFlow Quantum (TFQ). One of the quantum computing models is the Quantum Neural Network (QNN). QNN is adapted from classical neural networks capable of processing qubit data and passing quantum circuits. QNN is a machine learning model that allows quantum computers to classify image data. The image data used is classical data, but classical data cannot reach a superposition state. So in order to carry out this protocol, the data must be readable into a quantum device that provides superposition. QNN uses a supervised learning method to predict image data. Quantum Neural Network (QNN) with a supervised learning method for classifying handwritten numeric image data is implemented using a quantum computer circuit simulation using the Python 3.6 programming language. The quantum computer circuit simulation is designed using library Cirq and TFQ. The classification process is carried out on Google Colab. The results of training on the QNN model obtained a value of 0.337 for the loss value and 0.3427 for the validation loss value. Meanwhile, the hinge accuracy value from the training results is 0.8603 for the hinge accuracy value with training data and 0.8669 for the hinge accuracy validation value. Model testing is done by providing 100 handwritten number images that are tested, with 53 image data of number three and 47 image data of number six. The results obtained for the percentage of testing accuracy are 100% for the number three image and 100% for the number six image. Thus, the total percentage of testing is 100%.

***Keywords*—Image classification; quantum neural network; quantum computer; TensorFlow**

I. Introduction

Quantum Computer is a computer machine which uses quantum mechanics for doing its computation [10], [11]. The intensity of the most sophisticated computer now days, has not been satisfied human to possess a super speedy computer. It is not impossible; in the future the most sophisticated super computer will be left by many companies and replaced by quantum computer [10]. Quantum computer machine itself is still in developing phase and has not been spread, but TensorFlow supplies library to hybrid quantum-classical machine learning which is called TensorFlow Quantum (TFQ). One model of quantum computing is quantum neural network (QNN). QNN is adapted from classical neural network which is able to proceed qubit data and surpass quantum circuit. Qubit is a measurement unit data in quantum computer [1], [2], [3], [4], [5], [6].

QNN is a machine learning model that allows quantum computers to classify image data. The image data used is classical data, but classical data cannot reach a superposition state. So in order to carry out this protocol, the data must be readable into a quantum device that provides superposition. QNN uses a supervised learning method to predict image data [7], [9]. Supervised learning is an approach where trained data is already available, and consists of variable which is targeted [8].

Based on the background above, the following problems are formulated: How to implement the supervised learning method to classify handwritten numeric image data in the Quantum Neural Network (QNN) model in order to achieve a superposition state?

The paper focuses on implementing the quantum neural network (QNN) by using supervised learning method in order to classify numeric handwritten data image. QNN is modelled by using computer quantum simulator circuit which is designed by making use of python-based library Cirq and TFQ. Furthermore, the implementation of the classification process is carried out on Google Colab.

The limitations of the problem in this paper are:

1) Handwritten image data obtained from MNIST (Modified National Institute Of Standards and Technology) [12].

2) Handwritten image data used are handwritten images of number 3 and handwritten images of number 6 with 11,520 training data and 1,968 testing data.

3) The size of the handwritten image data used is 28x28 pixels which are converted to 4x4 pixels.

4) The resulting output is a Boolean number where the value 1.0 or true represents the number 3 and the value -1.0 or false represents the number 6.

5) The programming language used is Python 3.6 and utilizes the Keras, TensorFlow 2.1, Sympy, MatPlotlib, Numpy, and Collections libraries.

6) Quantum computation is performed using a quantum computer circuit simulator provided by TensorFlow Quantum (TFQ) and Cirq.

7) The learning method used is a supervised learning method.

In the rest of paper, we show the research methodology in Section II. In Section III, the implementation and testing of model is presented. The last section is conclusion of our research.

II. Research Methodology

The research method in this research is shown in the following Fig. 1.

A. *Collecting and Processing of Classical Data Image*

Data that is used to train model consist of classical image data. Those data are divided into 60.000 train data and 10.000 validity data. Classic image data is used constitute data which is obtained from MNIST database (Modified National Institute of Standards and Technology database).

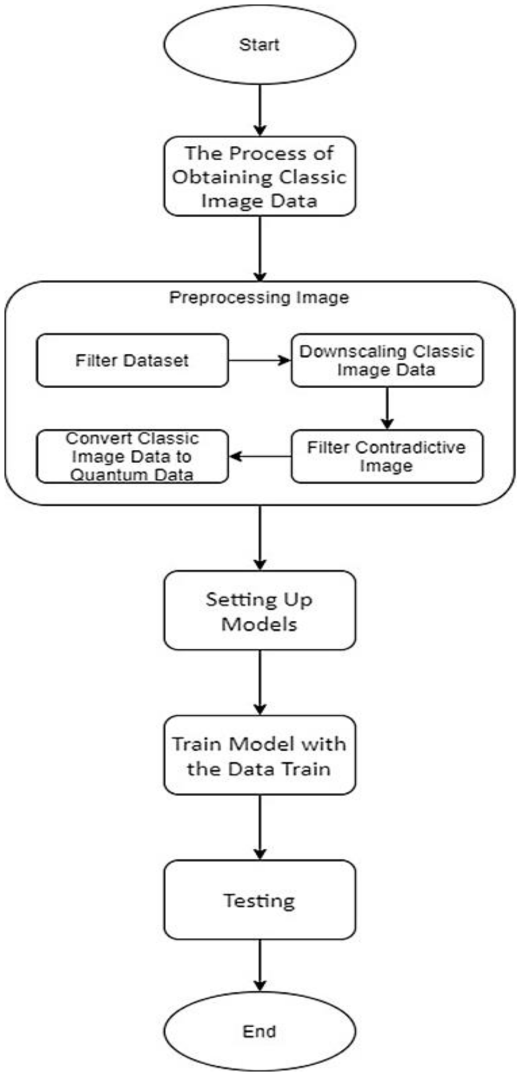

Start
The Process of Obtaining Classic Image Data
Preprocessing Image
Filter Dataset
Downscaling Classic Image Data
Filter Contradictive Image
Convert Classic Image Data to Quantum Data
Setting Up Models
Train Model with the Data Train
Testing
End

Fig. 1. Research Method.

