Hybrid Quantum Machine Learning

Project Report

Samuel Lehmköster (3223499), Dilara Yildiz (3225905)

March 23, 2023

1 Introduction

The airfoil self-noise refers to the collective sound generated by an airfoil in a smooth, non-turbulent inflow. The noise is caused by interactions between the airfoil blade and self-induced turbulence. Several phenomena contribute to the noise, including turbulence in the boundary layer as it passes the trailing edge, flow separation, stall over an airfoil and vortex shedding [1],[2]. While there are semi-empirical methods based on previous theoretical studies for modeling the airfoil self noise production, this work proposes a prediction method based on quantum machine learning.

2 Data Set

We used the Airfoil Self-Noise Data Set by Brooks et. al. with 1503 samples. The NASA data set comprises NACA 0012 airfoils of different sizes at various wind tunnel speeds and angles of attack. The span of the airfoil and the observer position were the same in all of the experiments. Five features were used as input for the prediction of noise:

- Frequency, in Hertzs.
- Angle of attack, in degrees.
- Chord length, in meters.
- Free-stream velocity, in meters per second.
- Suction side displacement thickness, in meters.

Noise is expressed as a scaled sound pressure level, in decibels.

3 Methods

The goal of this work is to predict the airfoil self noise level based on the five given features (section 2) with a quantum machine learning approach. For this purpose, in a first step, a classical neural network was developed, serving as a baseline model for all further network architectures. In addition, a classical autoencoder and Principal Component Analysis (PCA) were applied to the data set to minimize the number of features input to the neural network. Then quantum equivalents were built, including several Hybrid Quantum Neural Networks (hQNNs) in layers and a quantum autoencoder. All models were assessed in terms of their performance and accuracy. Due to lack of access to real quantum computers, all quantum computations were simulated on classical computers leading to exceedingly high computation times. The following network architectures and tools were investigated:

Classical Neural Network Deep neural network consisting of three layers, Rectified Linear Unit (ReLU) activation and RMSprop optimization.

PCA The feature space is reduced from 5 to the first three principle components.

Classical Autoencoder Reduction of feature space to 3 features in latent space of autoencoder.

hQNN Basic Entangling A hQNN, consisting of two classical Dense layers and one quantum layer with CNOT entangling between all 5 qubits.

hQNN Toffoli Entangling Like previous model, but with CCNOT gates instead of CNOT gates.

hQNN Strong Entangling Like the previous model, but including various types of entangling (CNOT, CCNOT, CRY, RY)

Quantum Autoencoder Pure strongly entangled quantum autoencoder with a reduction from 5 to 3 features in the latent space.

Encoder Test Testing a neural network (NN) with the encoded data of the PCA, of the classical autoencoder and the encoders of the first and the second quantum encoders in order to compare the quality of compression in the latent space.

3.1 Classical Neural Network

The classical neural network is a simple NN consisting out of three layers containing seven, fifteen and one neuron as illustrated in figure 1. It uses the ReLU and the linear activation function. Because such small classical NNs have only a small computational demand, no early stopping callbacks were applied.

