




Hybrid Helmholtz machines: a gate-based quantum circuit implementation

Teresa J. van Dam^{1,2} · Niels M. P. Neumann²  · Frank Phillipson² · Hans van den Berg^{2,3,4}

Received: 30 October 2019 / Accepted: 29 March 2020 / Published online: 22 April 2020
© Springer Science+Business Media, LLC, part of Springer Nature 2020

Abstract

Quantum machine learning has the potential to overcome problems that current classical machine learning algorithms face, such as large data requirements or long learning times. Sampling is one of the aspects of classical machine learning that might benefit from quantum machine learning, as quantum computers intrinsically excel at sampling. Current hybrid quantum-classical implementations provide ways to already use near-term quantum computers for practical applications. By expanding the horizon on hybrid quantum-classical approaches, this work proposes the first implementation of a gated quantum-classical hybrid Helmholtz machine, a gate-based quantum circuit approximation of a neural network for unsupervised tasks. Our approach focuses on parameterized shallow quantum circuits and effectively implements an approximate Bayesian network, overcoming the exponential complexity of exact networks. In addition, a new balanced cost function is introduced, preventing the need of millions of training samples. Using a bars and stripes data set, the model, implemented on the Quantum Inspire platform, is shown to outperform classical Helmholtz machines in terms of the Kullback–Leibler divergence.

Keywords Helmholtz machine · Machine learning · Quantum computing · Hybrid algorithms · Gate-based quantum computing

✉ Niels M. P. Neumann
niels.neumann@tno.nl

¹ Department of Data Science and Knowledge Engineering, Maastricht University, PO Box 616, 6200 MD Maastricht, The Netherlands

² The Netherlands Organisation for Applied Scientific Research (TNO), PO Box 96800, 2509 JE The Hague, The Netherlands

³ University of Twente, PO Box 217, 7500 AE Enschede, The Netherlands

⁴ Centre for Mathematics and Computer Science (CWI), PO Box 94079, 1090 GB Amsterdam, The Netherlands

1 Introduction

Machine learning is a widespread concept with many applications in a wide range of domains. Machine learning is especially well suited for tasks as image recognition [13] and pattern identification [6], tasks that are easy for us humans. Furthermore, it is used for tasks as spam filtering [7] and evaluating customer behavior [2,5]. In short, machine learning algorithms try to make predictions by extracting (meaningful) information from data. Machine learning algorithms giving exact answers are hard to develop, as, even for the simplest problems, they quickly become complex. Therefore, most algorithms focus on approximating model parameters by means of *learning*. This learning is done by training the algorithm based on the given data. The more data used, the better the model will approximate the real situation. However, using more data also implies an increase in computational complexity. Therefore, the efficiency of machine learning algorithms is critical for them to remain feasible.

Quantum computing has the potential to overcome some of these efficiency hurdles. Even though the current state of the art does not yet allow solving complex real-world problems, first implementations of toy problems are already available and show promising results [8,29–31]. Furthermore, for small toy problems, requiring only limited resources, the algorithms can also be simulated on classical computers to understand its workings.

Learning efficiency improvements are not the only benefit quantum machine learning offers. Also running time and capacity improvements in associative or content-addressable memories are expected [4,17,21,25].

While completely new quantum machine algorithms are possible and work is done on developing them, the resources required to run these quantum algorithms on real-life data are far from what is available today. Even for quantum versions of classical algorithms [28], implementations require just too many resources to be feasible. However, significant improvements over classical algorithms can already be obtained without full quantum algorithms. In fact, often there is only a single (sub)routine that takes up most of the computational resources. By changing these complex routines to more efficient quantum analogs, significant improvements can be obtained [4]. Such algorithms, consisting of both classical and quantum parts, are called *hybrid* algorithms.

One special type of hybrid algorithms is a variational quantum eigensolver (VQE), where the quantum operations are tweaked using classical routines [22]. In general, a VQE approximates the ground-state energy of a quantum system. As quantum resources are used together with classical resources, even on small near-term devices, so-called NISQ devices [23], quantum computing can be useful. Improvements on both the classical and the quantum parts of the original work are given in [16]. Variational quantum eigensolvers have already been used to solve different problems. Aside from (quantum) chemistry-related problems [20,26], VQEs have also been used to factor integers [1].