B. *Preprocessing Data*

Pre-processing data image is done by the following steps:

1) Filtering datasets: In MNIST datasets consist of ten labels in various images of handwritten number. At the time of writing, quantum computation could only identify Boolean values, so the dataset needed to be filtered so that it could be operated on quantum circuits. The data used comes from dataset 3 and dataset 6, where dataset 3 contains data handwritten image number three and dataset 6 contains data handwritten image number six.

The filter-function is as in the following Fig. 2:

```
def Filter_36(x, y):
    keep = (y == 3) | (y == 6)
    x, y  =  x[keep], y[keep]
    y = y == 3
    return x, y
```

Fig. 2. Filter-Function for Label 3 and Label 6.

Furthermore, the training data and test data are filtered, and at the same time labey y is converted into Boolean numbers, i.e. “true” value for number three and “false” value for number six. This process reduced the number of training data to 12,049 and test data to 1,968.

2) Downscaling/Resizing classic image data: The classic MNIST image data is 28x28 pixels. This size is too large for the quantum computations running on conventional computers at this time. The image will be resized to 4x4 pixels so that it can be fed into the quantum circuit.

Image training data used during training is image data that has gone through the downscaling stage. Below are the before and after downscaling images, as seen respectively in Fig. 3 and Fig. 4.

3) Filtering contradictive image: Contradictory images are images labeled 3 and 6. This is done to reduce data barriers which will reduce accuracy when conducting training data on the model to be created. Fig. 5 shows the steps of filtering the contradictive image.

Classical class 3 and class 6 image dataset will be filtered, contradictory images will be deleted and non-contradictory images will be saved. Unique handwritten image data will be saved while image data that is more than one and classified into two (contradictory) labels will be deleted. This process reduced the number of training data to 11,520.

4) Convert classic image data to quantum data: Image processing using quantum computers requires quantum data in qubits. An image is made up of pixels, each pixel in the classic image data representing one qubit of data, with a state that depends on its pixel value. The first step is to convert the classic image data to binary coding (see Fig. 6).

Classical image data that has gone through the binary coding process will be converted into quantum data using the Cirq library (see Fig. 7).

After the classical image data becomes quantum data using circuits from Cirq, the circuits are converted to tensors (see Fig. 8).

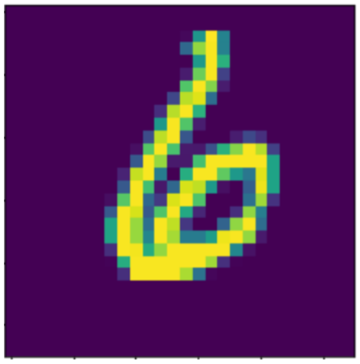


Fig. 3. Image before Downscaling.

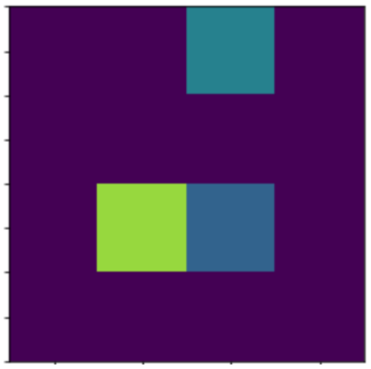


Fig. 4. Image after Downscaling.

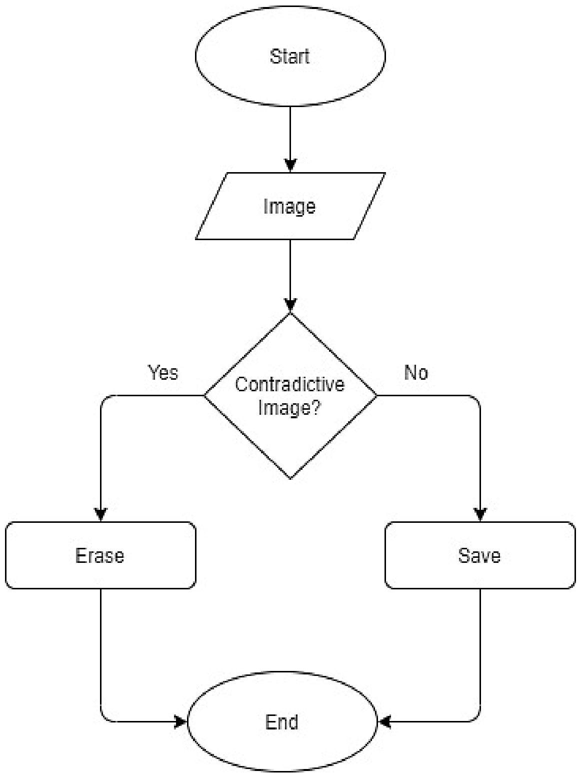
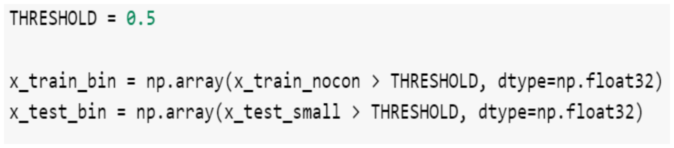
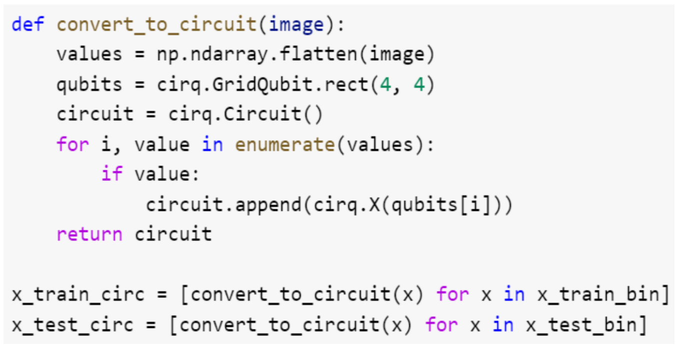


Fig. 5. Filter Contradictive Image Flowchart.



