Hybrid classical-quantum learning

www.qu.antum.ml

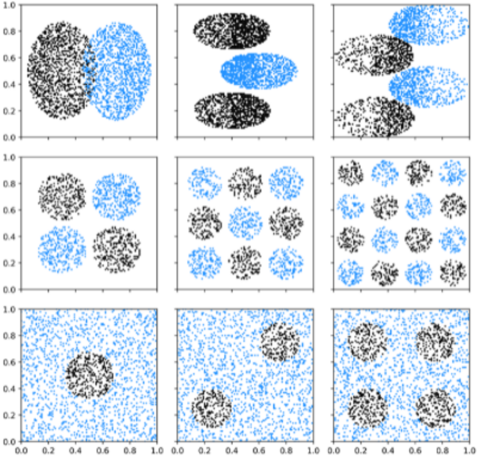
PATRICK STECHER

Technical University Munich
patrick.stecher@tum.de

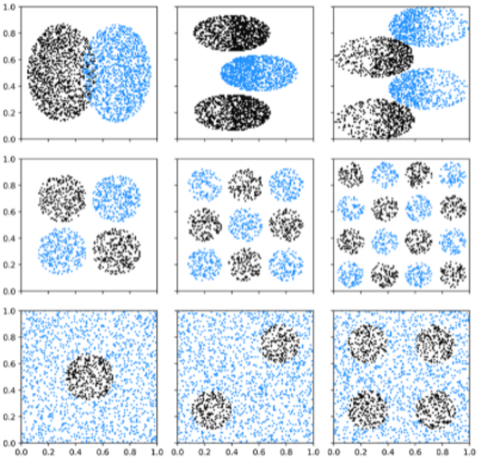
2020-06-18

Section 1

Introduction

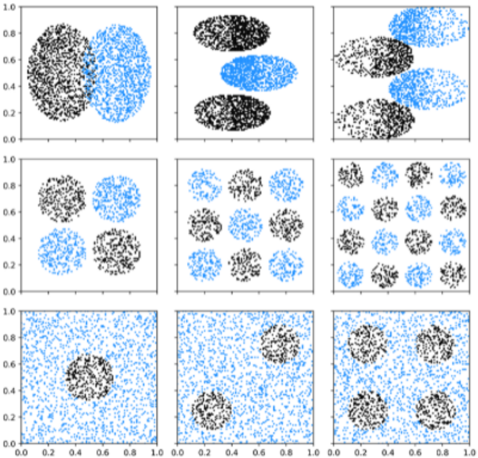


[1] [Hubregtsen et al, 2020]



Goal: Classify this Data set![1]

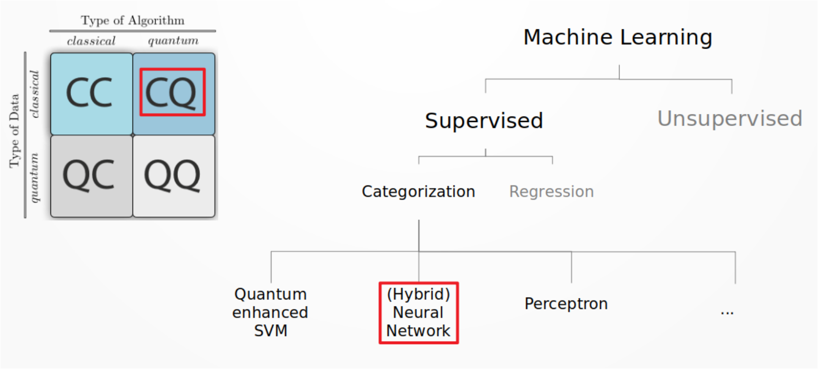
[1] [Hubregtsen et al, 2020]



Goal: Classify this Data set![1] (Using a quantum computer ☺)

[1] [Hubregtsen et al, 2020]

30.000ft overview



Definition(s)

No fixed definition for "Hybrid":

- Using classical data on a quantum circuit (or the other way around)

Definition(s)

No fixed definition for "Hybrid":

- Using classical data on a quantum circuit (or the other way around)
- Down projection of high-dimensional inputs, (e.g. pictures) to low-dimensional representations (e.g. using convolutional NN), suitable for running on a quantum computer typically consisting of a single-digit number of qubits.