We focus on another application of VQEs: generative modeling. Generative models are used to learn probability distributions over (high-dimensional) data sets. By increasing the depth of a generative model, the generalization capability grows and more abstract representations of the data can be found, however, at the cost of intractable inference and training. Both inference and training rely on variational

approximations and Markov chain Monte Carlo sampling, both being computationally expensive. As quantum computers allow for efficient sampling, the expensive sampling subroutine can be run on a quantum computer, thus reducing the computational complexity of generative models significantly.

This paper gives an implementation of a Helmholtz machine, a special type of generative model, on a gate-based quantum computer. A Helmholtz machine is an artificial neural network consisting of a bottom-up recognition network and a top-down generative network. The recognition network takes data and produces probability distributions over it, while the generative network generates representations of the data and the hidden variables. In our approach, the generative sampling is implemented on a gate-based quantum computer.

Some other work that considered the Helmholtz machine focused on implementing the Helmholtz machine on quantum annealing devices [4], a different way of quantum computing [10]. Furthermore, work is done on a quantum circuit learning algorithm used to train shallow circuits [3] and a qubit neuron model [19], where the learning is performed by a quantum-modified backpropagation algorithm.

In this paper, we first explain the Helmholtz machine and the related learning algorithm in Sect. 2. The main result of this work is given in Sect. 3, as it presents the shallow circuit implementation of the hybrid Helmholtz machine and introduces the data set used in the experiments. The training of this circuit is considered in Sect. 4. In Sect. 5, we will compare the performance of a hybrid Helmholtz machine with a classical Helmholtz machine. Finally, Sect. 6 gives the conclusions as well as directions of future work.

2 The Helmholtz machine

A Helmholtz machine is a special artificial neural network used to learn hidden structures in data. The Helmholtz machine sees the world as patterns of bits, in which each bit pattern \mathbf{d} (i.e., a binary string) appears with certain probability $p(\mathbf{d})$.

2.1 Structure

Specifying $p(\mathbf{d})$ exactly requires a lot of information, and even approximating the probability might be exponentially hard [27], as the number of possible patterns \mathbf{d} is exponential in the number of bits. In a completely random world, this indeed poses a challenge; however, in practice often less information is required to still obtain a good approximation of the probability $p(\mathbf{d})$, as reality often is not purely random. The Helmholtz machine aims at taking advantage of this [11]. A Helmholtz machine consists of two artificial neural networks: the first, a bottom-up recognition network R that approximates inference on the hidden units with information extracted from real data, and the second, a top-down generative network G that generates artificial data. These networks together aim on learning probability distributions of a data set by finding hidden structures.

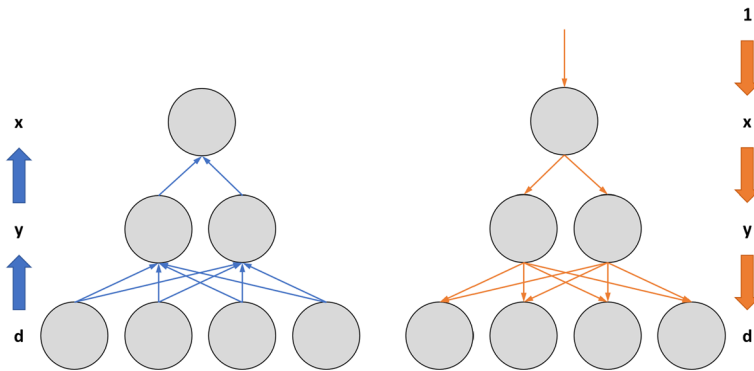


Fig. 1 Illustration of a classical Helmholtz machine (CHM) 4–2–1 containing a visible layer with four nodes and two hidden layers with two and one nodes, respectively

The generative network G performs top-down sampling, starting from the deepest hidden layer. In each layer, it tries to reconstruct the data obtained from the previous layer [9]. To perform this, it assumes a generative distribution $p_G(\mathbf{d})$ that produces pattern \mathbf{d} in a probabilistic way from a casual chain, given by

$$1 \rightarrow \mathbf{x}^1 \rightarrow \dots \rightarrow \mathbf{x}^{n-1} \rightarrow \mathbf{d},$$

where each \mathbf{x}^i is a hidden layer, with \mathbf{x}^1 the deepest hidden layer. The probability for a pattern \mathbf{d} is given by

$$\begin{aligned} p_G(\mathbf{x}^1, \dots, \mathbf{x}^{n-1}, \mathbf{d}) &= p_G(\mathbf{d} \mid \mathbf{x}^{n-1}) \cdot p_G(\mathbf{x}^{n-2} \mid \mathbf{x}^{n-1}) \cdot \dots \cdot \\ & p_G(\mathbf{x}^2 \mid \mathbf{x}^1) \cdot p_G(\mathbf{x}^1). \end{aligned} \tag{1}$$

