Supervised Learning Using Quantum Technology

Daniel Jaroszewski[1], Benedikt Sturm[1], Wolfgang Mergenthaler[1], Kay Steffen[1],
Patrick Linke[2], Michael Regel[3], Björn Haack[2]

[1] *FCE Frankfurt Consulting Engineers GmbH, Frankfurt am Main, Germany*
daniel.jaroszewski@frankfurtconsultingengineers.de
benedikt.sturm@frankfurtconsultingengineers.de
wolfgang.mergenthaler@frankfurtconsultingengineers.de
kay.steffen@frankfurtconsultingengineers.de
[2] *DB Systemtechnik GmbH, Brandenburg an der Havel, Germany*
patrick.linke@deutschebahn.com
bjoern.haack@deutschebahn.com
[3] *DB Regio AG, Berlin, Germany*
michael.regel@deutschebahn.com

ABSTRACT

In this paper, we present classical machine learning algorithms enhanced by quantum technology to classify a data set. The data set contains binary input variables and binary output variables. The goal is to extend classical approaches such as neural networks by using quantum machine learning principles. Classical algorithms struggle as the dimensionality of the feature space increases. We examine the usage of quantum technologies to speed up these classical algorithms and to introduce the new quantum paradigm into machine diagnostic domain. Most of the prognosis models based on binary or multi-valued classification have become increasingly complex throughout the recent past. Starting with a short introduction into quantum computing, we will present an approach of combining quantum computing and classical machine learning to the reader. Further, we show different implementations of quantum classification algorithms. Finally, the algorithms presented are applied to a data set. The results of the implementations are shown and discussed in the last section.

1. INTRODUCTION

FCE Frankfurt Consulting Engineers GmbH and DB Systemtechnik GmbH (DB ST) work together in the field of preventive maintenance for the last seven years. In particular, the continuous analysis of fault memory data is an essential part of our work. We use maintenance information to categorize operational data into relevant error classes, which act as output variables. Further, we take the entire fault memory (or a subset of it) as input variables. After substantial extraction and state vector labeling, we use neural networks to calibrate the input-output relationship. The training duration depends of the size the corresponding network and the amount of data. The so-called 'curse of dimensionality' is a limiting factor and influences the computing time.

Quantum computing seem to solve problems with large solution spaces in less computing time as conventional computers. While conventional computers solve computing tasks in a row, quantum computers can operate highly parallel. A conventional computer with for instance 50 bits, can be in exactly one (of 2^{50}) state in time, and only operate iteratively. A quantum computer with 50 fully connected qubits can be in 2^{50} states at the same time. This is the reason for an immense speed-up using quantum technology. Because many industrial problems are of high combinatorial complexity (so called 'NP-hard problems') the installed algorithms probably find local (sub-) optimal solutions. Sub-optimal because it is unknown, if there is another better solution for the problem (limited by the finite computing time). Quantum computers enable to find global solutions for certain NP-hard problems.

2. DATA SET

The data set used in the paper includes the fault memory data of 46 vehicles operated by DB ST. The training set contains the time period from July 2014 until end of December 2018. The evaluation set contains data from the beginning of 2019 until end of September 2019 (see figure 1). Fault memory data is binary, recording positive flag if a previously defined