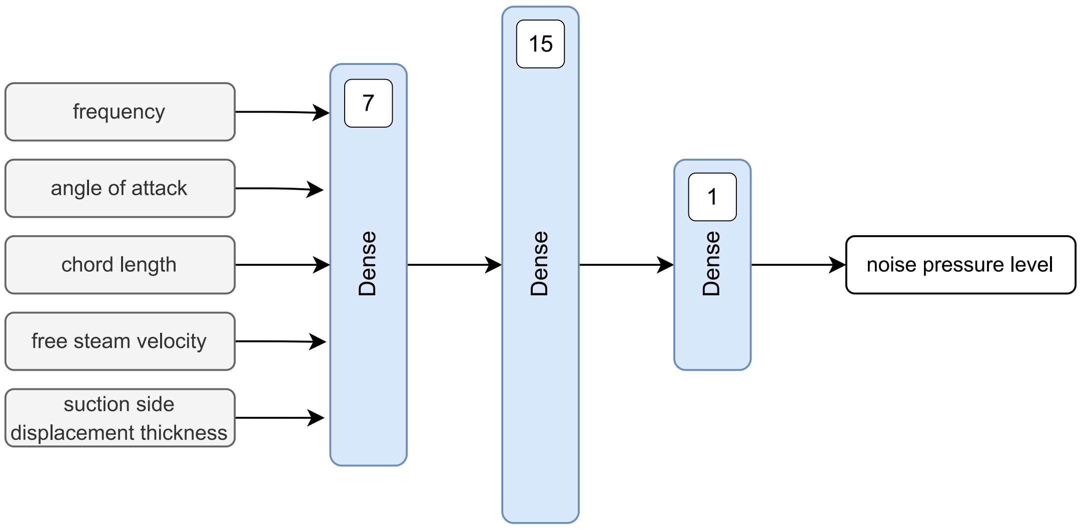


Figure 1: Classical neural network architecture

3.2 Hybrid Quantum Neural Networks (hQNNs)

The hQNNs are implemented in a embedded manner, meaning that the quantum layer is surrounded by classical machine learning layers (fig. 2). This is a hybrid of classical and quantum machine learning. Although the quantum layer varies in it's construction, the number of neurons/qubits in every layer stay the same over the hQNN. The quantum circuit which is here called a quantum node contains a circuit with five qubits which are entangled. The first entangling parameters are the inputs coming from the left-side neurons of a classical layer. Then further parameters act as weights of a classical machine learning layer. Here they lead to stronger entanglement between the qubits and can be tuned in the learning process. At the end of the quantum node qubits are measured in basis of a Pauli-Z gate. These values are passed on to the next (classical) layer.

3.2.1 Basic Entangling

For the first version of the hQNN only a simple entangling is considered, taken from the quantum machine learning example from the pennylane documentation [3]. The

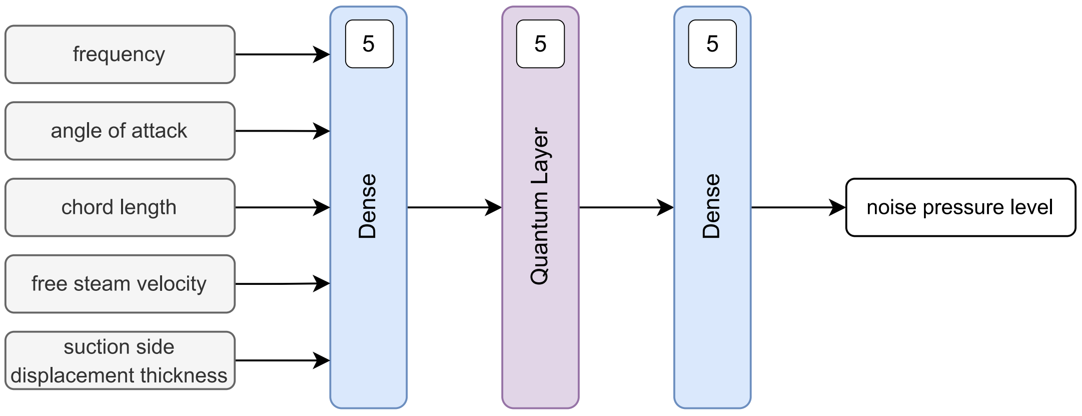


Figure 2: hQNN architecture

input here is described as a rotation around the X-axis of the Bloch sphere of a qubit. The weights are further x-rotations of controlled X-gates. These gates are applied in a ring-shaped manner meaning that each qubit is connected with every neighboring qubit as shown in figure 3.

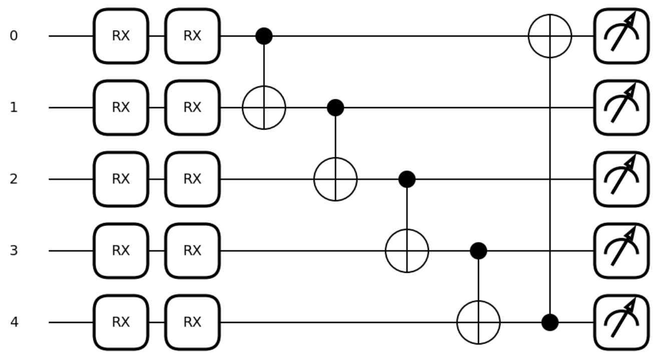


Figure 3: Basic Entangling quantum layer

3.2.2 Toffoli Entangling

In the second iteration of the quantum neural network (QNN), the CNOT-gates utilized in the previous version have been substituted with Toffoli gates (CCNOT). Figure 4 illustrates the entangled quantum layer, which reflects the utilization of Toffoli gates in a ring-wise fashion. Notably, the input mechanism remains the same through the use of angular embedding, where the input is transformed into rotations around the X-axis.

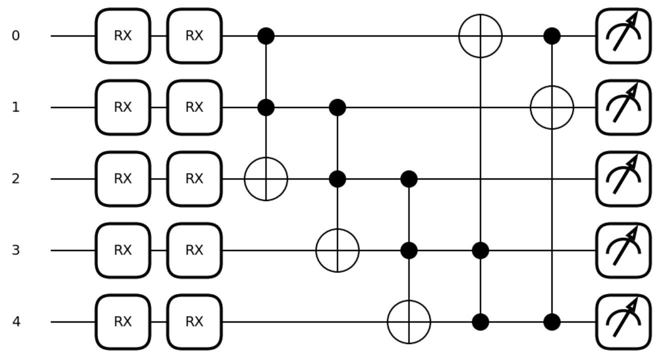


Figure 4: Toffoli Entangling quantum layer

3.2.3 Strong Entangling

Following the implementation of two basic entanglements, the subsequent approach involved maximizing the entanglement between the qubits in order to capture more intricate relationships that are not readily apparent in classical machine learning. To achieve this, a diverse set of gates, including CNOT gates, Toffoli gates, and controlled rotations, were applied in a ring-wise manner. The exact entanglements employed in the quantum neural network can be found in figure 5.

