Dummy Titlepage

Boye Gravningen Sjo

Autumn 2022

TMA4500

Industrial Mathematics Specialisation Project

Department of Mathematical Sciences

Faculty of Information Technology and Electrical Engineering

Norwegian University of Science and Technology

Contents

Chapter 1

Introduction

1.1 Relevant work

1.1.1 Cerezo et al. (2021)

Variation quantum algorithms (VQAs) are envisioned as the most likely candidate for quantum advantage to be achieved. By optimising a set of parameters that describe the quantum circuit, classical optimisation techniques are applicable, and only using the quantum hardware for what can be interpreted as function calls, limits the circuit depths needed. Running the same circuit many times with slightly different parameters and inputs in a classical-quantum-hybrid fashion, rather than a complete quantum implementation, means that the quantum operations can be simple enough for the noise and decoherence to be manageable.

Generally, VQAs start with defining a cost function, depending on some input data (states) and the parametrised circuit, to be minimised with respect to the parameters of the quantum circuit. For example, the cost function for the variational quantum eigensolver (VQE) is the expectation value of some Hamiltonian, which is the energy of a system. The cost function should be meaningful in the sense that the minimum coincides with the optimal solution to the problem, and that lower values generally implies better solutions. Additionally, the cost function should be complicated enough to warrant quantum computation by not being easily calculated on classical hardware, while still having few enough parameters to be efficiently optimised.

The optimisation of the cost function is often done with gradient descent methods. To evaluate the gradient of the quantum circuit w.r.t. the parameters, the very convenient parameter shift rule is often used. Though appearing almost as a finite difference scheme, relying on evaluating the circuit with slightly shifted parameters, it is indeed and exact formula. Furthermore, it may be used recursively to evaluate higher order derivatives, which is useful for optimisation methods that require the Hessian.

VQA's applications are numerous. The archetypical example is finding the ground state of a Hamiltonian for a molecule. Such problems are exponential in the particle count, and thus intractable on classical hardware for larger molecules, while the problem of evaluating the Hamiltonian on quantum hardware is typically polynomial. VQAs are also well suited for general mathematical problems and optimisation, even machine learning, another common example being QAOA for the max-cut problem.

Still, there are many difficulties when applying VQAs. Barren plateaus are a common occurrence, making the optimisation futile. The choosing of the ansatz determines the performance and feasibility of the algorithms, and there are many strategies and options. Some rely on ex-

ploiting the specific quantum hardware's properties, while some use the specifics of the problem at hand. Finally, the inherent noise and errors on near-term hardware will still be a problem and limit circuit depths.

1.1.2 Moll et al. (2018)

The computational performance of quantum computers is decided by five main factors. Naturally, the total qubit count is important, but also their connectivity (if they are not connected, intermediate operations like swapping is needed). How many gates/operations can be used before decoherence, noise and errors ruins the result also determines what programmes are feasible. Furthermore, which physical gates are available also matters, as transpiling to native gates will increase the circuit depth. Lastly, the degree of gate parallelisation can allow for shallower circuits and increased performance.

With all these factors in mind, the metric of quantum volume is defined, giving a single number describing the performance. It is effectively defined as the largest rectangular circuit of two-qubits a quantum computer may execute.

1.1.3 Torlai et al. (2020)

Due to the probabilistic nature of quantum computers and their exponentially great number of states, measuring complex observables accurately requires many samples. By post-processing the measurements using an artificial neural network, the variance of the samples are significantly reduced, though at the cost of some increased bias.

1.1.4 Schuld et al. (2019)

In optimising the parameters of variational circuits, having access to the gradient of the cost function (with respect to the parameters) is beneficial. The individual measurements are probabilistic, but the expectation is a deterministic value whose gradient can be calculated. Often, this is possible exactly using the parameter shift rule, allowing for evaluating the gradient using the same circuit with changed parameters. For circuits containing gates whose derivatives are not as nice, a method of linear combination of unities can be used. This method requires an extended circuit including an ancillary qubit.

1.1.5 Pesah et al. (2021)

The problem of barren plateaus plagues the optimisation of variational circuits and quantum neural network; for randomly initialised ansätze, the gradient of the cost function may exhibit exponentially small gradients, prohibiting gradient based optimisation. Under certain assumptions, it is shown that for quantum convolutional neural networks, the gradient of the cost function is no worse than polynomially small, such that the networks can be trainable.