Definition(s)

No fixed definition for "Hybrid":

- Using classical data on a quantum circuit (or the other way around)
- Down projection of high-dimensional inputs, (e.g. pictures) to low-dimensional representations (e.g. using convolutional NN), suitable for running on a quantum computer typically consisting of a single-digit number of qubits.
- **Optimization of a Parameterized Quantum Circuit (PQC) using classical optimization techniques such as gradient descent.**

Section 2

Architecture

Architecture

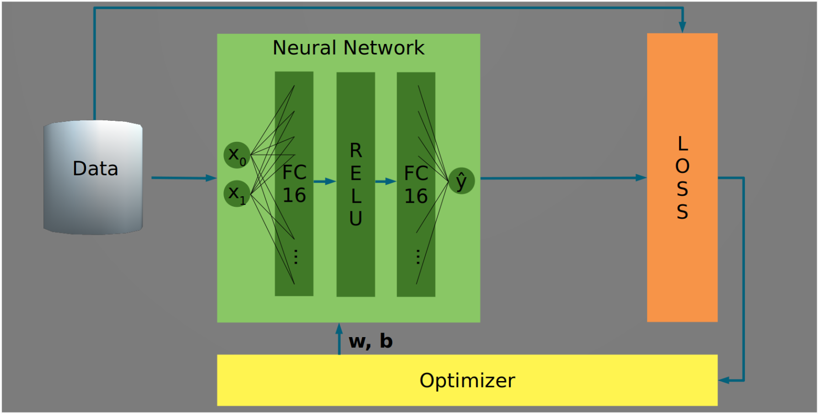


Figure: architecture of a classical network

Architecture

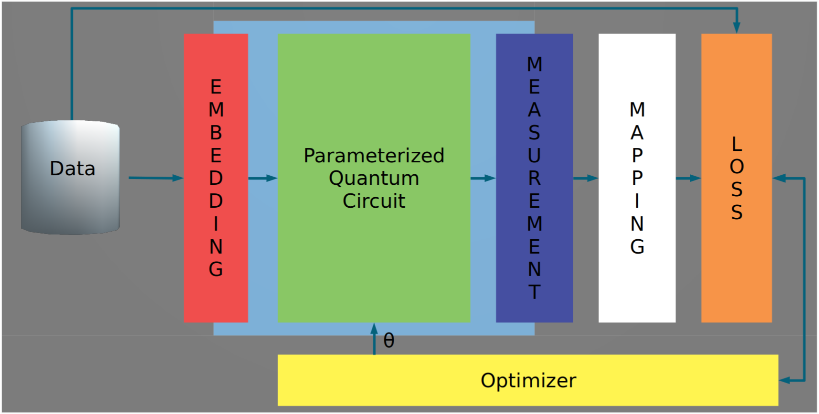
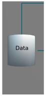


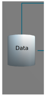
Figure: General architecture of a hybrid-classical network

The Data



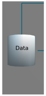
- 9 synthetic data sets

The Data



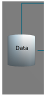
- 9 synthetic data sets
- 1500 samples each

The Data

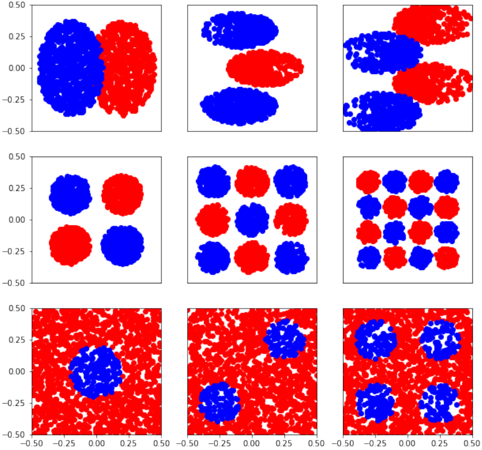


- 9 synthetic data sets
- 1500 samples each
- different "difficulties"

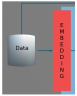
The Data



- 9 synthetic data sets
- 1500 samples each
- different "difficulties"



Embedding

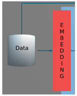


How to get the data into the Quantum circuit?

- A mapping function $f : \mathbf{x} \rightarrow U_{\phi(\mathbf{x})}|0\rangle^{\otimes n}$ (for n qubits) is needed.
- a feature map[2] $\phi(x)$ is needed.

[2] e.g.: $\phi : \mathbf{x} \rightarrow [0, 2\pi)$
[3] [Benedetti et al, 2019]

Embedding



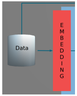
How to get the data into the Quantum circuit?

- A mapping function $f : \mathbf{x} \rightarrow U_{\phi(\mathbf{x})}|0\rangle^{\otimes n}$ (for n qubits) is needed.
- a feature map[2] $\phi(x)$ is needed.