A Helmholtz machine also has a recognition network R that is used to learn the weights of the generative network and hence the generative probability distribution. The causal chain for a recognition network is given by

$$\mathbf{d} \rightarrow \mathbf{x}^{n-1} \rightarrow \dots \rightarrow \mathbf{x}^1.$$

Note the link with the generative network. The recognition network approximates the true intractable distribution $p(\mathbf{d})$ using a bottom-up pass. Here, sampling is performed on the available data set and then passed through the network R to approximate the posterior distribution.

In Fig. 1, an example of a three-layered Helmholtz machine is shown, with the blue arrow indicating the recognition networking and the orange arrows indicating the generative network. The causal chain of the generative network is given by $1 \rightarrow \mathbf{x} \rightarrow \mathbf{y} \rightarrow \mathbf{d}$, while that of the recognition network is given by $\mathbf{d} \rightarrow \mathbf{y} \rightarrow \mathbf{x}$.

For this example, the generative probability of a bit pattern is given by

$$\begin{aligned} p_G(\mathbf{x}, \mathbf{y}, \mathbf{d}) \\ = p_G(\mathbf{d} | \mathbf{y}) \cdot p_G(\mathbf{y} | \mathbf{x}) \cdot p_G(\mathbf{x}). \end{aligned}$$

A Helmholtz machine tries to match the generative distribution $p_G(\mathbf{d})$ with the real-world distribution $p(\mathbf{d})$ and has learned perfectly if the two match. Often, it is enough for the generative distribution to approximate the real-world distribution sufficiently well. Parameter settings are obtained through a process of learning.

2.2 Learning the Helmholtz machine

Training the Helmholtz machine is done in an iterative manner, where each iteration consists of two phases. In the first phase, the recognition network R is fixed, while the generative network G learns the posterior probability distribution. In the second phase, G is fixed and R learns the prior probability distribution. The Wake–Sleep algorithm [9] is a well-suited example of such an iterative learning algorithm.

This algorithm is an unsupervised learning algorithm, used to learn probability distributions with which data cannot only be efficiently recognized, but also accurately generated. A Helmholtz machine learns the structure of the world by minimizing the variational free energy

$$F_G^R(\mathbf{d}) = F_G(\mathbf{d}) + \text{KL}[p_R(\text{hidden}|\mathbf{d})||p_G(\text{hidden}|\mathbf{d})], \quad (2)$$

with KL the Kullback–Leibler divergence [12], used to quantify the similarity between probability distributions. In this case, the probability distributions of the hidden layers found by the recognition and generative networks R and G , $p_R(\text{hidden}|\mathbf{d})$ and $p_G(\text{hidden}|\mathbf{d})$, respectively.

By minimizing $F_G^R(\mathbf{d})$, simultaneously the Kullback–Leibler divergence

$$\text{KL}[p_R(\text{hidden}|\mathbf{d})||p_G(\text{hidden}|\mathbf{d})]$$

and the free energy $F_G(\mathbf{d})$ are also minimized as both quantities are positive. Each iteration of the Wake–Sleep algorithm involves two phases: the Wake phase, adjusting the weights of the generative network θ_G , and the Sleep phase, adjusting the weights of the recognition network θ_R . Hence, during training, alternately small changes to the weights of both networks are made.

In the Wake phases during training, small changes to θ_G are made to decrease the variational free energy $F_G^R(\mathbf{d})$, where \mathbf{d} is a random sample (data point) from the real world $p(\mathbf{d})$. Note that \mathbf{d} is not taken from the generative network G . In this phase, the data point is given to the recognition causal chain, which then generates the explanation $\mathbf{x}^1 \dots \mathbf{x}^{n-1}$. Afterward, the weights are updated based on a delta rule.

In the second phase, the Sleep phase, a random sample is taken from the generative distribution and given to the causal chain. This will eventually produce the pattern \mathbf{d} . The recognition weights θ_R are updated to decrease

$$\tilde{F}_G^R(\mathbf{d}) = F_G(\mathbf{d}) + \text{KL}[p_G(\text{hidden}|\mathbf{d})||p_R(\text{hidden}|\mathbf{d})]. \quad (3)$$

Note the difference between the order of the arguments in Eqs. (2) and (3). Due to the asymmetry of the Kullback–Leibler divergence, see also Eq. (5), the two results differ.