1.1.6 Farhi et al. (2018)

Quantum neural networks (QNNs) are simply an abstraction of parametrised quantum circuits with some sort of data encoding. As with classical neural networks or supervised learning in general, the parameters are optimised by minimising a cost function. For QNNs, the output can be a single designated read-out qubit, where the states are interpreted as classes in a binary classification problem. This was shown to indeed be feasible for handwritten digit recognition, using downsampled MNIST data. With the qubit count on current quantum devices and the

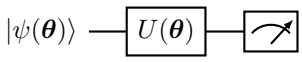

Figure 1.1: Test figure.

amount that can be easily simulated, the dimensionality of the data can not be much more than a dozen.

1.1.7 Abbas et al. (2021)

Whether quantum neural networks have inherent advantages is still an open question. Using the Fisher information of models, the authors calculate the effective dimension as a measure of expressibility. For models comparable in their input, output and parameter count, the effective dimension of particular quantum neural networks can be significantly higher. This advantage is empirically shown to be useful with a particular model on real quantum hardware, showing convergence in fewer steps than a similarly specced classical network.

The importance of feature maps is remarked upon, affecting both the expressibility of the model and the risk of barren plateaus, which in turn determines trainability.

1.2 Quantikz

Admire fig. 1.1. It is a quantum circuit. It is drawn using the package `quantikz`.

Chapter 2

Machine learning

2.1 Introduction

2.2 Neural networks

2.2.1 Universal approximation theorem

Chapter 3

Quantum computing

Chapter 4

Quantum machine learning

4.1 Data encoding

4.2 Quantum neural networks

4.2.1 QNN versus NN

Fisher's iris dataset is perhaps the most used dataset for classification in statistics, containing samples of three different species of iris flowers. For each species, there are 50 samples, each with four features: sepal length, sepal width, petal length, and petal width. Like in [1], only the two first species are considered, which happen to be linearly separable in the feature space.

To compare the performance of the QNN and the NN, architectures suited for binary classification parametrised by exactly 8 parameters are used. The QNN structure is shown in fig. 4.1. The four dimensional input data $\boldsymbol{x}$ is mapped to some quantum state $|\psi\boldsymbol{x}\rangle$ using the ZZFeatureMap with two repetitions, which is a second-order Pauli-Z evolution circuit. This feature map is essentially a mix of rotation around the Z axis parametrised by the input features or functions thereof and CNOT and Hadamard gates. The quantum state is then evolved by a the RealAmplitudes ansatz. All four qubits a rotated in the Y direction by some parameter before CNOT-gates are applied pairwise, for a final parametrised rotation in the Y direction, for a total of 8 parameters. For details on the components, see Qiskit's documentation [10] or the original paper [1]. The parity of the output 4-bit string is interpreted as the prediction, and with 100 shots used, the model gave a probability for each of the two iris classes. Both exact simulations and noisy simulations were performed, with the latter using noise modelled after the 27-qubit IBM Montreal architecture, the actual hardware used in the original paper.

The classical neural network was a standard feed-forward model, with a 4-1-1-1-2 layered structure without biases, giving a total of 8 parameters. The activation functions were leaky ReLUs,

$$\text{LeakyReLU}(x) = \begin{cases} x & x \geq 0 \\ 0.01x & x < 0 \end{cases}, \tag{4.1}$$

and the output layer used a softmax activation function.

Both models were implemented using PyTorch, with the QNN being implemented using Qiskit's PyTorch interface. Consequently, the models could be trained in the exact same manner, using the Adam optimiser with a learning rate of 001. The models were trained for 100 epochs, with the loss function being cross-entropy.

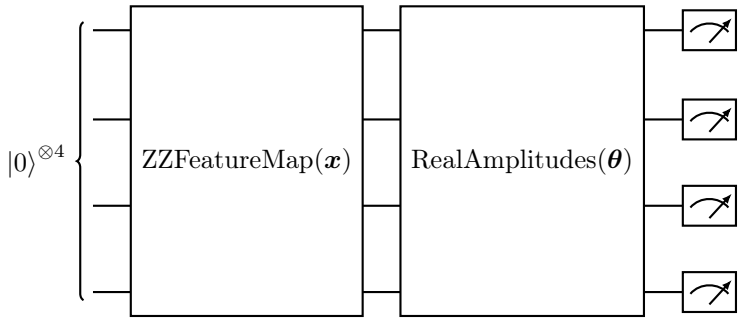

Figure 4.1: Structure of the QNN used for classification of the iris dataset. The first block maps the input data $\boldsymbol{x}$ to the quantum state $|\psi(\boldsymbol{x})\rangle$, before the second block is the variational circuit, parametrised by $\boldsymbol{\theta}$. Finally, all qubits are measured, where the parity is interpreted as the prediction.