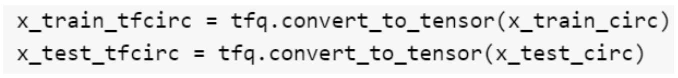
```
THRESHOLD = 0.5

x_train_bin = np.array(x_train_nocon > THRESHOLD, dtype=np.float32)
x_test_bin = np.array(x_test_small > THRESHOLD, dtype=np.float32)
```

Fig. 6. Binary Encoding.



```
def convert_to_circuit(image):
    values = np.ndarray.flatten(image)
    qubits = cirq.GridQubit.rect(4, 4)
    circuit = cirq.Circuit()
    for i, value in enumerate(values):
        if value:
            circuit.append(cirq.X(qubits[i]))
    return circuit

x_train_circ = [convert_to_circuit(x) for x in x_train_bin]
x_test_circ = [convert_to_circuit(x) for x in x_test_bin]
```

Fig. 7. Classic Data Convert to Quantum Data.



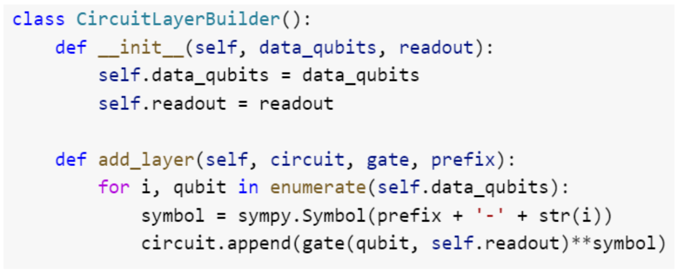
```
x_train_tfcirc = tfq.convert_to_tensor(x_train_circ)
x_test_tfcirc = tfq.convert_to_tensor(x_test_circ)
```

Fig. 8. Convert Cirq to Tensor.

C. *Setting up Model*

The quantum neural network (QNN) model that is built will be used to recognize images. This model uses two qubit gates, with continuously readout qubit. Thus, this model use layered approach; each layer will pass through the same gate. Each qubit data plays a role in the qubit readout.

The script above in Fig. 9 is a readout preparation for creating a gate layer, then entering it into the circuit in Fig. 10, as well as preparing for building a quantum model.



```
class CircuitLayerBuilder():
    def __init__(self, data_qubits, readout):
        self.data_qubits = data_qubits
        self.readout = readout

    def add_layer(self, circuit, gate, prefix):
        for i, qubit in enumerate(self.data_qubits):
            symbol = sympy.Symbol(prefix + '-' + str(i))
            circuit.append(gate(qubit, self.readout)**symbol)
```

Fig. 9. Build Gate Layer.

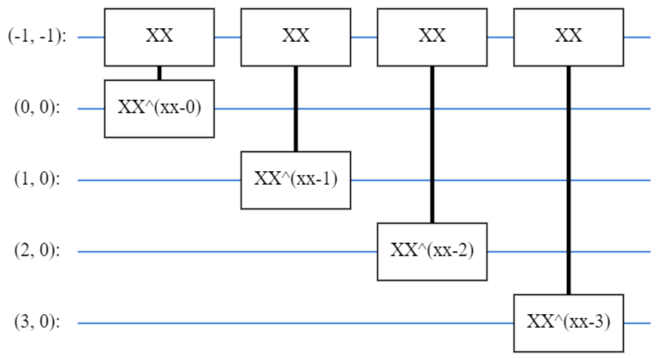


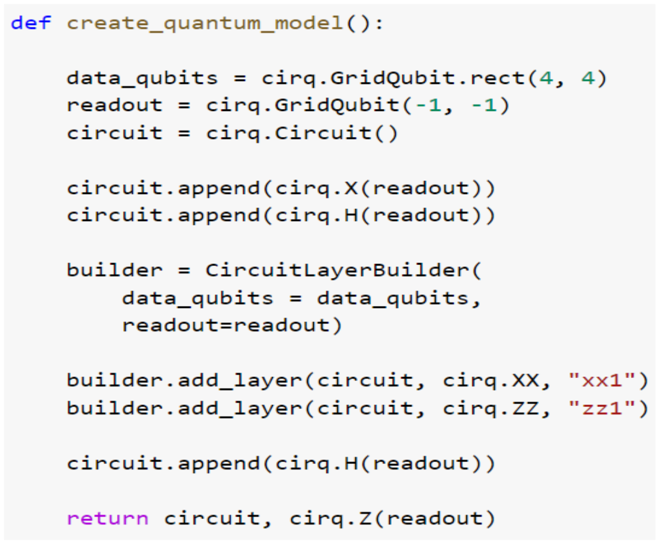
Fig. 10. Grid 4x4.

Data entered through gate XX, each qubit data will become one qubit that is read. The image used is 4x4 pixels in size. Each qubit in its pixel will be read to predict the image. After the XX gate circuit is built, next build a two layer model equal to the circuit size and enter the readout preparation that was made previously. The process of building the two layer moden is shown in Fig. 11.

The model is wrapped into the tfq-keras Model. Here, the Keras model is built with quantum components. This model uses quantum data by using the Parameterized Quantum Circuits (PQC) layer for training circuit models on quantum data [7].

Range readout from layer PQC is [-1, 1], thus, optimum *hinge loss* is required. Re-adjust is required. The data type label y_train_nocon which is initially Boolean will be converted to a matrix [-1, 1], according to the *hinge loss*. Threshold 0.0 is used to change data type when the output emerges, data type will return to be Boolean. Thus the output which comes out there is -1.0 for false value and 1.0 for true value.