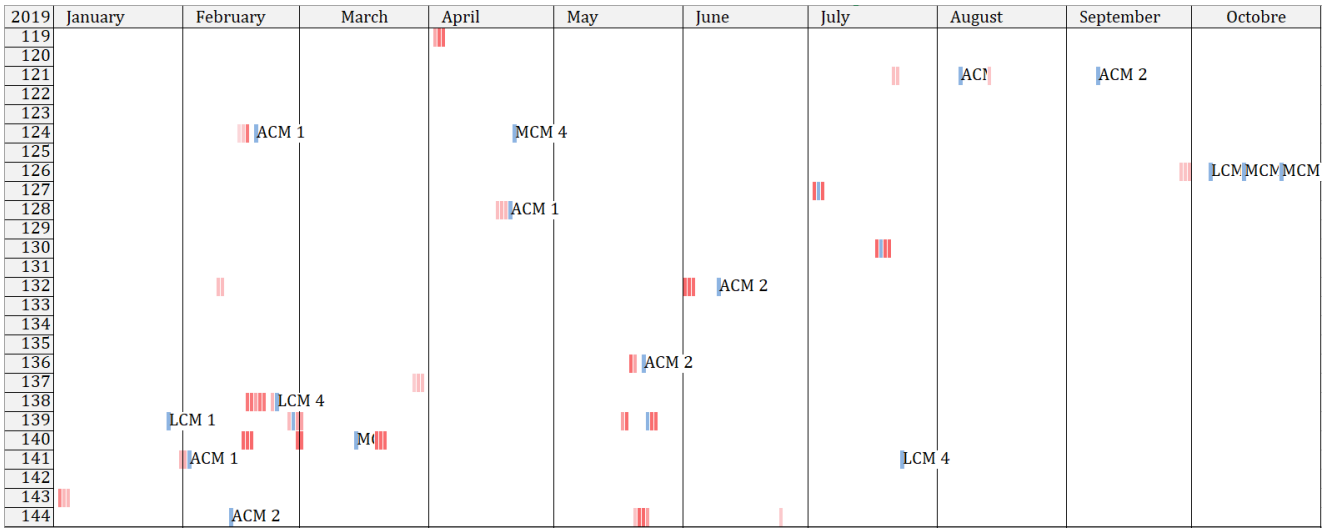

Figure 1. Overview of period including predictions (red) and maintenance events (blue) for whole fleet.

barrier within a measured value (here we are talking about continuous sensor data) is exceeded. The fault memory data also include measurement series from approx. 6500 sensors. Accordingly, the state space is highly dimensional.

The response data are the observed problem classes collected by the maintenance units of the underlying industry. The operating year 2019 (until end of September) includes exactly twenty failures (of eight failure classes). The following figure 2 shows the line current module and its different components.

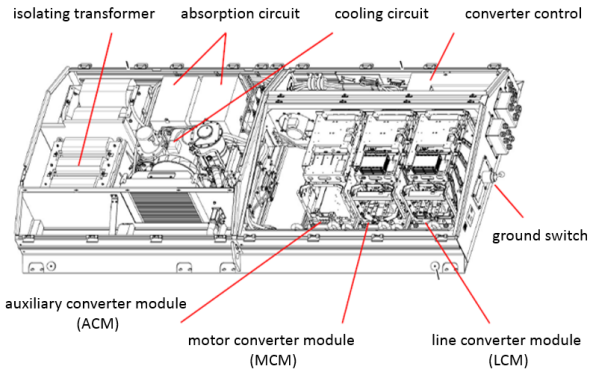

Figure 2. Line current module at the top of a train.

For the analysis in the following sections, fault memory data is reduced on the basis of physical and vehicle-specific relationships. In a further pre-processing step, the amount of inputs is also automatically restricted using statistical criteria. All further pre-processing steps are described in the sections on the classical and quantum-based approach.

3. CLASSICAL APPROACH

In the first part of this paper we deal with a classical approach to classification using neural networks. Before we take a closer look at the neural networks and the pre-processing of the data, we would like to talk briefly about the classification in the context of this paper.

Classification, as a sub-area of supervised learning, deals with the objective of assigning a data point from an underlying data set to a specific class. In the training step, an attempt is made to learn the best possible connection between input data (images, time series data, etc.) and output data (a class, here a binary characteristic). This relationship can be mapped using a mathematical classification (such as logistic regression) or methods from the field of artificial intelligence (such as neural networks). Ultimately, it is an input-output relationship. The output data is complete (and usually error-free) in the training step. In the application step, an unknown data point is evaluated and automatically assigned to one of the classes learned. The better a classification procedure, the more data points are classified correctly, and the fewer errors occur during the classification. The classification is used in many areas, such as image recognition (differentiation between 'dog' and 'cat') or preventive maintenance (to predict the failure of large systems).

In this paper, as described in the previous section, we examine a data set from the area of preventive maintenance. Before the data can be evaluated using a neural network, the data has to be pre-processed. To do this, we combine all input data into several input vectors as follows.