For validation, 10-fold cross-validation was used. That is, the dataset was split into 10 equal parts or *folds*. Each fold us used as the validation set once, their accuracies being recorded during the training with the other nine folds. The mean accuracy over the 10 folds was used for the final performance metric, shown in fig. 4.2.

As in the original paper, the QNN converges much quicker and more consistently, with an out-of-fold accuracy of 100% for all ten folds. The NN, on the other hand, requires more iterations to converge and does not always do so. In some cases, the model diverges and only predicts one class, which is why the out-of-fold accuracy is not 100% for all folds.

4.2.2 Quantum convolutional neural networks

Testing a QCNN on synthetic data shown in fig. 4.3; classify pictures as either horizontal or vertical lines. The model without errors achieve 100% accuracy on both training and test data, while the noisy model only managed around 90% accuracy on the training data and 80% on test data. The loss function of the noisy and exact model during training is shown in fig. 4.4.

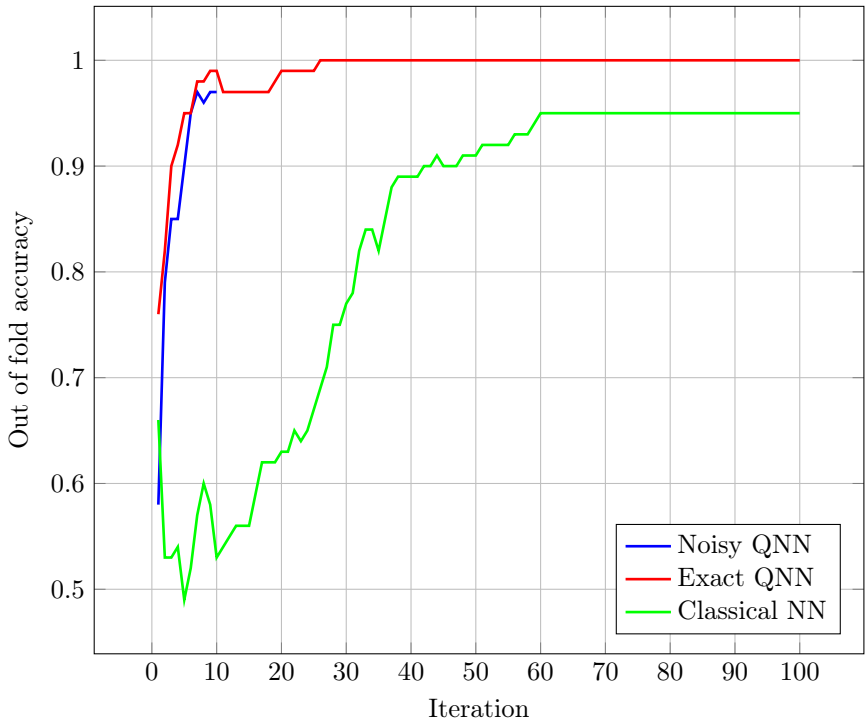

Figure 4.2: Mean accuracy during training for the iris dataset using 10-fold cross validation. All models have 8 parameters and are trained using the Adam optimiser with a learning rate of 0.1, using cross-entropy as the loss function. Due to the computational cost, the noisy (simulated IBM Montreal backend) QNN was only trained for 10 epochs due to the computational cost.

Figure 4.3: Synthetic data for the QCNN.