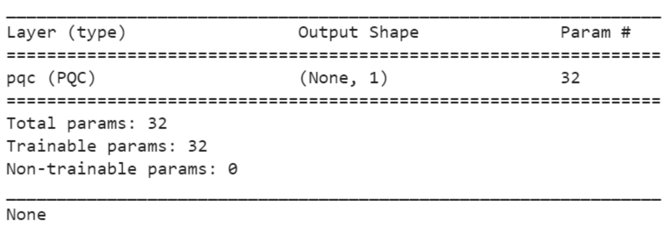
Describing training procedures to the model is carried out using the compile method. Image data originating from conventional computers will enter through the PQC layer, then the image data will be quantum processed and the processed data will be sent again to a conventional computer to display the output.



```
def create_quantum_model():

    data_qubits = cirq.GridQubit.rect(4, 4)
    readout = cirq.GridQubit(-1, -1)
    circuit = cirq.Circuit()

    circuit.append(cirq.X(readout))
    circuit.append(cirq.H(readout))

    builder = CircuitLayerBuilder(
        data_qubits = data_qubits,
        readout=readout)

    builder.add_layer(circuit, cirq.XX, "xx1")
    builder.add_layer(circuit, cirq.ZZ, "zz1")

    circuit.append(cirq.H(readout))

    return circuit, cirq.Z(readout)
```

Fig. 11. Quantum Model Function.