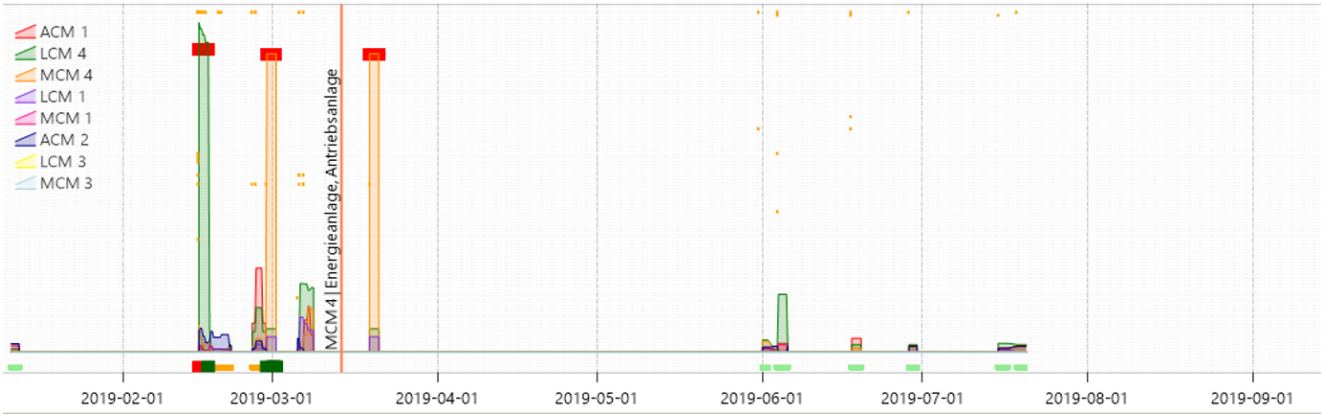

Figure 3. Multi dimensional forecast of maintenance events for a certain vehicle.

In the pre-processing step, the neural network sweeps across the entire amount of data with a fixed step size (in this paper it is six hours) and the number of events occurring within a time interval is counted by sensors. Furthermore, there is the possibility of taking the time development within a sensor into account. This is done by dividing a time interval into two sub intervals. As before, the number of events is counted within a sub interval. The two counters (from the respective sub intervals) are also saved as two vector entries. Thus, the number of vector entries of each vector is doubled by the addition of a simple temporal pattern. In addition to summing the events within a time interval, further aggregations are possible.

Each input vector is labeled using an output vector. Each component of the output vector represents a different class of error. This depends on the choice of the user or the number of error types recorded. We cover two cases in the paper. In the first case we work with all available error classes and in the second case we generate a one-dimensional output vector for the quantum computer example. Figure 3 shows a detailed part of the evaluation. The vertical red line represents a maintenance action and the small red rectangles are representing a warning.

The neural networks used here are part of the open source project 'TensorFlow' from Google. These can easily be controlled and used via a Python interface. The user can build and test his own neural network with a high degree of flexibility.

This paper uses a neural network with the following configuration:

- Number of input neurons equal to dimension of input vector
- Three hidden layers (128, 64 and 16 neurons)
- Number of output neurons equal to dimension of output vector
- 25 training epochs

The evaluation of our best configuration was evaluated in cooperation with DB ST from January to September 2019. We have the following key figures:

- 24 vehicles
- 20 maintenance events during period (January - September 2019)
- 8 of 20 predicted events before occurrence (40% detection rate*)
- 53 warnings depending on neural network
- 37 of 53 successful warnings (70% Precision := $\frac{tp}{tp+fp}$)

(* Detection rate: Ratio of detected maintenance events versus all maintenance events. As opposed the well-known key performance indicator recall := $\frac{tp}{tp+fn}$.)

4. QUANTUM APPROACH

In this section we give a brief introduction to the subject of quantum computers. First, we will dive into the theory of quantum computing. Then we will discuss the pre-processing steps necessary and finally we will present a first approach to an implementation on a quantum computer. The underlying theory can be mainly found in Fahri (2018). Furthermore, we use the conventional Dirac notation from Rieffel (2011) or Scherer (2016) to describe the mathematical processes within quantum information technology.