At the beginning of training, all weights are set to zero. Hence, every neuron, initially, has an equal chance of *firing* and *not firing*, i.e., be activated or not. Afterward, in each iteration first a Wake phase is performed to update θ_G and then a Sleep phase to update θ_R , as explained before. By means of updating the weights, the difference between the generative probability distribution $p_G(\mathbf{d})$ and the real probability distribution $p(\mathbf{d})$ is reduced.

In the next section, we will give a detailed explanation of our implementation of the hybrid quantum-classical Helmholtz machine, as well as the data set and cost function used in our experiment.

3 Parameterized shallow quantum circuit

Implementing a hybrid quantum-classical Helmholtz machine on a quantum computer faces us with some challenges, one of which is the available resources. At least on the near term, coherence times of qubits (T_1 times) are not yet sufficient for long quantum circuits with many gates. The resource demands of a hybrid Helmholtz machine are high, while the available resources are limited, at least on the short term. Therefore, it is difficult to implement a quantum circuit able to perform uniform sampling from the data [3]. The main source of computational complexity comes from the intractability of performing Bayesian inference over probability distributions. The complexity of an exact representation of a Bayesian network in the form of a quantum circuit is exponential in the number of qubits; hence, an exact representative quantum circuit exists with complexity $\mathcal{O}(2^n)$ [14]. Because of this, we propose an approximation of such quantum circuit: a parameterized shallow quantum circuit (PSQC), which is of reduced complexity, compared to an exact quantum circuit, but still capable enough to characterize an arbitrary joint probability distribution. We focus on N -dimensional binary vectors $\mathbf{x} \in \{0, 1\}^N$; however, extensions to other data formats are easily made. Examples of such binary vectors are black and white images, as there exists a one-to-one mapping between input vectors and the computational basis of an N -qubit quantum system: $\mathbf{x} = (x_1, \dots, x_N) \leftrightarrow |x_1 x_2 \dots x_N\rangle$.

3.1 From network topologies to quantum circuits

Following [14, Theorem 1], we know that arbitrary probability distributions can be prepared using an exponential number of gates. However, implementations with an exponential number of gates are impractical and inefficient when training the circuit. Instead, we use an approximated circuit using fewer quantum gates, thus allowing for more efficient training. Therefore, our PSQC is implemented by using an approxima-

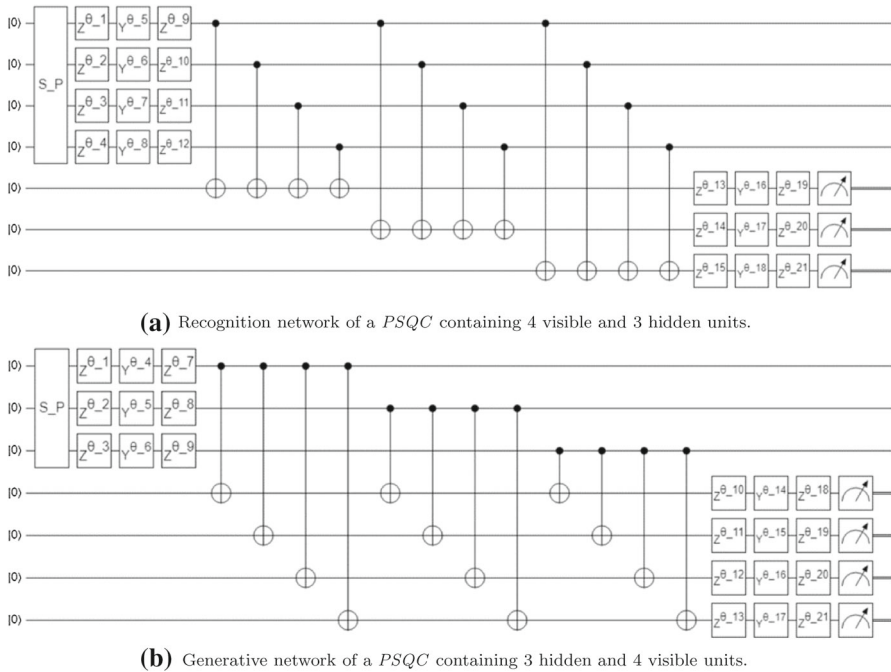


Fig. 2 Recognition and generative network of a PSQC, with four visible nodes and three hidden nodes each. The visible nodes for the recognition network are the input nodes. The output nodes are the visible nodes of the generative network

tion of the quantum circuit representation of the generative and recognition networks G and R of the hybrid Helmholtz machine.