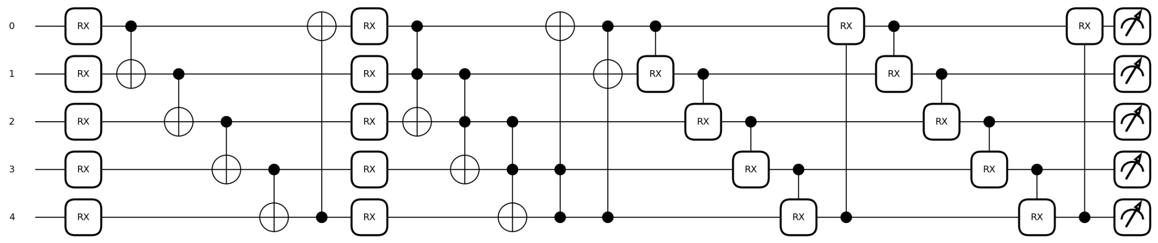


Figure 5: Strong Entangling quantum layer

3.3 Feature Reduction

Feature reduction can help to create smaller, more efficient models, without losing important information. Especially highly correlated features can be reduced by a simple Principal Component Analysis (PCA). However, this method cannot depict nonlinear relations in the data set. Autoencoders (AEs) solve this issue, but involve high training costs. Figure 6 shows the correlation matrix of the original features. Since all features are correlated with each other, a feature reduction is reasonable. In the following steps we reduced the five features to only three with four different approaches: PCA, classical AE and two different quantum AEs.

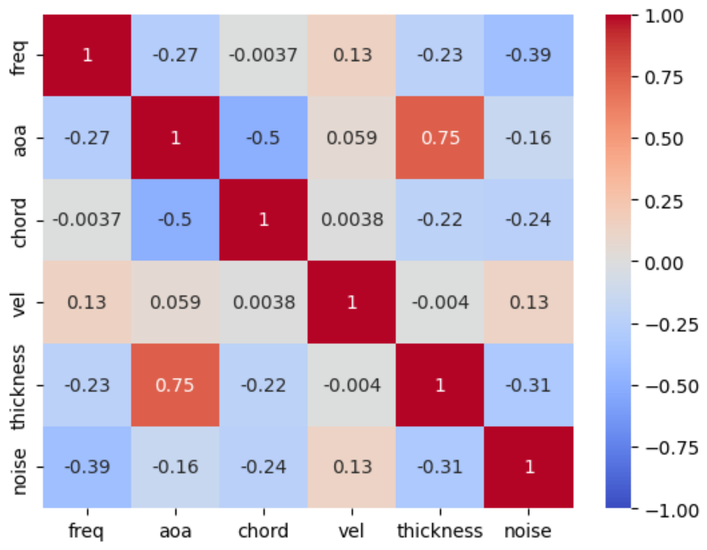


Figure 6: Correlation

3.3.1 Principal Component Analysis (PCA)

The PCA is a statistical method for finding features (main variance axis) in the data. It also serves as a comparison for the compressed features which will be generated by the encoders of the autoencoders in this project.

3.3.2 Classical Autoencoder

The classical autoencoder here is a slim version of an autoencoder. It compresses the five input features to three compressed features. The autoencoder consists only of 38 trainable parameters in two layers of three and five neurons. It is a comparison model for the quantum autoencoder later in the project.

3.3.3 Quantum Autoencoder

For the quantum autoencoder, different approaches were tested. It is important to note that these are only quantum layers and no classical machine learning layers (apart from the input layer) are included. The first setup was inspired by the literature [5] and is a very intuitive approach. It was implemented as shown in fig. 18. It consists out of five neurons from which only three are measured (latent space) in basis of Pauli-Z gates. The quantum layers contain circuits composed out of y-rotations, controlled rotation in y-direction which connect the states of the qubits to each other. The encoder and the decoder have a similar set-up. For the second quantum autoencoder version a structure