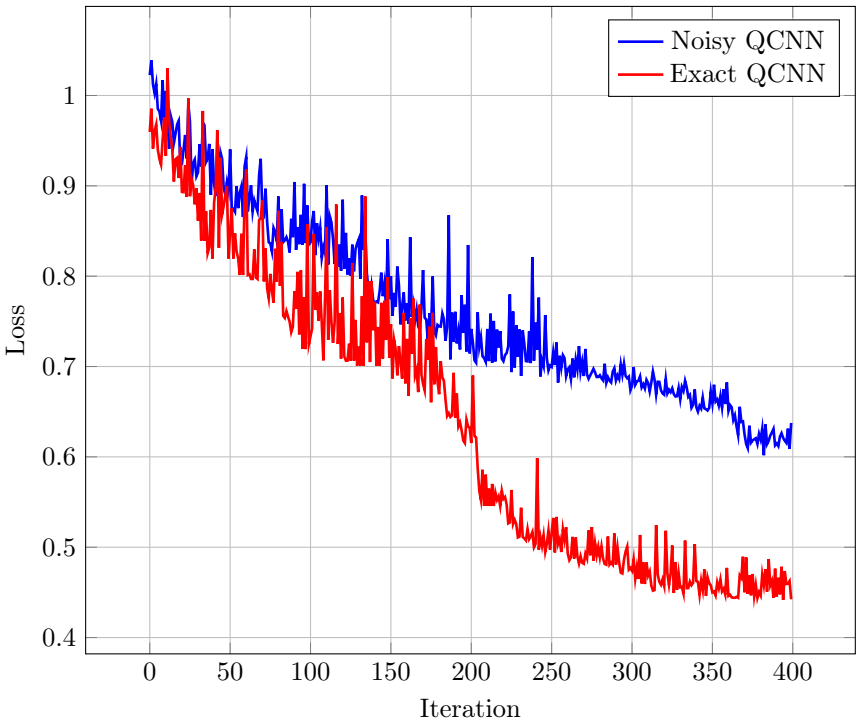

Figure 4.4: Loss function of QCNN model during training with both exact simulations and noisy.

References

[1] Amira Abbas et al. 'The power of quantum neural networks'. In: *Nature Computational Science* 1.6 (June 2021), pp. 403–409. ISSN: 2662-8457. DOI: 10.1038/s43588-021-00084-1. URL: http://www.nature.com/articles/s43588-021-00084-1 (visited on 15/09/2022).

[2] M. Cerezo et al. 'Variational quantum algorithms'. In: *Nature Reviews Physics* 3.9 (Aug. 2021), pp. 625–644. DOI: 10.1038/s42254-021-00348-9. URL: https://doi.org/10.1038%2Fs42254-021-00348-9.

[3] Edward Farhi and Hartmut Neven. *Classification with Quantum Neural Networks on Near Term Processors*. 2018. DOI: 10.48550/ARXIV.1802.06002. URL: https://arxiv.org/abs/1802.06002.

[4] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2014. DOI: 10.48550/ARXIV.1412.6980. URL: https://arxiv.org/abs/1412.6980.

[5] Nikolaj Moll et al. 'Quantum optimization using variational algorithms on near-term quantum devices'. In: *Quantum Science and Technology* 3.3 (June 2018), p. 030503. DOI: 10.1088/2058-9565/aab822. URL: https://doi.org/10.1088%2F2058-9565%2Faab822.

[6] Adam Paszke et al. 'PyTorch: An Imperative Style, High-Performance Deep Learning Library'. In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[7] Arthur Pesah et al. 'Absence of Barren Plateaus in Quantum Convolutional Neural Networks'. In: *Physical Review X* 11.4 (Oct. 2021). DOI: 10.1103/physrevx.11.041011. URL: https://doi.org/10.1103%2Fphysrevx.11.041011.

[8] Maria Schuld et al. 'Evaluating analytic gradients on quantum hardware'. In: *Phys. Rev. A* 99 (3 Mar. 2019), p. 032331. DOI: 10.1103/PhysRevA.99.032331. URL: https://link.aps.org/doi/10.1103/PhysRevA.99.032331.

[9] Giacomo Torlai et al. 'Precise measurement of quantum observables with neural-network estimators'. In: *Phys. Rev. Research* 2 (2 June 2020), p. 022060. DOI: 10.1103/PhysRevResearch.2.022060. URL: https://link.aps.org/doi/10.1103/PhysRevResearch.2.022060.

[10] Matthew Treinish et al. *Qiskit: Qiskit 0.37.2*. Version 0.37.2. Aug. 2022. DOI: 10.5281/zenodo.7017746. URL: https://doi.org/10.5281/zenodo.7017746.