To map a classical Helmholtz machine to a gate-based quantum circuit, a qubit is assigned for every pixel in the input data. This allows us to represent the input vectors by the first qubits of the quantum circuit. Furthermore, a qubit is assigned to every node in each subsequent layer. Now, each node of the classical Helmholtz machine is represented by a qubit. Different nodes in consecutive layers are connected by CNOT operations, and the corresponding weights are applied by parameterized single-qubit rotations on the qubits of the former layer. These parameters, or angles, are in the range $[-\pi, \pi]$.

An example of a recognition circuit with four visible and three hidden nodes is shown in Fig. 2a. Here, S_P stands for *state preparation*, applying an X operation on qubits, depending on the corresponding pixel value. This gives the qubit the same initial value as that of the pixel. Now, single-qubit rotation gates are applied to each qubit in the visible layer, following the Z - Y decomposition [18], such that $U = e^{i\alpha} R_z(\beta)R_y(\gamma)R_z(\delta)$, where α, β, γ and δ are real-valued. This decomposition allows a qubit to be mapped to every possible single-qubit quantum state, given the correct parameters.

An optimal set of parameters allows us to discover the hidden probability distribution present in the input data (i.e., $p(\mathbf{d})$). The visible layer is then connected with

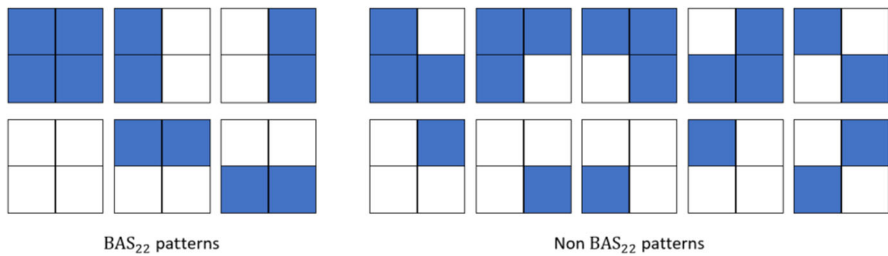


Fig. 3 Illustration of the BAS_{22} data set. All patterns in the BAS_{22} class have probability $1/6$, while all other patterns have probability 0

the first hidden layer using CNOT operations, analogous to the bipartite structure of a Helmholtz machine. Again, single-qubit operations are applied, this time on the qubits in the first hidden layer. For each additional hidden layer, extra CNOT operations are used, again with single-qubit operations on the corresponding qubits. Finally, the qubits of the last hidden layer are measured.

The corresponding example of a generative circuit is shown in Fig. 2b, again with four visible and three hidden nodes. The visible layer of the generative circuit is the last layer, in accordance with the classical networks shown in Fig. 1. State preparation is performed similarly as for the recognition circuit, by means of X gates, based on the corresponding pixel values. Afterward, single-qubit rotations are applied and CNOT gates are used between different layers, corresponding to the bipartite structure of the generative network. Again, the qubits of the last layer are measured, which are, for the generative network, those in the visible layer.

For this 4–3 topology of the Helmholtz machine, there are 42 parameters to be optimized through learning, 21 in the generative network and 21 in the recognition network.

3.2 Data set

Our implemented hybrid Helmholtz machine focuses on the bars and stripes (BAS_{nm}) data set [15, p. 526], consisting of $n \times m$ pixel patterns. Each pattern in this data set contains either only bars or only stripes. The BAS_{nm} data set has $2^n + 2^m - 2$ valid patterns, out of the 2^{nm} possible patterns. Each valid pattern is equally likely. All other patterns have zero probability of occurring. We use the BAS_{22} data set, see Fig. 3, requiring four visible input and output nodes.