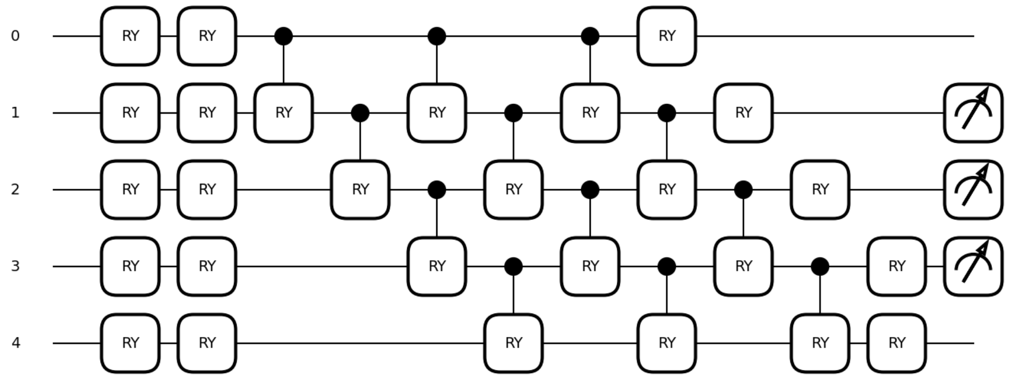


Figure 7: Encoder of the first quantum autoencoder

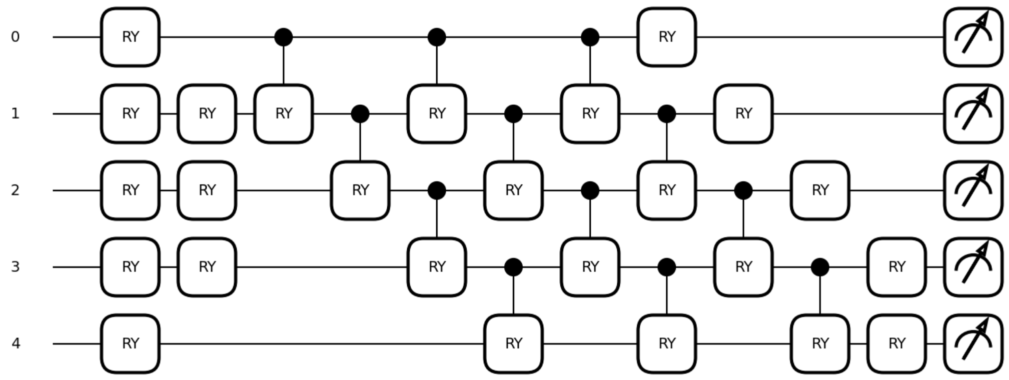


Figure 8: Encoder of the first quantum autoencoder

similar to the quiskit documentation was realized [4]. In this approach the encoder is trained separately from the decoder. Later encoder and decoder are combined together and get trained again on a small part of the actual training set. The encoder is set up with a reference space and a auxiliary qubit. This is in order to execute a SWAP-test on the trash space with the reference qubits as a reference. The idea is to force the trash space to be similar to the reference space (0), so that all information is contained in the latent space qubits.. Therefore this encoder and so the encoding is completely independent from decoding. The decoder following is here similar to the decoder in the first quantum autoencoder.