Fig.12 shows the summary of model that is built.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
pqc (PQC)                    (None, 1)                 32        
=================================================================
Total params: 32
Trainable params: 32
Non-trainable params: 0
_________________________________________________________________
None
```

Fig. 12. Model Summary.

D. *Training Model with the Data Train*

Model is trained by using trained image data and the output to be validated.

E. *Testing*

Classic image data handwritten which was trained will be tested by image. The output in the form of Boolean number where value of 1.0 or true represents number 3 and value of -1.0 or false represents number 6.

III. Implementation and Testing

Testing of the model is performed using image test data. The testing is recorded by observing the results of the QNN testing process which is processed using the supervised learning method to predict classical handwritten image data.

A. *Results of Model Training*

Fig. 13 shows a graph of the loss value from the results of the training that has been performed. The level of learning in the model built is 0.3370 for the loss value with training data and 0.3427 for the validation loss value.

Meanwhile, the hinge accuracy value from the training results is 0.8603 for the accuracy value on the training data and 0.8669 for the validation hinge accuracy value. The graph of the training accuracy and validation accuracy values can be seen in Fig. 14.

B. *Results of Model Testing*

Testing is performed by providing several images; the model will predict the answer. Image number 3 will be represented as true with an answer output of 1.0 and image number six 6 will be represented as false with an answer output of -1.0. The model will be tested with 100 handwritten numeric images. The test results of the test data can be seen in the Table I.

Based on the data from the results of the tests that have been carried out, the accuracy percentage can be obtained with the following calculations:

$$\text{Accuracy Percentage}=\frac{\text{The Quantity of True Testing Data}}{\text{The Quantity of Testing Data}}\times 100\%$$

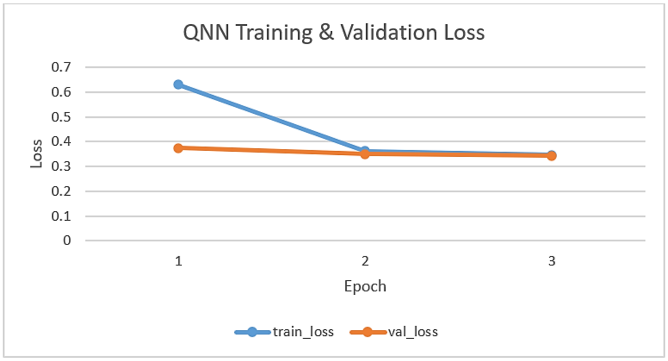


Fig. 13. Loss Value of the Training Process Diagram.

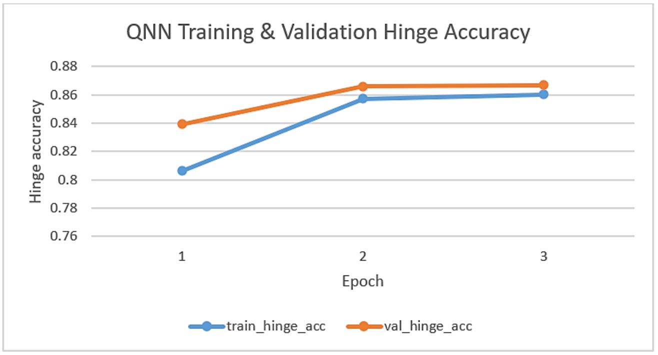
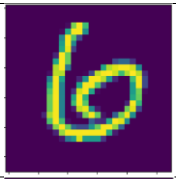
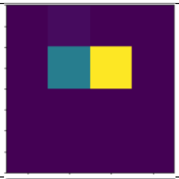
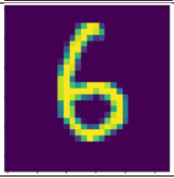
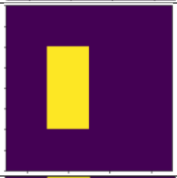
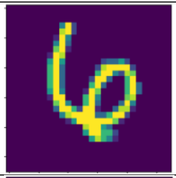
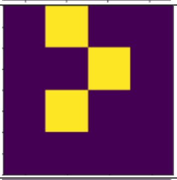
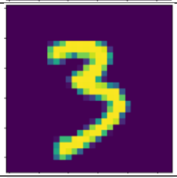
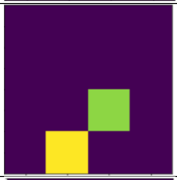
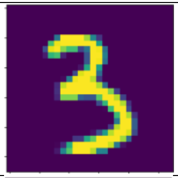
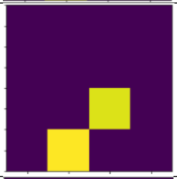
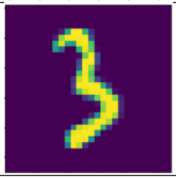
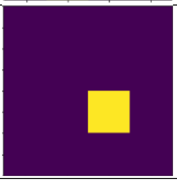
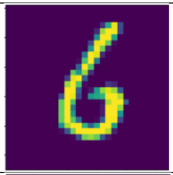
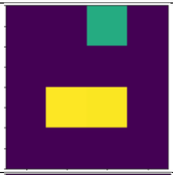
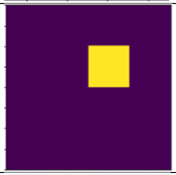
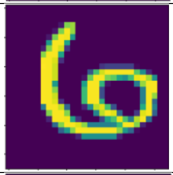
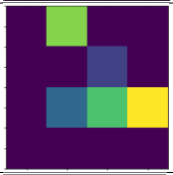
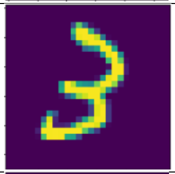
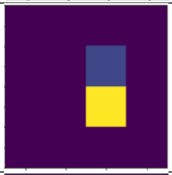
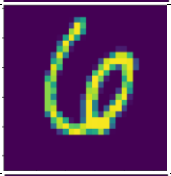
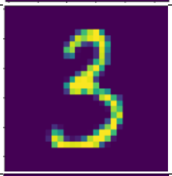
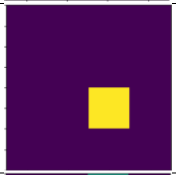
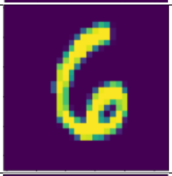
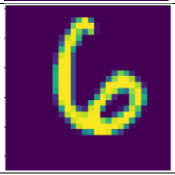
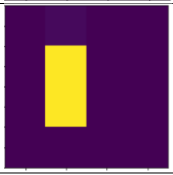
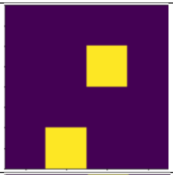
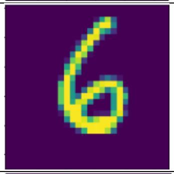
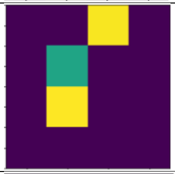
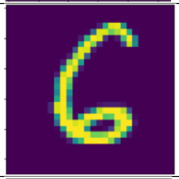