3.3 Cost function

In optimization problems, and hence also in training machine learning models, a cost function is used to express the quality of a solution or a specific set of parameters. We use a cost function to estimate the fit of the probability distribution p_G . To prevent us from having to take millions of samples to get a good estimate of the probability distribution, we propose the following cost function $C_\theta(R, G)$:


```

Learning of the Quantum Circuit

 $\theta_R = \{\text{uniform.random}[-\pi, \pi]\}^{21}$ 
 $\theta_G = \{\text{uniform.random}[-\pi, \pi]\}^{21}$ 

For i in range(nCycles):
  While maxFuncEval  $\leq$  1,000:
    Wake Phase to update  $\theta_G$ 
  While maxFuncEval  $\leq$  1,000:
    Sleep Phase to update  $\theta_R$ 
    
```

Algorithm 1: Learning algorithm for our *PSQC* model

```

WAKE-PHASE - QUANTUM CIRCUIT
#Set initial angles at random
anglesR =  $\theta_R$ 
anglesG =  $\theta_G$ 

#Experience Reality
visibleR = SampleFromWorld(BAS22)

#Bottom-Up pass through Recognition Network
hiddenR = Recognition(visibleR, anglesR)

# Top-down pass through Generator Network
hiddenG = hiddenR
visibleG = Generative(hiddenG, anglesG)

#Calculate value of cost function
 $C_\theta(R, G) = f(\text{visibleR}, \text{hiddenR}, \text{hiddenG}, \text{visibleG})$ 

#Update generator parameters by gradient-free optimizer
anglesG = Optimizer.minimize( $C_\theta(R, G)$ )
    
```

Algorithm 2: *Wake*-phase in the training of the *PSQC*. The passes through Recognition and Generative are evaluated using a quantum computer

$$\begin{aligned}
 C_\theta(R, G) = & \sum_i \sum_j (\langle V_i \cdot H_j \rangle_R - \langle V_i \cdot H_j \rangle_G)^2 \\
 & + \sum_i (\langle V_i \rangle_R - \langle V_i \rangle_G)^2 \\
 & + \sum_j (\langle H_j \rangle_R - \langle H_j \rangle_G)^2.
 \end{aligned}
 \tag{4}$$

This, together with the transformation $s \mapsto 2s - 1$, gives a balanced cost function as $1 \mapsto 1$ and $0 \mapsto -1$. Our cost function uses the first and second moments of visible (V_i) and hidden (H_j) variables of both networks (indicated by a subscript) without the need of estimating probabilities. The estimated cost is used to update the model parameters to further lower the cost.

```

SLEEP-PHASE QUANTUM CIRCUIT
#Set initial angles at random
anglesR =  $\theta_R$  used in Wake phase
anglesG =  $\theta_G$  coming from the previous Wake phase

#Initiate dream
hiddenG = SampleFromHidden(Hidden)

#Top-down pass through Generative Network
visibleG = Generative(hiddenG, anglesG)

#Bottom-Up pass through Recognition Network
visibleR = visibleG
hiddenR = Recognition(visibleR, anglesR)

#Calculate value of cost function
 $C_\theta(R, G) = f(\text{visibleR}, \text{hiddenR}, \text{hiddenG}, \text{visibleG})$ 

#Update recognition parameters by gradient-free optimizer
anglesR = Optimizer.minimize( $C_\theta(R, G)$ )

```

Algorithm 3: *Sleep*-phase in the training of the PSQC. The passes through Generative and Recognition are evaluated using a quantum computer

4 Training the PSQC

The learning process in our PSQC is performed by implementing a Wake–Sleep algorithm; see also Sect. 2.2. In this implementation, the *Powell* method is used as a black-box gradient-free optimizer to train the circuit. Per iteration the *Powell* method evaluates the cost function up to 1,000 times, to find optimal parameters for the recognition and generative networks R and G . The pseudo-code of the learning algorithm is given in Algorithm 1, where the *Wake* and *Sleep* phases are described in Algorithms 2 and 3, respectively.

To measure the performance of the PSQC, the learned probability distribution of the Helmholtz machine is compared with the real probability distribution using the Kullback–Leibler divergence [12] defined by

$$\text{KL}[P||P_\theta] = \sum_{\mathbf{d}} p(\mathbf{d}) \log \frac{p(\mathbf{d})}{p_\theta(\mathbf{d})}, \quad (5)$$

with P the real probability distribution and P_θ the generated probability distribution for parameters θ . Note, $\text{KL}[P||P_\theta] = 0$ holds if and only if both distributions are equal.