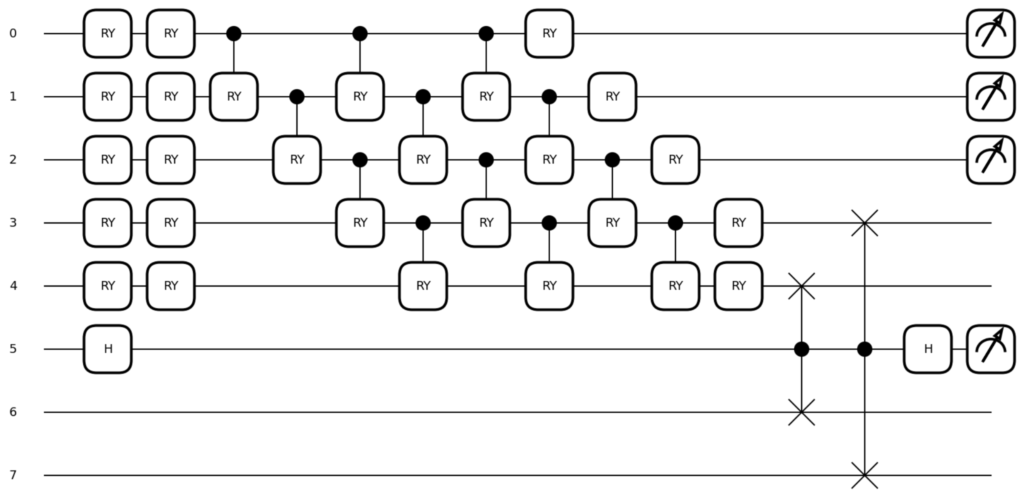


Figure 9: Encoder of the second quantum autoencoder

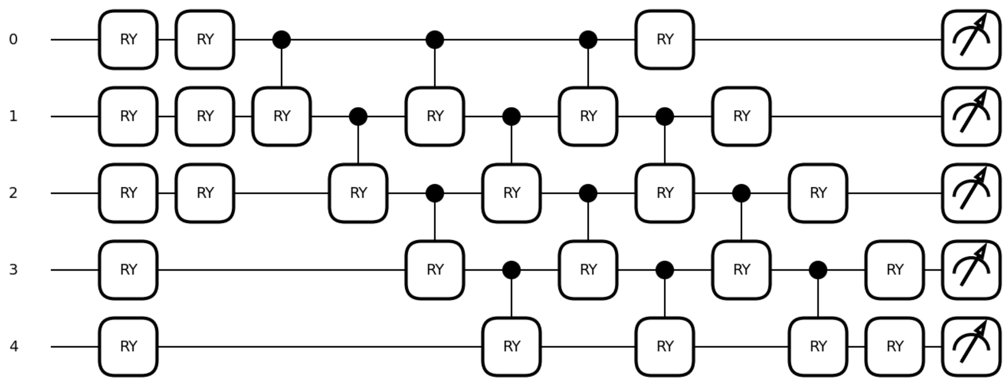


Figure 10: Encoder of the second quantum autoencoder

4 Results

4.1 Classical Neural Network

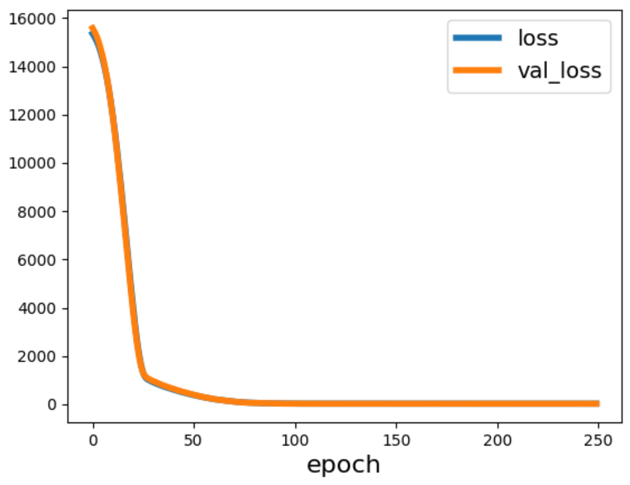


Figure 11: The training history of the classical neural network

The classical neural network has a strong convergence to the minimum plateau in the training history, indicating that it learns easily for this data set. It fully converged after around 70 epochs. The computational time for this learning process is in the range of under a minutes for a personal computer. Since there was no early stopping callback, the NN trained through all 250 epochs, even after it was long converged to its minimum.

4.2 Principal Component Analysis (PCA)

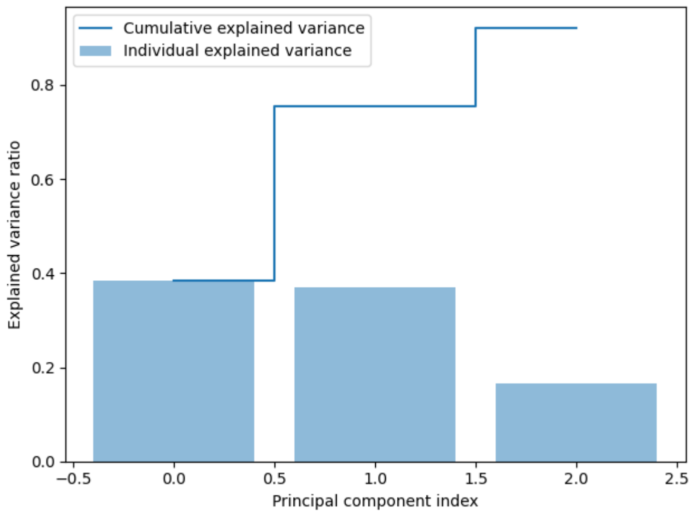


Figure 12: The first 3 principal components of the training set

The PCA shows that within the first three principal components over 80% variance in the data set is covered (fig.12). This is an indicator that three compressed features are a good latent space dimension for an autoencoder to reach in this case. The first three principal components will be used as a comparison case later.

4.3 Classical Autoencoder

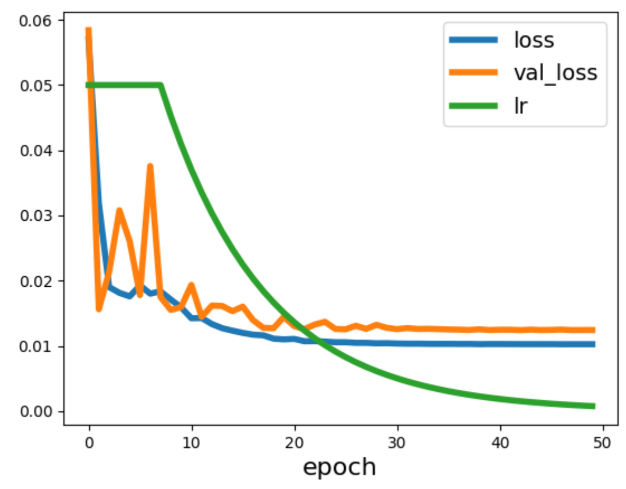


Figure 13: Training history of the classical autoencoder

The classical autoencoder yields a good compression to three parameters with reaching up to a error of around 0.01. The training history (fig. 13) shows some oscillation while converging to a plateau of the minimum which is approx. reached after 30 epochs already. The oscillation might be explained with the high convergence rate at the beginning of the training which starts to decrease exponentially after 8 epochs.