Conventional computers and quantum computers are fundamentally different. While classical computers are based on physical elements such as integrated circuits (the flow of information takes place via the flow of electricity), quantum computers are based on the principles of quantum mechanics. The fundamental elements in this case are called qubits. Qubits can be implemented in various ways. The particular representation is not relevant for the rest of this paper.

The main difference between classical architecture (bits) and quantum computers (qubits) are the respective properties of bits and qubits. While a classic bit can only be in one state (0 or 1), qubits can be in both states at the same time. This phenomenon is called 'superposition'. Furthermore, *n* classical bits can be in exactly one of 2^n possible states. In a way, qubits, on the other hand, can be in all 2^n states at the same time.

The second property, which distinguishes qubits from conventional bits, is called 'entanglement'. Two (or more) qubits can be entangled. This means that one qubit affects another qubit. If we measure the state of the first qubit, we can immediately make a statement about the state of the second qubit without measuring it. Mathematically speaking, the state of a pair of entangled qubits cannot be written as the product of the individual state vectors. Measuring a qubit, however, destroys the superposition property and provides a unique value. Before measurement, a system of qubits is in any of its possible states with a certain probability. This means that conventional computers perform arithmetic operations one by one, while quantum computers work in parallel to a high degree.

We write a qubit that is in a superposition and we assume the two states 0 and 1 with the same probability, as follows

$$\frac{1}{\sqrt{2}} \cdot (|0\rangle + |1\rangle) \tag{1}$$

The system of two entangled qubits could look like this:

$$\frac{1}{\sqrt{2}} \cdot (|00\rangle + |11\rangle) \tag{2}$$

While the first digit in the ket notation $|00\rangle$ represents the state of the first qubit, the second digit represents the state of the second qubit. It can quickly be seen that reading the first qubit provides immediate information on the second qubit: If the first qubit is in state 0, then the second qubit is also in state 0. The various terms $|xx\rangle$ correspond to the states of the underlying state space.

If we now connect several qubits to a system, one speaks of a quantum register or quantum computer. Various operations, such as the creation of superpositions, are carried out via quantum gates: interfaces for manipulating qubits. For a more detailed theory on the physical principles of quantum theory, see Nielsen (2010) or Yanofsky (2008).

Research creates hope that quantum computers will assume substantial progress in the area of solving combinatorial optimization questions, molecular biology (especially materials science) and some more. Furthermore, attempts are being made to implement existing algorithms from the field of artificial intelligence on quantum computers. So now we come to the main focus of this paper, the neural networks on quantum computers.

Our analysis is heavily influenced by the work as presented in Fahri (2018). Let us start with pre-processing the data. As already presented in the second section, we use the same but reduced data set for the calculations on the quantum computer. However, since actual quantum computers are not yet able to handle such large amounts of data, we reduce the input data to sixteen parameters. Then the vector entries are converted into binary variables. If the value is above a certain limit, the entry is set to one, if the value is below the limit, the entry is set to minus one.

Now we come to the mathematical approach of this work. Let the data be prepared as above. Then each input vector can be written as a sixteen-dimensional bit string $z = z_1 ... z_n$. Furthermore, exactly one label is assigned to each bit string $l(z) = \{-1, 1\}$. Exactly sixteen qubits are therefore required. A seventeenth qubit, which is the output qubit, is also used. We are now looking for a collection of L unitary matrices $U(\theta)$, which approximate the relationship between the input data z and the label $l(z)$. We construct the input for the quantum computer as follows

$$|z, 1\rangle = |z_1 z_2 \dots z_n, 1\rangle \tag{3}$$

and search

$$U(\overrightarrow{\theta}) = U_L(\theta_L) \cdot \ldots \cdot U_1(\theta_1) \tag{4}$$

so that

$$U(\overrightarrow{\theta}) \, |z, 1\rangle = l(z). \tag{5}$$

Y_{n+1} is the measurement of the $(n+1)$-th qubit, which corresponds to the function value. In order to get a probability distribution of the output, we repeat the calculations several times and measure Y_{n+1}. We get the averaged result

$$\langle z, 1| \, U^\dagger(\overrightarrow{\theta}) Y_{n+1} U(\overrightarrow{\theta}) \, |z, 1\rangle \tag{6}$$

In the calibration step, the following objective function must be minimized

$$loss(\overrightarrow{\theta}, z) = 1 - l(z) \, \langle z, 1| \, U^\dagger(\overrightarrow{\theta}) Y_{n+1} U(\overrightarrow{\theta}) \, |z, 1\rangle \tag{7}$$

This can be achieved through a gradual gradient process:

1) Start with a random set $\overrightarrow{\theta}$ of unitary matrices.