5 Results: classical versus hybrid Helmholtz machine

We compared the performance of a classical Helmholtz machine with that of a hybrid Helmholtz machine in terms of the KL divergence. Both the classical and the hybrid Helmholtz machines are trained using 1,000 training iterations and in each iteration

Table 1 KL divergence averaged over 10 runs for both the parameterized shallow quantum circuit PSQC and classical Helmholtz machine CHM for the topology 4–3

	Patterns	$P(\mathbf{d})$	$P_G(\mathbf{d})$	
			PSQC 4–3	CHM 4–3
WORLD	10 10	0.167	0.051	0.080
	01 01	0.167	0.059	0.070
	11 00	0.167	0.082	0.070
	00 11	0.167	0.108	0.070
	11 11	0.167	0.079	0.080
	00 00	0.167	0.121	0.090
OTHERS	10 00	0.000	0.066	0.050
	10 01	0.000	0.036	0.040
	10 11	0.000	0.060	0.060
	11 01	0.000	0.032	0.040
	11 10	0.000	0.025	0.060
	00 01	0.000	0.046	0.070
	00 10	0.000	0.055	0.040
	01 00	0.000	0.067	0.080
	01 10	0.000	0.047	0.060
	01 11	0.000	0.068	0.050
KL divergence ($p(\mathbf{d})$, $p_G(\mathbf{d})$)			0.793	0.945

100 samples are drawn from the *real world* containing the BAS_{22} patterns. In each iteration, the *Wake* and *Sleep* phase use the Powell method to optimize the parameters, based on the corresponding evaluations of the cost function of Eq. (4). The number of learning cycles in the Wake–Sleep algorithm is fixed to 10 for both the classical and the hybrid Helmholtz machines. The performance is consequently evaluated using Eq. (5).

For both the classical and the hybrid Helmholtz machines, we used a 4–3 topology, with four visible nodes and three hidden ones. The PSQC is implemented in a hybrid fashion on the Quantum Inspire simulator [24] and trained using the Wake–Sleep algorithm as given in Algorithms 2 and 3, respectively. The classical Helmholtz machine is implemented in Python and trained with a standard and unmodified Wake–Sleep algorithm.

The resulting probability distributions learned with both the classical and the hybrid Helmholtz machines are shown in Table 1. Furthermore, the two are compared based on the Kullback–Leibler convergence. In the table, we see that the PSQC 4–3 has a smaller divergence than its classical analog. This means that our proposed model of a quantum circuit of reduced complexity is capable of performing generative tasks and scoring at least as good as its classical equivalent. In Fig. 4, we show the evolution of the KL divergence over the number of iterations of both models.

6 Conclusions and future work

In this work, a parameterized shallow quantum circuit implementation of a hybrid Helmholtz machine was given. This PSQC captures aspects of Bayesian networks and

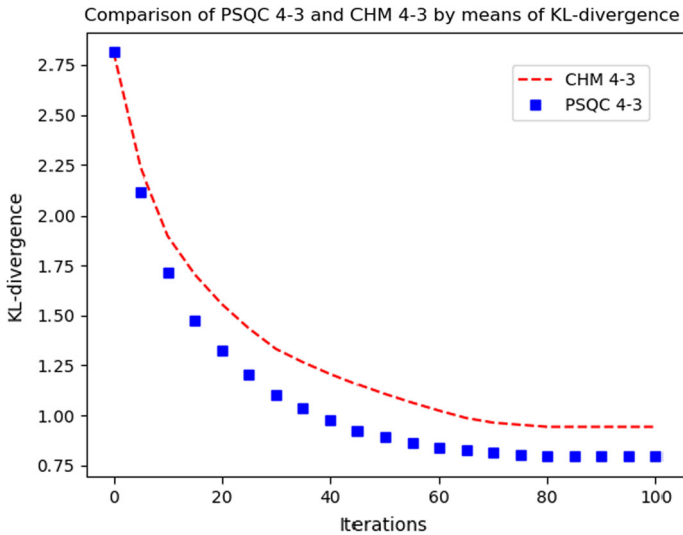


Fig. 4 KL divergence of the PSQC and the CHM for topology 4–3

Helmholtz machines and was trained by a gradient-free optimizer under an adaptation of the *Wake–Sleep* algorithm. The implementation of this circuit was done on the Quantum Inspire simulator and has the potential to be run on a few qubit quantum device.

We tested the proposed hybrid Helmholtz machine for a small toy problem on the BAS_{22} data set, consisting of two by two pixel images, of which the valid patterns are those that contain only bars or only stripes. For both the hybrid and the classical Helmholtz machines, we used four visible nodes and three hidden ones. The used Powell optimization method gave a promising set of parameters, for which the hybrid Helmholtz machine slightly outperformed its classical counterpart. Using the cost function, the probability distribution $p(\mathbf{x})$ can be approximated efficiently, without the need for an exponential number of circuit evaluations.