4.4 Quantum Neural Network (hQNN)

4.4.1 Basic Entangling

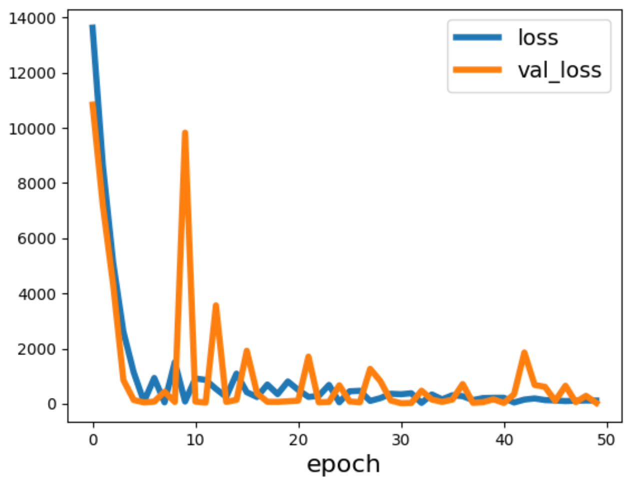


Figure 14: Training history of first quantum neural network

The first hQNN yields already good results. It converged after 5 epochs to the minimum plateau. The oscillations decreased until epoch 15, but still remain at a high level in the loss history. The stronger oscillations might indicate a more sensitive model to changes of the weights. This is in contrast to the more sophisticated models following.

4.4.2 Toffoli Entangling

The second hQNN shows a different loss history over the training. Although the magnitude of the final minimal loss is in the same order compared to the simpler hQNN (4.4.1), it does not show a transient oscillation while converging to the minimal loss plateau. In comparison to the classical this hQNN reaches the minimum plateau after 4 epochs, which astonishingly fast even compared to the classical NN. It shows nearly no deviation in the further training. This might indicate that with this quantum circuit the hQNN finds a lot of solutions for complete range of weights where the loss remains in a very close area. Meaning that the loss landscape might have a vast minimal plateau that is very flat.

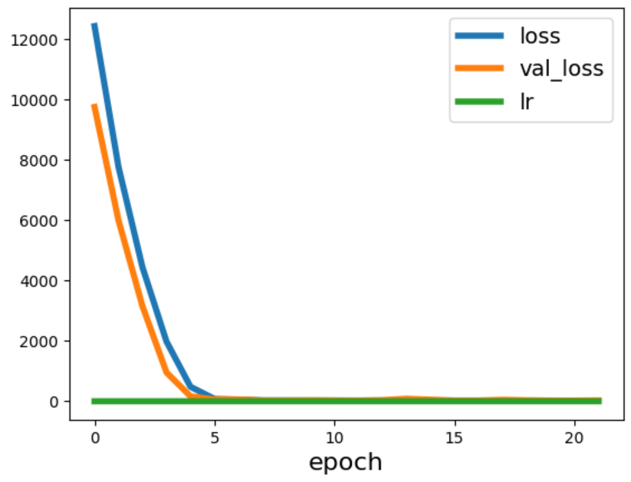


Figure 15: Training history of the quantum layer with Toffoli entangling

4.4.3 Strong Entangling

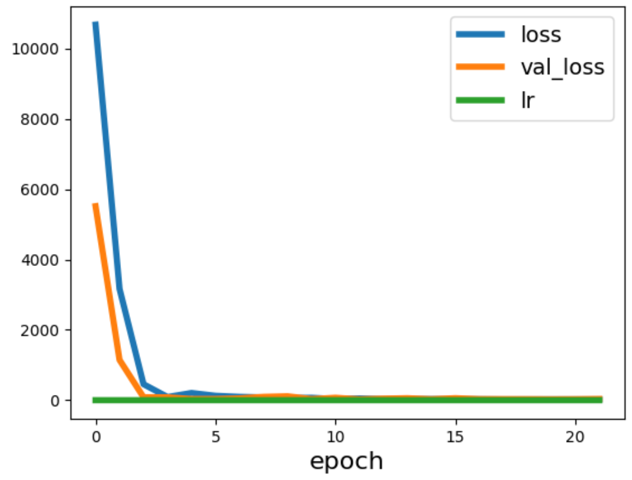


Figure 16: Training history of the strongly entangled quantum layer

The hQNN with strong entanglement (fig.16) experienced a similar issue. It is hard do differentiate to the Toffoli entangling above. It seems that although it contains more weights and entanglements, it does not find a better solution and yields in the end a very vast and flat minimal plateau in the loss landscape.