2) Choose a string z^1.
3) Initialize the quantum computer with $U(\overrightarrow{\theta})z^1$ and measure Y_{n+1}.
4) Calculate the objective function $loss(\overrightarrow{\theta}, z)$.
5) Change $\overrightarrow{\theta}$ according to the steepest descent algorithm.
6) Repeat steps 2) to 5) for all further strings z^2 to z^n.

We set some requirements for the unitary matrices. For more details, see Fahri (2018). The following then applies to the calculation of the first derivative for the k-th unitary matrix

$$\frac{\Delta loss(\overrightarrow{\theta}, z)}{d\theta_k} = 2Im\left(\langle z,1|\mathcal{U}(\overrightarrow{\theta})|z,1\rangle\right) \tag{8}$$

with

$$\mathcal{U}(\overrightarrow{\theta}) = U_1^\dagger \cdots U_L^\dagger Y_{n+1} U_L \cdots U_{k+1}\Sigma_k U_k \cdots U_1 \tag{9}$$

With a little trick we can measure the right side of the term with the auxiliary of a help qubit

$$|z,1\rangle \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \tag{10}$$

multiply by imaginary $i \cdot \mathcal{U}(\overrightarrow{\theta})$ to the auxiliary qubit

$$|z,1\rangle \frac{1}{\sqrt{2}}(|0\rangle + i \cdot \mathcal{U}(\overrightarrow{\theta})|z,1\rangle|1\rangle) \tag{11}$$

and then insert a Hadamard gate. So, we get the following state

$$\frac{1}{2}\left(|z,1\rangle + i \cdot \mathcal{U}(\overrightarrow{\theta})|z,1\rangle|0\rangle\right)$$

$$+\frac{1}{2}\left(|z,1\rangle + i \cdot \mathcal{U}(\overrightarrow{\theta})|z,1\rangle|1\rangle\right). \tag{12}$$

If we now measure the auxiliary qubit, we get the value zero with the probability of

$$\frac{1}{2} - \frac{1}{2}Im\left(\langle z,1|\mathcal{U}(\overrightarrow{\theta})|z,1\rangle\right). \tag{13}$$

By repeating the previous step and measuring the auxiliary qubit again, we get a good statement about the k-th component of the gradient and can change them.

The procedure just presented depends heavily on the number of qubits available and the decoherence time. The application has shown that the first approach is currently difficult to implement. Accordingly, we present a second approach, which can be implemented in the current state of research. We use this approach in this paper for the further classification of the data set. At the beginning, the pre-filtered data is placed in a single superposition (see figure 4). This is done by iteratively reading in the individual data points

$$|+1\rangle = c_+ \cdot \sum_{z:l(z)=1} \exp(i\psi_z)|z,1\rangle \tag{14}$$

$$|-1\rangle = c_- \cdot \sum_{z:l(z)=-1} \exp(i\psi_z)|z,1\rangle \tag{15}$$

where c_+ and c_- are factors for normalization. Figure 4 shows a schematic representation of the above initialization.

In order to find an optimal set of unitary matrices $U(\overrightarrow{\theta})$ the following term must be minimized with regard to $\overrightarrow{\theta}$

$$1 - \frac{1}{2}\left(\langle +1|U^\dagger(\overrightarrow{\theta})Y_{n+1}U(\overrightarrow{\theta})|+1\rangle\right)$$

$$+\frac{1}{2}\left(\langle -1|U^\dagger(\overrightarrow{\theta})Y_{n+1}U(\overrightarrow{\theta})|-1\rangle\right). \tag{16}$$

This can be implemented using a gradual gradient process.

The theory listed is implemented with the help of the open source project "TensorFlowQuantum" (TFQ) from Google. A detailed introduction to the technology of TFQ is described in Broughton (2020). The calibration step (gradient method) is implemented in the TFQ package in the background and cannot be viewed more precisely. This remains a secret from Google.