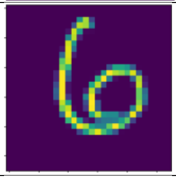
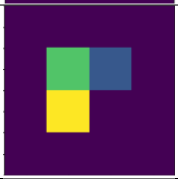
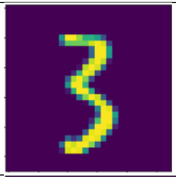
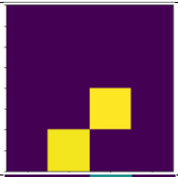
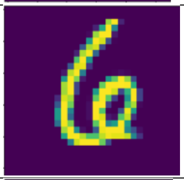
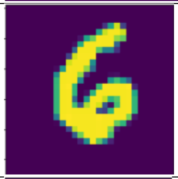
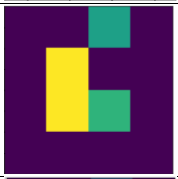
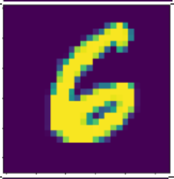
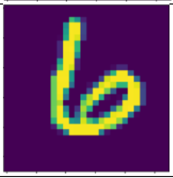
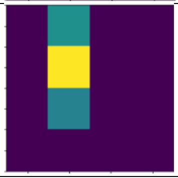
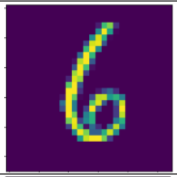
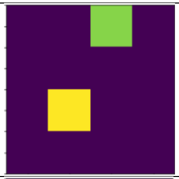
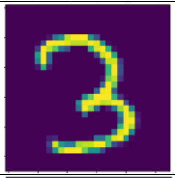
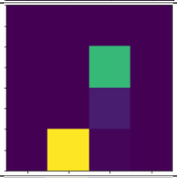
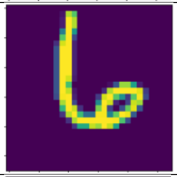
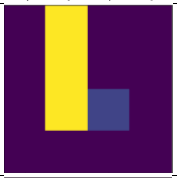
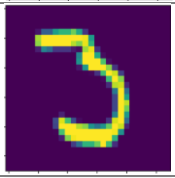
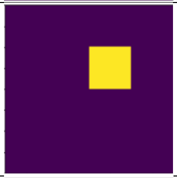
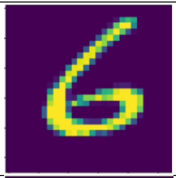
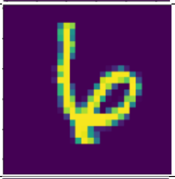
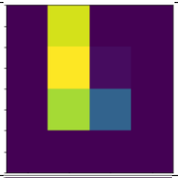
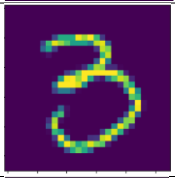
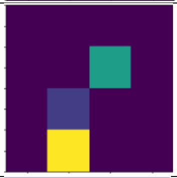
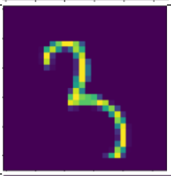
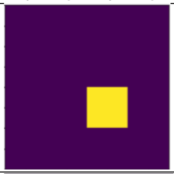
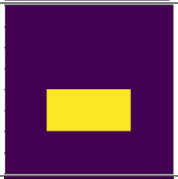
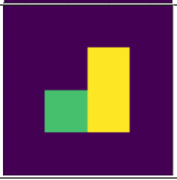


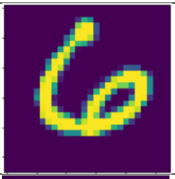
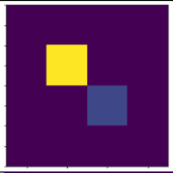
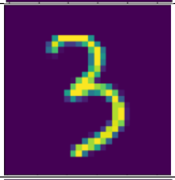
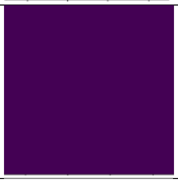
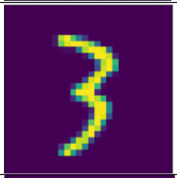
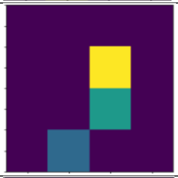
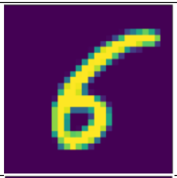
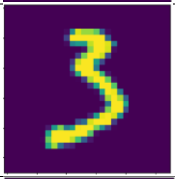
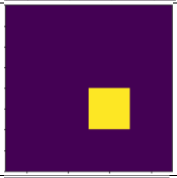
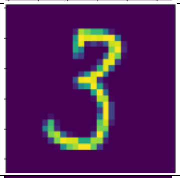
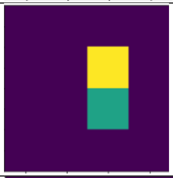
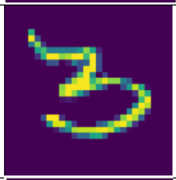
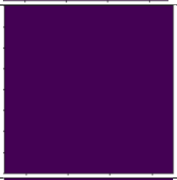
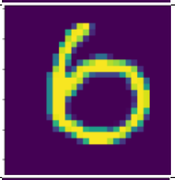
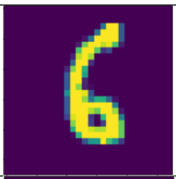
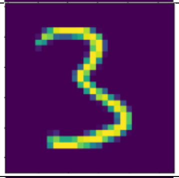
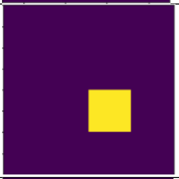
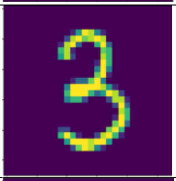
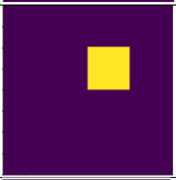
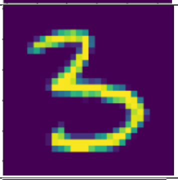
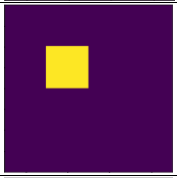
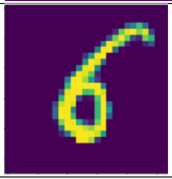
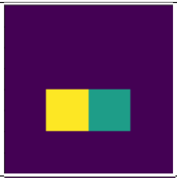
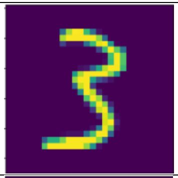
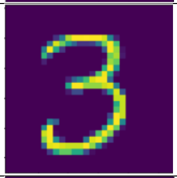
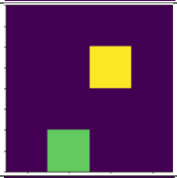
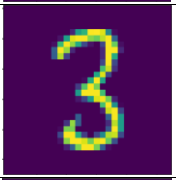
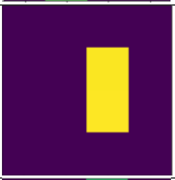
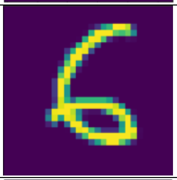
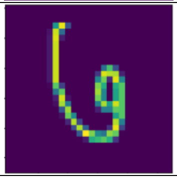
Fig. 14. Accuracy Value of the Training Process Diagram.

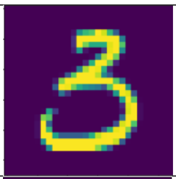
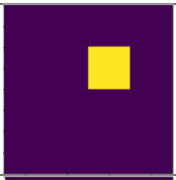
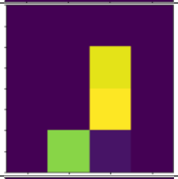
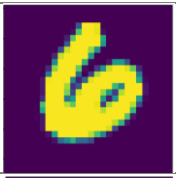
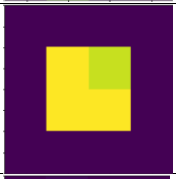
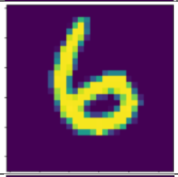
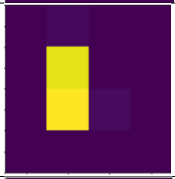
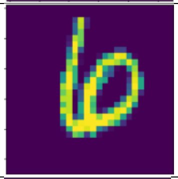
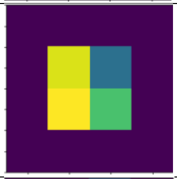
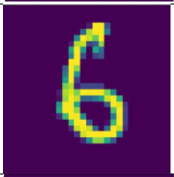
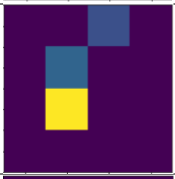
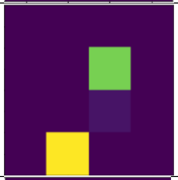
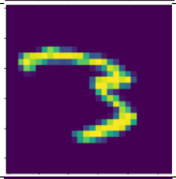
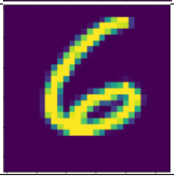
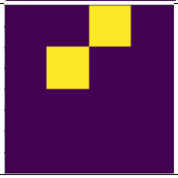
TABLE I. MODEL TEST RESULT

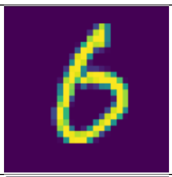
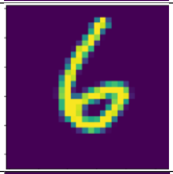
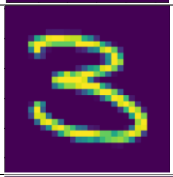
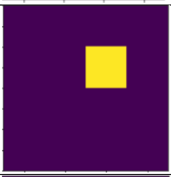
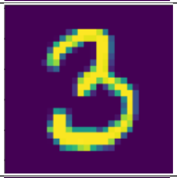
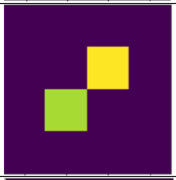
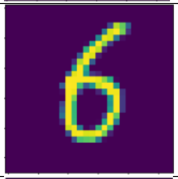
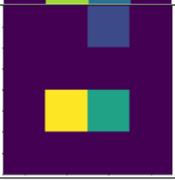
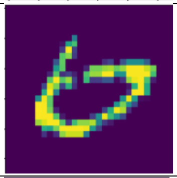
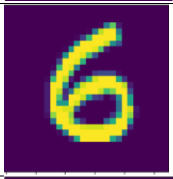
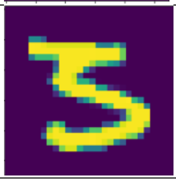