4.5 Quantum Autoencoder

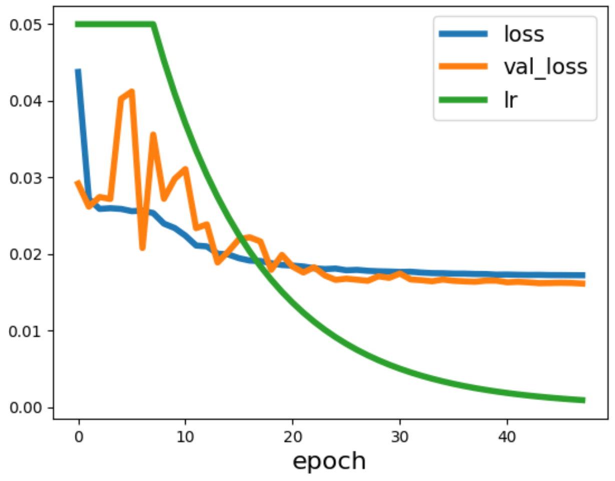
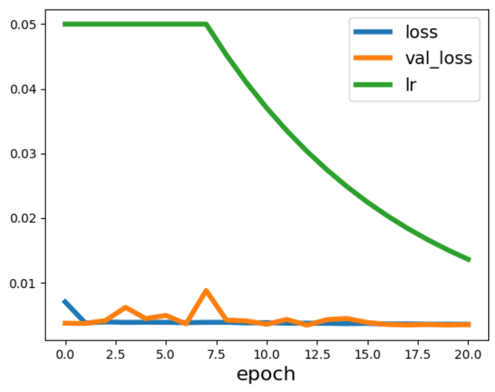


Figure 17: Training history of first quantum autoencoder

The loss of the first quantum autoencoder is around 0.01 higher after converging compared to the classical autoencoder(see fig. 13). Additionally the emulated quantum autoencoder consumes far more computational time than the slim classical autoencoder, several 100 times more.

The second quantum autoencoder is even more off with around 0.04 compared to the classical autoencoder(see fig. 13). It also consumed a lot of computational time, since there were three training's to run. But although the decoder training has only found

(a) (b)



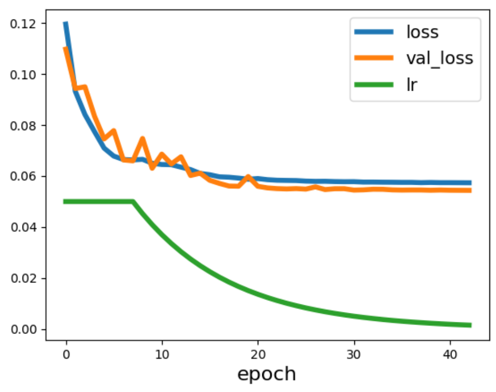


Figure 18: Training of the second quantum autoencoder, separately for the encoder (a) and the decoder (b)

a higher minimal loss (fig.18b) compared to the classical autoencoder, the run of the encoder which was the main difference to the first quantum autoencoder seems promising good (fig.18a). But it has to be mentioned that here the minimizing task is a different one, since this quantum encoder tries only to set the trash space to zero so the complete representation of the data is stored in the latent space.

4.6 Encoder Test

As a final test, all encoders of the autoencoders were loaded and create their compression of the same training data to from five to three features. Further, the PCA was applied to this data to find also three principal components. All of these inputs were then used to train a similar, very simple, classical NN. Using three neurons to process the compressed data and on further layer with only one neuron for the prediction of the noise pressure level. The results of the training are compared in fig. 20. One can see that in this case, the classical autoencoder scores best when it comes to convergence rates. Nevertheless, it seems that the final plateau near zero is reached first by the second quantum encoder version.

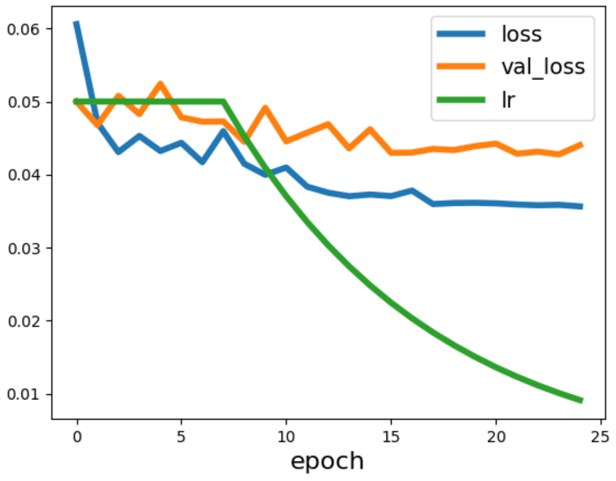


Figure 19: Training history of second quantum autoencoder after concatenating the encoder and the decoder

5 Conclusion

In the end it can was found that quantum neural networks do not generally perform better compared to the classical neural networks. But is has to be remarked that they score way worse, reminding the emulation of the quantum computer consumes a lot of computational time and makes it therefore a inefficient way of solving such problems. Another problem could be the inaccuracies of real quantum computers as they exist today which are not take into account in this project, since a try with a noisy quantum layer had to be interrupted due to it's extreme high computational effort. The advantages of the quantum computer seems to arise for more pure quantum circuit as it was constructed for the quantum autoencoders. There the quantum encoder seems to have a slim advantage to a classical autoencoder, although the computational time is not comparable. Especially the ansatz with quantum encoder where the trash space of the encoder is forced to be the same as an auxiliary constant space seems promising.