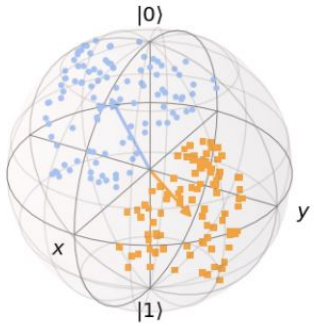

Figure 4. Quantum data in a superposition - Bloch sphere. (Source: Fahri (2018))

5. RESULTS

The original data set contains approximately 6500 different event codes of 46 vehicles over five years with eight differ-

ent failure classes. For the quantum computer experiment, the size of the set is significantly reduced. We do this in two steps. First, we only use system depending fault memory codes. Secondly, the analysis of the conditional probabilities of every diagnosis code led to a choice of the 16 most important codes. The number 16 is a power of two, which fits to the architecture of the quantum computing technology. Further, as seen at the end of section 4, the number of qubits is a limiting factor. In many time periods there are no occurrences of diagnostic codes, so that a lot of zero vectors are filtered out. Furthermore, we only consider nine vehicles within six IGBT explosion for the selected time period. The number of state vectors decreases to 253 samples. It should also be noted that a high occurrence frequency of a certain event code is not counted multiple times resulting in a binary state space.

Property	Whole data set	Reduced data set
time period	60m	18m
vehicles	24	9
inputs	6.500	16
outputs	8	1

Table 1. Comparison of data sets.

The periods before the failures are marked as critical so that misclassified samples can be included in the critical set. Labeling each vector is very important regardless of the method and this misassignment of individual vectors cannot be avoided. This is the reason why we do not only use 'Recall' and 'Precision' for the analysis. In addition, we use a kpi called 'Detection Rate', which is comprehensible for the operation.

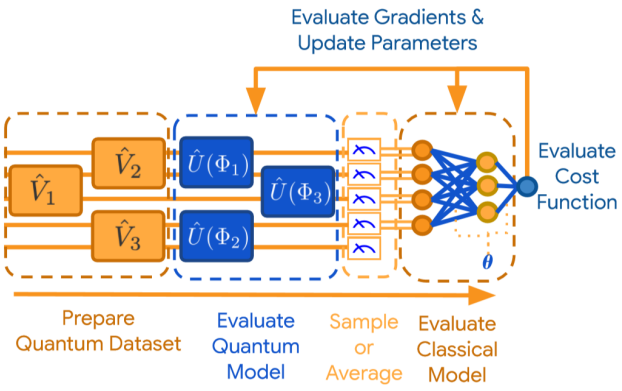

Figure 5. TFQ and CIRQ architecture. Optimization using QAOA. (Source: Fahri (2018))

A multitude of programming languages, architectures and optimization approaches exists in the field of quantum computers. We chose the TensorFlowQuantum/ Cirq language from Google, which work on a gate based QPU as the computing core. The underlying calibration procedure is a QAOA illustrated in Figure 5.

In recent years, the QAOA framework has always been expanded to include additional problem classes. The aim of these is to find a low-energy state, which is represented by Ising Hamiltonians.

In general, it is important to find a local (optimally global) minima of a pseudo-Boolean function $l(z)$ with $z \in \{0,1\}^n$ on n bits, which corresponds to the solution of the QAOA for binary variables.

There are therefore two problems, namely a) training a parameterized quantum circuit for a discrete optimization problem and b) minimizing the cost function of the parameterized quantum circuit. The TFQ should include a) the conversion of simple circuits into TFQ tensors, b) the evaluation of gradients for quantum circuits and c) the use of gradient-based optimizers from TF Adam as basic functions.

In order to be able to calculate on the quantum computer correctly, we have created a small but representative data set as described above. To verify our calculation, the classic neural network (NN) competes against the hybrid calibrated quantum neural network (QNN). Our focus was to ensure that as many properties as possible are similar for both approaches. This can be seen in the following table:

Property	NN	QNN
epochs	30	30
loss	hinge	hinge
optimizer	adam	adam
layer shape	16–64-32-16–1 (relu)	16 – U(θ)–1
# parameter	3713	32
# cross validation	10	10
# training samples	90%	90%
# test samples	10%	10%

Table 2. Overview of neural network parameter settings.