Embedding scheme	Formula	Kernel[3]
Amplitude	$x \in \{0,1\}^n \rightarrow \|x\rangle$	linear kernel
Product	$x \in \mathbb{R}^N \rightarrow \|\psi_x\rangle = \cos(x_j)\|0\rangle + \sin(x_j)\|1\rangle$	highly non-linear kernels

[2] e.g.: $\phi : \mathbf{x} \rightarrow [0, 2\pi)$
[3] [Benedetti et al, 2019]

Embedding Circuit



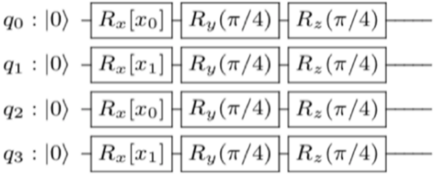
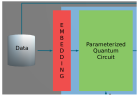


Figure: Embedding circuit

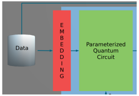
Parameterized Quantum Circuit



- The basic idea of quantum computing is similar to that of kernel methods in machine learning, to efficiently perform computations in an intractably large Hilbert/Kernel space.

[4] [Sim et al. 2019]

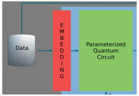
Parameterized Quantum Circuit



- The basic idea of quantum computing is similar to that of kernel methods in machine learning, to efficiently perform computations in an intractably large Hilbert/Kernel space.
- Parameterized Quantum Circuits are a combination of multiple quantum gates operating on one or multiple qubits. The quantum gates have free parameters which can be optimized to fit a desired probability distribution.

[4] [Sim et al. 2019]

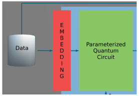
Parameterized Quantum Circuit



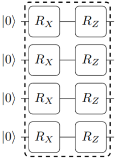
- The basic idea of quantum computing is similar to that of kernel methods in machine learning, to efficiently perform computations in an intractably large Hilbert/Kernel space.
- Parameterized Quantum Circuits are a combination of multiple quantum gates operating on one or multiple qubits. The quantum gates have free parameters which can be optimized to fit a desired probability distribution.
- The circuits presented here and investigated in *Hubregtsen et al*. were originally proposed by *Sim et al.*[4]

[4] [Sim et al. 2019]

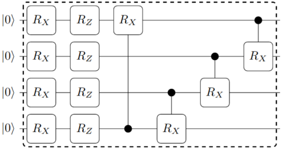
Parameterized Quantum Circuit



- The basic idea of quantum computing is similar to that of kernel methods in machine learning, to efficiently perform computations in an intractably large Hilbert/Kernel space.
- Parameterized Quantum Circuits are a combination of multiple quantum gates operating on one or multiple qubits. The quantum gates have free parameters which can be optimized to fit a desired probability distribution.
- The circuits presented here and investigated in *Hubregtsen et al.* were originally proposed by *Sim et al.*[4]



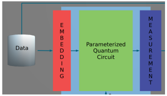
Circuit 1



Circuit 19

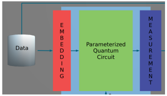
[4] [Sim et al. 2019]

Measurement



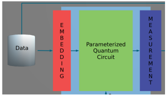
- The four qubits are then measured in a fixed but arbitrary basis. Arbitrary since you can always rotate into a different basis, in this case this rotation is just learned.

Measurement



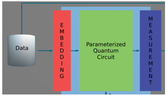
- The four qubits are then measured in a fixed but arbitrary basis. Arbitrary since you can always rotate into a different basis, in this case this rotation is just learned.
- The process of preparing quantum states, embedding data, applying rotations according to the parameter vector theta and measuring the outcome are performed multiple times.

Measurement

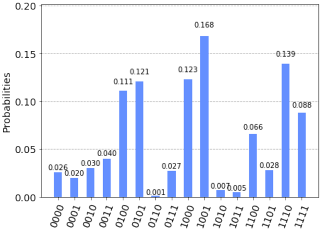


- The four qubits are then measured in a fixed but arbitrary basis. Arbitrary since you can always rotate into a different basis, in this case this rotation is just learned.
- The process of preparing quantum states, embedding data, applying rotations according to the parameter vector theta and measuring the outcome are performed multiple times.
- This is referred to as "shots" and will lead to a probability distribution over the possible states.