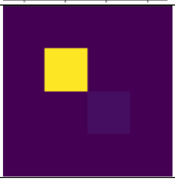
No.	Original Test Image	Test Image That the Model Read	Answer
1.			-1.0
2.			1.0
3.			-1.0
4.			-1.0
5.			1.0
6.			1.0
7.			1.0
8.			-1.0
9.			1.0
10.			-1.0
11.			1.0
12.			-1.0
13.			1.0
14.			1.0
15.			-1.0
16.			1.0
17.			-1.0

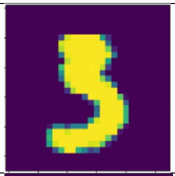
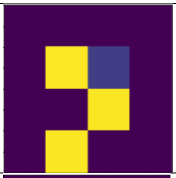
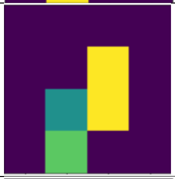
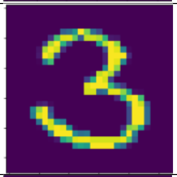
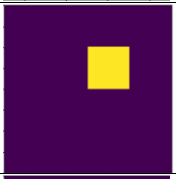
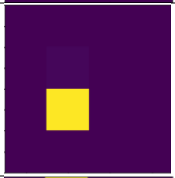
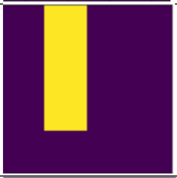
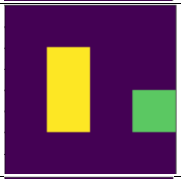
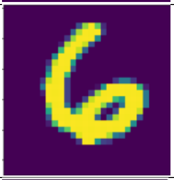
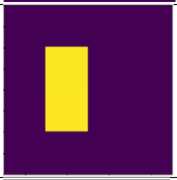
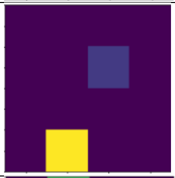
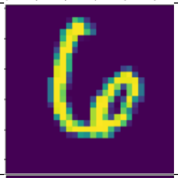
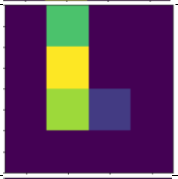
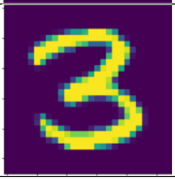
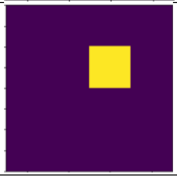
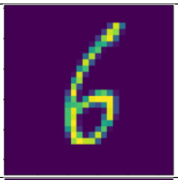
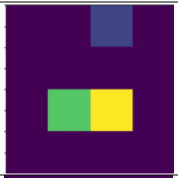
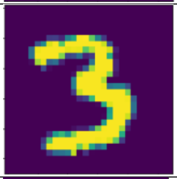
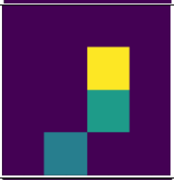
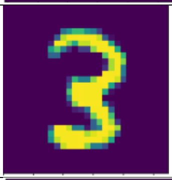
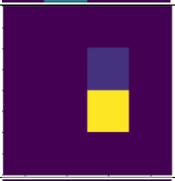
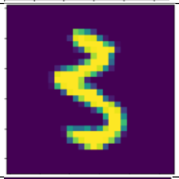
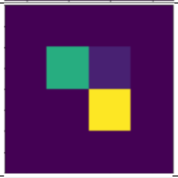
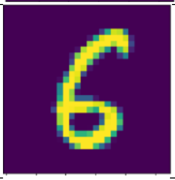
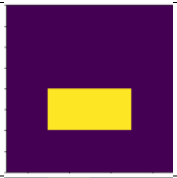
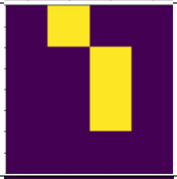
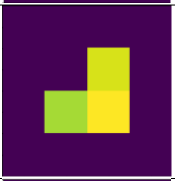
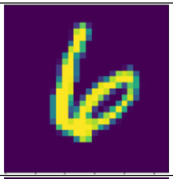
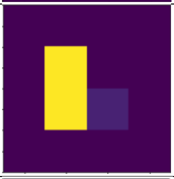
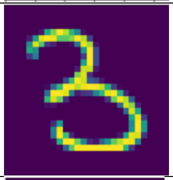
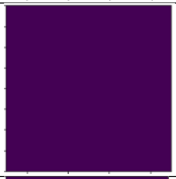
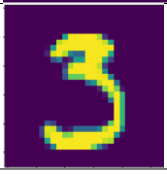
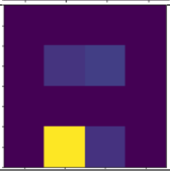
No.	Original Test Image	Test Image That the Model Read	Answer
18.			1.0
19.			-1.0
20.			1.0
21.			-1.0
22.			-1.0
23.			1.0
24.			-1.0
25.			-1.0
26.			-1.0
27.			-1.0

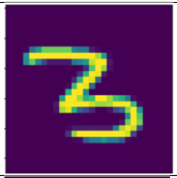
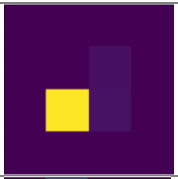
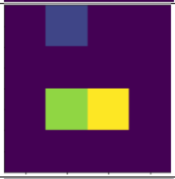
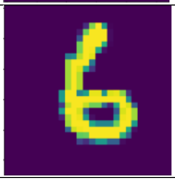
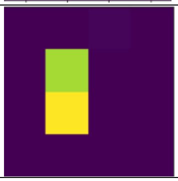
No.	Original Test Image	Test Image That the Model Read	Answer
28.			-1.0
29.			1.0
30.			-1.0
31.			1.0
32.			-1.0
33.			-1.0
34.			1.0
35.			1.0
36.			-1.0
37.			1.0

No.	Original Test Image	Test Image That the Model Read	Answer
38.			-1.0
39.			1.0
40.			1.0
41.			1.0
42.			-1.0
43.			1.0
44.			1.0
45.			1.0
46.			-1.0
47.			1.0
48.			-1.0
49.			1.0
50.			1.0
51.			1.0
52.			-1.0
53.			1.0
54.			1.0
55.			1.0
56.			-1.0
57.			-1.0

No.	Original Test Image	Test Image That the Model Read	Answer
58.			1.0
59.			1.0
60.			-1.0
61.			-1.0
62.			-1.0
63.			-1.0
64.			1.0
65.			1.0
66.			1.0
67.			-1.0

No.	Original Test Image	Test Image That the Model Read	Answer
68.			-1.0
69.			-1.0
70.			1.0
71.			1.0
72.			1.0
73.			-1.0
74.			1.0
75.			-1.0
76.			-1.0
77.			1.0

No.	Original Test Image	Test Image That the Model Read	Answer
78.			1.0
79.			1.0
80.			1.0
81.			-1.0
82.			-1.0
83.			-1.0
84.			-1.0
85.			1.0
86.			-1.0
87.			1.0
88.			-1.0
89.			1.0
90.			1.0
91.			1.0
92.			-1.0
93.			-1.0
94.			1.0
95.			-1.0
96.			1.0
97.			1.0

No.	Original Test Image	Test Image That the Model Read	Answer
98.			1.0
99.			1.0
100.			-1.0

The testing with 53 image data of number three represented by a value of 1.0 or true, the percentage of accuracy is obtained as follows:

$$\text{Accuracy Percentage}=\frac{53}{53}\times 100\%$$

Accuracy Percentage=100%

The testing with 47 image data of number six represented by a value -1.0 or false, the percentage of accuracy is obtained as follows:

$$\text{Accuracy Percentage}=\frac{47}{47}\times 100\%$$

Accuracy Percentage=100%

C. *Summary of Test Results*

From the results of tests that have been carried out using 100 handwritten numeric image data with the Quantum Neural Network (QNN) model, it shows that the accuracy results for each image type and the overall accuracy results are in the following Table II:

TABLE II. SUMMARY OF TEST RESULT

Type of Images	Last Percentage
Image of number three	100%
Image of number six	100%
Total	100%

IV. CONCLUSION

The implementation of the supervised learning method to classify handwritten numeric image data on the Quantum Neural Network (QNN) model in order to achieve a superposition state has been successful and the data can be read on a quantum computer circuit simulator.

The results of training on the QNN model obtained a value of 0.337 for the loss value and 0.3427 for the validation loss value. Meanwhile, the hinge accuracy value from the training results is 0.8603 for the hinge accuracy value with training data and 0.8669 for the hinge accuracy validation value.

Model testing is done by providing 100 handwritten numeric images that are tested, with 53 image data of number three and 47 image data of number six. The results obtained for the percentage of testing accuracy are 100% for the number three image and 100% for the number six image. Thus, the total percentage of testing is 100%. Based on the results of the total accuracy, the overall performance of the model is maximized.

ACKNOWLEDGMENT

This document is the results of the research project funded by the Ministry of Research and Technology / the national agency for Research and Innovation of Indonesia (Grant No. 079/SP2H/LT/DRPM/2020).

REFERENCES

[1] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mane, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shnlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viegas, F., Vinyals, O., Warden, P, Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. 2016. Tensorflow: Largescale Machine Learning on Heterogeneous Distributed Systems. https://arxiv.org/abs/1603.04467.

[2] Anonim. 2020. About Keras, https://keras.io/about/.

[3] Anonim. 2018. Apa Itu Colabotory?. https://colab.research.google.com/notebooks/welcome.ipynb?hl=id .

[4] Anonim. 2020. Quantum Machine Learning Concept. https://www.tensorflow.org/quantum/concepts.

[5] Anonim. 2020. Tensorflow Quantum. https://www.tensorflow.org/quantum/overview.

[6] Anonim. 2020. Tensorflow Quantum Design. https://www.tensorflow.org/quantum/design.

[7] Benedetti, M., Lloyd, E., Sack, S., and Fiorentini, M. 2019. Parameterized Quantum Circuit As Machine Learning Models. https://arxiv.org/abs/1906.07682.

[8] Fahriz. 2019. Mengenal Jenis Pembelajaran Mesin Supervised Learning dan Unsupervised Learning. https://medium.com/kelompok1 /mengenal-jenis-pembelajaran-mesin-supervised-learning-dan-unsupervised-learning-c588881e8ef5.

[9] Farhi, E. and Neven, H. 2018. Classification with Quantum Neural Networks on Near Term Processors. California: Google.Inc. URL: https://arxiv.org/pdf/1802.06002.pdf.

[10] Gotarane, V. and Gandhi, S. 2016. Quantum Computing: Future Computing. Mumbai: P.G. Scholar A.R.M.I.E.T. Vol. 3. https://www.researchgate.net/publication/328133812_Quantum_Computing_Future_Computing.

[11] Hidary, J. 2019. Quantum Computing: An Applied Aproach. California: Springer Nature Switzerland AG.

[12] LeCun, Y., Cortes, C., and Burges, C. 2016. The MNIST Database of Handwritten. http://yann.lecun.com/exdb/mnist/.