The key performance indicators are presented in the tables 3 and 4. The hinge loss function of the QNN in case of the validation set is comparable with the result of the classical approach. The hinge loss function of the QNN in case of the training set is significantly higher as the results of the classical neural network. For both approaches, the standard deviation is small. Note that the loss function is hard to interpret, so that we analyze our results on derived kpis, which are in the statistics well known as 'Recall' and 'Precision'.

NN	loss	standard deviation
training	0.3746	0.0767
validation	0.7972	0.0784

Table 3. Overview of evaluation results for NN.

QNN	loss	standard deviation
training	0.9799	0.0949
validation	0.7305	0.0434

Table 4. Overview of evaluation results for quantum-based NN.

Figure 6 represents additional key performance indicators. 'RND' is a completely random generated prediction. As expected, both approaches, NN and QNN, deliver better results than random prediction. In this paper, the key performance indicators of the conventional neural networks are better as the indicators of the quantum based neural networks. Nonetheless, it is remarkable that the QNN perform nearly as well as the NN. A reason for the worse prediction of the QNN could be the computing exactness of the current hardware.

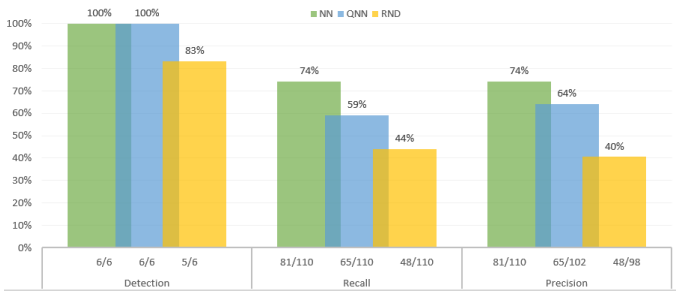

Figure 6. Key performance indicators for NN and QNN.

6. OUTLOOK

As we saw in the previous sections, it is possible to classify data sets on a quantum computer, in particular to calculate a simple version of neural networks. Nevertheless, it will take a while before larger data sets can be processed on a quantum computer. This really depends on the development of the quantum computing hardware.

In section 5 we saw that currently only small amounts of data can be used with binary input and output formats. Here the number of qubits currently available represents the limitation. Google writes that progress in the number of controllable qubits is expected in the next five to ten years and, as a result, larger industry-related problems can be solved on a quantum computer. Thus, we saw in section 5, that the QNN act very similar to conventional NN on small examples. As long as quantum computers can only operate on small data sets, there are two ways to go on with research. One way is to learn and understand the functionality and behavior of quantum computers, identify complex (optimization) problems and solve a small version of these problems on a quantum computer. The other way is to find complex problems, which cannot be solved using a conventional computer, but can be solved on a quantum computer already this time.

Furthermore, we have not yet dealt with the implementation of the calibration step in this paper. As written in the fourth section, we use Google's routines here. It is certainly of interest to understand this step in detail. Within the PlanQK initiative founded by the German government, we will explore further technologies in the area of supervised learning on a quantum computer.

REFERENCES

Broughton, M. e. a. (2020). *Tensorflow quantum: A software framework for quantum machine learning* (Tech. Rep. No. arXiv:2003.02989).

Fahri, H., E. Neven. (2018). *Classification with quantum neural networks on near term processors* (Tech. Rep. No. MIT-CTP/4985). Massachusetts Institute of Technology Cambridge, MA 02139.

Nielsen, M. A. . C. I. L. (Ed.). (2010). *Quantum computation and quantum information*. Cambridge University Press. 10th Anniversary Edition.

Rieffel, W., E. Polak (Ed.). (2011). *Quantum computing: A gentle introduction (scientific and engineering computation)*. CMIT Press. 1st Edition.

Scherer, W. (Ed.). (2016). *Mathematik der quanteninformatik: Eine einführung*. Springer Spektrum. 1st Edition.

Yanofsky, M., N.S. Mannucci (Ed.). (2008). *Quantum computing for computer scientists*. Cambridge University Press.