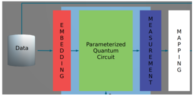
Measurement



- The four qubits are then measured in a fixed but arbitrary basis. Arbitrary since you can always rotate into a different basis, in this case this rotation is just learned.
- The process of preparing quantum states, embedding data, applying rotations according to the parameter vector theta and measuring the outcome are performed multiple times.
- This is referred to as "shots" and will lead to a probability distribution over the possible states.

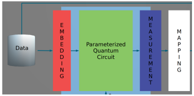


Mapping function



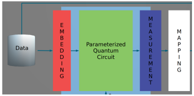
- How to get class labels from a distribution?

Mapping function



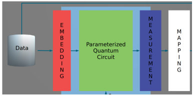
- How to get class labels from a distribution?
- When measuring n qubits: $j \in \{0,1\}^n$, some mapping function $j \to \{0,1\}$ is needed.

Mapping function



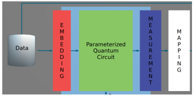
- How to get class labels from a distribution?
- When measuring n qubits: $j \in \{0,1\}^n$, some mapping function $j \to \{0,1\}$ is needed.
- Big hurdle while implementing

Mapping function

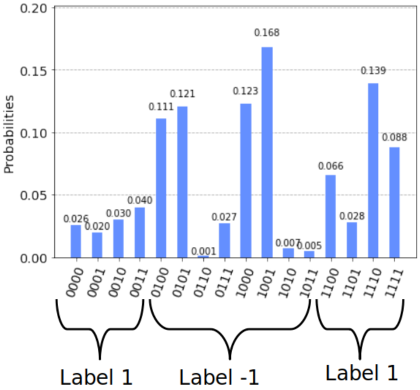


- How to get class labels from a distribution?
- When measuring n qubits: $j \in \{0,1\}^n$, some mapping function $j \to \{0,1\}$ is needed.
- Big hurdle while implementing
- $2^{2^4} = 65536$ possible binary mapping functions!

Mapping function

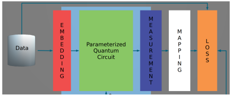


- How to get class labels from a distribution?
- When measuring n qubits: $j \in \{0,1\}^n$, some mapping function $j \to \{0,1\}$ is needed.
- Big hurdle while implementing
- $2^{2^4} = 65536$ possible binary mapping functions!



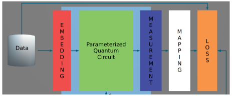
-

Loss function



- The task of learning an arbitrary function from data is mathematically expressed as the minimization of a loss function $L(\boldsymbol{\theta})$, also known as the objective function, with respect to the parameter vector $\boldsymbol{\theta}$.

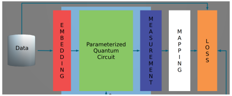
Loss function



- The task of learning an arbitrary function from data is mathematically expressed as the minimization of a loss function $L(\boldsymbol{\theta})$, also known as the objective function, with respect to the parameter vector $\boldsymbol{\theta}$.
- Squared Loss:

$$L(\boldsymbol{\theta}) = (y - \hat{y})^2$$

Loss function



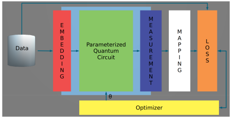
- The task of learning an arbitrary function from data is mathematically expressed as the minimization of a loss function $L(\boldsymbol{\theta})$, also known as the objective function, with respect to the parameter vector $\boldsymbol{\theta}$.
- Squared Loss:

$$L(\boldsymbol{\theta}) = (y - \hat{y})^2$$

-

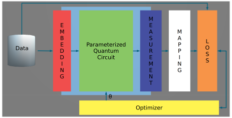
$$\hat{y} = label * max(probabilities)$$

Optimizer



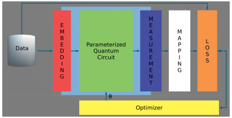
- Goal: minimize loss function

Optimizer



- Goal: minimize loss function
- Problem: Backpropagation is no easy feat

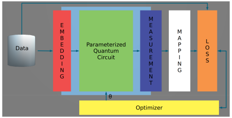
Optimizer



- Goal: minimize loss function
- Problem: Backpropagation is no easy feat
- Solution: Parameter shift rule:

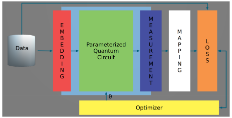
$$\frac{\partial L}{\partial \theta_j} \approx \frac{L(\boldsymbol{\theta} + c\boldsymbol{\Delta}) - L(\boldsymbol{\theta} - c\boldsymbol{\Delta})}{2c\Delta_j}$$

Optimizer



- Goal: minimize loss function
- Problem: Backpropagation is no easy feat
- Solution **???**

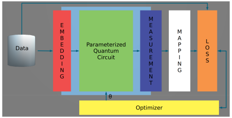
Optimizer



- Goal: minimize loss function
- Problem: Backpropagation is no easy feat
- Solution **???**
- Has to be executed twice for each parameter!