References

[1] Thomas F Brooks, D Stuart Pope, and Michael A Marcolini. *Airfoil self-noise and prediction.* Tech. rep. 1989.

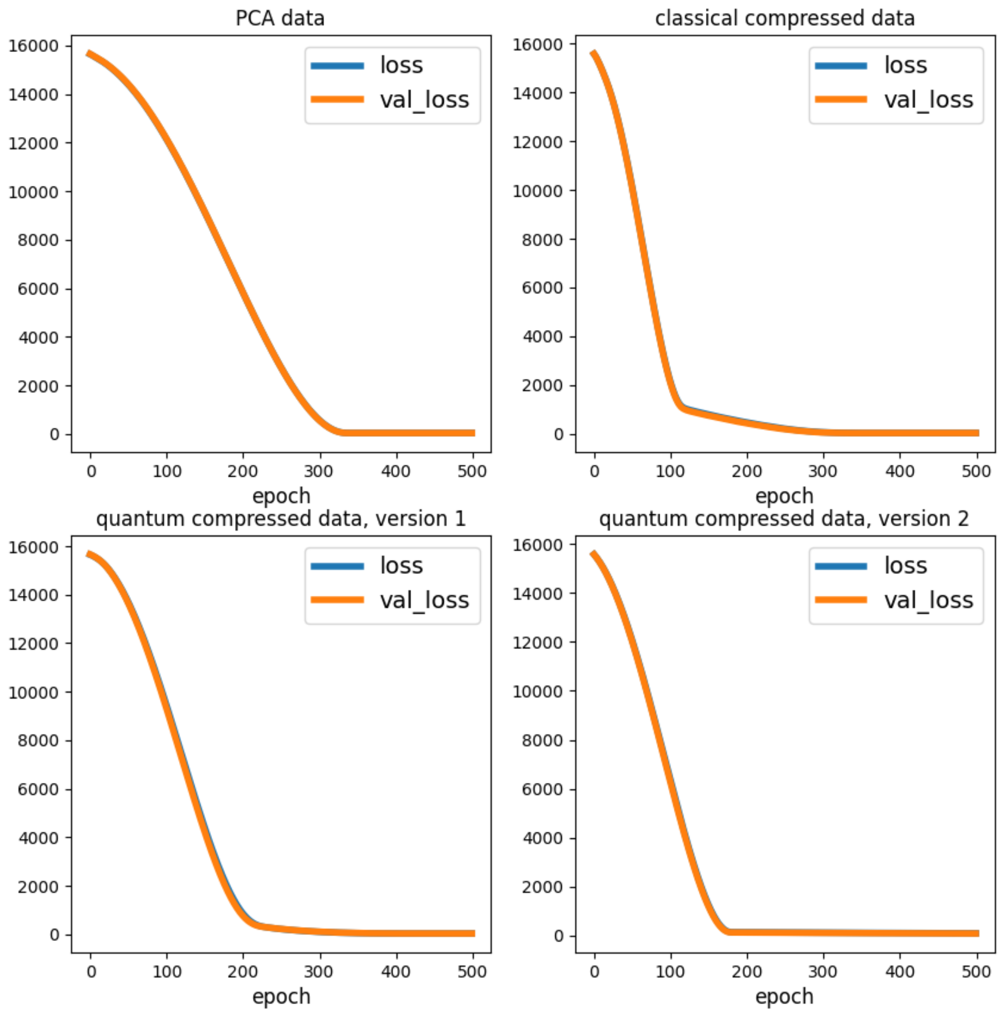


Figure 20: Comparison of feature reduction methods: Training histories of classical neural network with different inputs (PCA reduced data, classical compressed data, quantum compressed data)

[2] Con Doolan and Danielle Moreau. “Airfoil Noise Mechanisms and Control”. In: *Flow Noise: Theory*. Singapore: Springer Nature Singapore, 2022, pp. 139–171. ISBN: 978-981-19-2484-2. DOI: 10.1007/978-981-19-2484-2_8. URL: https://doi.org/10.1007/978-981-19-2484-2_8.

[3] Pennylane Tutorial. *Turning quantum nodes into Keras Layers*. URL: https://pennylane.ai/qml/demos/tutorial_qnn_module_tf.html.

[4] Quiskit Tutorial. *The Quantum Autoencoder*. URL: https://qiskit.org/documentation/machine-learning/tutorials/12_quantum_autoencoder.html#The-SWAP-Test.

[5] Jonathan Romero, Jonathan P Olson, and Alan Aspuru-Guzik. “Quantum autoencoders for efficient compression of quantum data”. In: *Quantum Science and Technology* 2.4 (Aug. 2017), p. 045001. DOI: 10.1088/2058-9565/aa8072. URL: https://doi.org/10.1088%2F2058-9565%2Faa8072.