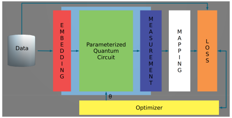
$$2 * \#Params * \#Shots$$

Optimizer



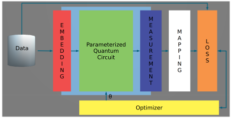
- Solution: SPSA (Simultaneous perturbation stochastic approximation)

Optimizer



- Solution: SPSA (Simultaneous perturbation stochastic approximation)
- only $2 * \#Shots$ evaluations

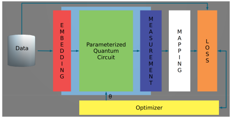
Optimizer



- Solution: SPSA (Simultaneous perturbation stochastic approximation)
- only $2 * \#Shots$ evaluations

$$\frac{\partial L}{\partial \theta_j} \approx \frac{L(\boldsymbol{\theta} + \Delta \mathbf{e}_j) - L(\boldsymbol{\theta} - \Delta \mathbf{e}_j)}{2\Delta}$$

Optimizer

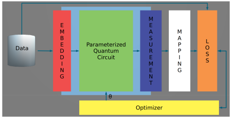


- Solution: SPSA (Simultaneous perturbation stochastic approximation)
- only $2 * \#\mathit{Shots}$ evaluations

$$\frac{\partial L}{\partial \theta_j} \approx \frac{L\left(\boldsymbol{\theta} + \Delta \mathbf{e}_j\right) - L\left(\boldsymbol{\theta} - \Delta \mathbf{e}_j\right)}{2\Delta}$$

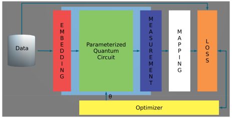
- $\rightarrow$ evaluate the gradient for each dimension joint, not independently.

Optimizer



- Now, that we have a gradient, gradient descent can be used!

Optimizer



- Now, that we have a gradient, gradient descent can be used!
- Or, even better ADAM, which has some improvements over regular gradient descent

Section 3

Results

Pennylane.ai

- I used pennylane by XANADU.

Pennylane.ai

- I used pennylane by XANADU.

- python framework

Pennylane.ai

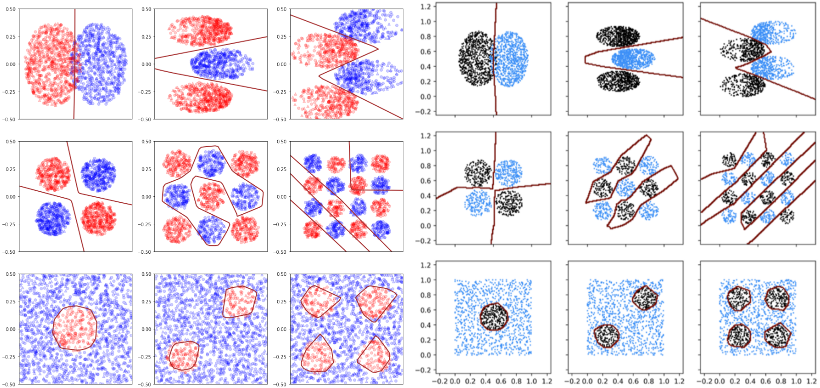
- I used pennylane by XANADU.
- python framework
- focused on quantum machine learning and quantum chemistry

Pennylane.ai

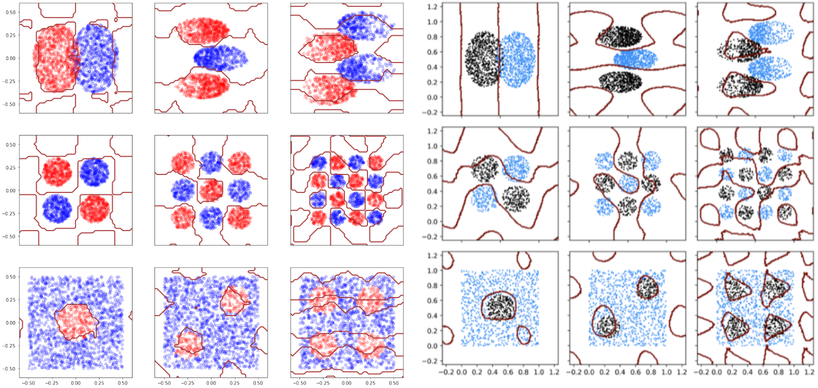
- I used pennylane by XANADU.

- python framework
- focused on quantum machine learning and quantum chemistry
- Interfaces with Pytorch, Tensorflow and even Qiskit.

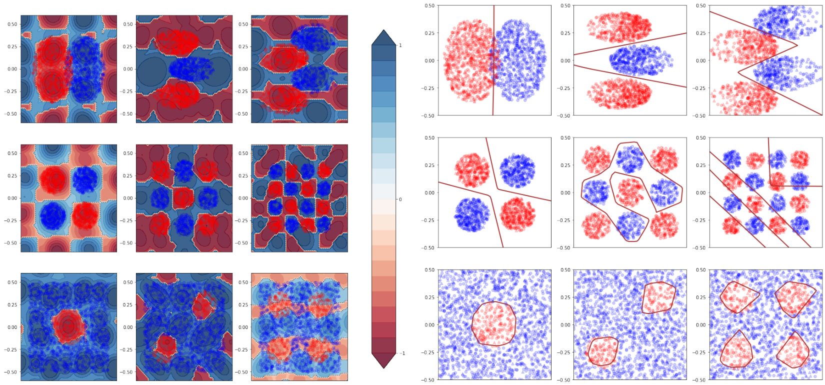
Results I: Neural Networks



Results II: Quantum Circuits



Results III: QC vs. NN



Section 4

Conclusion & Outlook

Conclusion

- Hybrid learning will be a powerful technique on near term noisy quantum computers
- To date, all supervised learning experiments involved scaled-down, often trivial, data sets due to the limitation of available quantum hardware, and demonstrations at a more realistic scale are desirable.
- One possible pitfall is that as the circuits become more expressive, the optimization landscape might also become harder to explore.
- Provided that we can efficiently load or prepare quantum data in a qubit register, PQC models will deliver a clear advantage over classical methods for quantum learning tasks.

Outlook

What's next?

- Have a look at the code and try it yourself!
- Try a different initialization scheme to avoid barren plateaus
- Try combinations of different circuits, layers, embedding or cost functions!
- Try different optimizers, such as gradient free optimizer (e.g. genetic algorithms) of try an analytical gradient!
- Try regularization!
- Try to find a real-world data set!
- Go deeper!
- read the documentation in-depth at www.qu.antum.ml

Thank you for your attention!
:)

Eyecandy

Training scheme

- Another problem; as only the Learning rate schedule parameters are mentioned:

Training scheme

- Another problem; as only the Learning rate schedule parameters are mentioned:
- Learning rate schedule: changed every 30 epochs (4, 3, 2, 1.5 ,1 ,0.5)

Training scheme

- Another problem; as only the Learning rate schedule parameters are mentioned:
- Learning rate schedule: changed every 30 epochs (4, 3, 2, 1.5 ,1 ,0.5)
- Minibatch size, betas, learning rate decay, ... are unknown (and were determined by trial& error)

Backup: Results IV: accuracy scores

Setup		Performance									
Type	Alg.	$1a$	$1b$	$1c$	$2a$	$2b$	$2c$	$3a$	$3b$	$3c$	Average
Classical	$\mathrm{NN}-2\mathrm{l}$	96%	100%	100%	100%	99%	98%	100%	99%	97%	99%
Classical (mine)	$\mathrm{NN}-2\mathrm{l}$	95%	100%	100%	100%	96%	87%	100%	98%	94%	97%
Hybrid	$\mathrm{c}19-2\mathrm{l}$	95%	97%	84%	91%	94%	74%	93%	91%	87%	90%
Hybrid (mine)	$\mathrm{c}19-2\mathrm{l}$	95%	99%	98%	100%	99%	98%	96%	97%	80%	95%

Sources

Hubregtsen, T. and Pichelmaier, J and Bertels, K. (2020).
"Evaluation of Parameterized Quantum Circuits: on the design, and the relation between classification accuracy, expressibility and entangling capability"
arXiv preprint arXiv:2003.09887.

Benedetti, M. and Lloyd, E. and Sack, S. and Fiorentini, M. (2019).
"Parameterized quantum circuits as machine learning models"
arXiv preprint arXiv:1906.07682v2.

Sim, S. and Johnson, PD. and Aspuru-Guzik, A. (2019).
"Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms"
arXiv preprint arXiv:1906.07682v2.