The proposed hybrid Helmholtz machine can be improved upon further by a more refined training, possibly based on the specific data set. The used symmetric cost function might be replaced by other cost function, possibly forfeiting the symmetry of the cost function if the data set asks for it. Finally, the sampling power of quantum computers can be exploited even further, as a Helmholtz machine depends heavily on sampling.

References

1. Anschuetz, E., Olson, J., Aspuru-Guzik, A., Cao, Y.: Variational quantum factoring. In: Feld, S., Linnhoff-Popien, C. (eds.) *Quantum Technology and Optimization Problems*, pp. 74–85. Springer, Cham (2019)
2. Badea Stroeie, L.M.: Predicting consumer behavior with artificial neural networks. *Procedia Econ Finance* **15**, 238–246 (2014). [https://doi.org/10.1016/S2212-5671\(14\)00492-4](https://doi.org/10.1016/S2212-5671(14)00492-4)

3. Benedetti, M., Garcia-Pintos, D., Perdomo, Leyton-Ortega, V., Nam, Y., Perdomo-Ortiz, A.: A generative modeling approach for benchmarking and training shallow quantum circuits. *npj Quantum Inf* **5**, 45 (2019). <https://doi.org/10.1038/s41534-019-0157-8>
4. Benedetti, M., Realpe Gomez, J., Perdomo-Ortiz, A.: Quantum-assisted helmholtz machines: a quantum-classical deep learning framework for industrial datasets in near-term devices. *Quantum Sci Technol* (2018). <https://doi.org/10.1088/2058-9565/aab98>
5. Berry, M.J., Linoff, G.: *Data Mining Techniques: For Marketing, Sales, and Customer Support*. Wiley, New York (1997)
6. Bishop, C.M.: *Neural Networks for Pattern Recognition*. Oxford University Press Inc, New York (1995)
7. Clark, J., Koprinska, I., Poon, J.: A neural network based approach to automated e-mail classification. In: *Proceedings IEEE/WIC International Conference on Web Intelligence (WI 2003)*, pp. 702–705 (2003). <https://doi.org/10.1109/WI.2003.1241300>
8. Coles, P.J., Eidenbenz, S., Pakin, S., Adedoyin, A., Ambrosiano, J., Anisimov, P., Casper, W., Chennupati, G., Coffrin, C., Djidjev, H., et al.: Quantum algorithm implementations for beginners. *arXiv:1804.03719* [quant-ph] (2018)
9. Hinton, G., Dayan, P., Frey, B., Neal, R.: The “wake-sleep” algorithm for unsupervised neural networks. *Science* **268**(5214), 1158–1161 (1995). <https://doi.org/10.1126/science.7761831>
10. Kadowaki, T., Nishimori, H.: Quantum annealing in the transverse ising model. *Phys. Rev. E* **58**, 5355–5363 (1998). <https://doi.org/10.1103/PhysRevE.58.5355>
11. Kirby, K.G.: A tutorial on helmholtz machines (2006). <https://www.nku.edu/~kirby/docs/HelmholtzTutorialKoeln.pdf>. Accessed 20 Apr 2020
12. Kullback, S., Leibler, R.A.: On information and sufficiency. *Ann Math Stat* **22**(1), 79–86 (1951). <https://doi.org/10.1214/aoms/1177729694>
13. Lawrence, S., Giles, C.L., Tsoi, A.C., Back, A.D.: Face recognition: a convolutional neural-network approach. *IEEE Trans Neural Netw* **8**(1), 98–113 (1997). <https://doi.org/10.1109/72.554195>
14. Low, G.H., Yoder, T.J., Chuang, I.L.: Quantum inference on Bayesian networks. *Phys Rev A* (2014). <https://doi.org/10.1103/PhysRevA.89.062315>
15. MacKay, D.J.C.: *Information Theory, Inference, and Learning Algorithms*, vol. 6. Cambridge University Press, Cambridge (2003)
16. McClean, J.R., Romero, J., Babbush, R., Aspuru-Guzik, A.: The theory of variational hybrid quantum-classical algorithms. *New J. Phys.* **18**(2), 023023 (2016). <https://doi.org/10.1088/1367-2630/18/2/023023>
17. Neumann, N.M.P., Phillipson, F., Versluis, R.: Machine learning in the quantum era. *Digitale Welt* **3**(2), 24–29 (2019). <https://doi.org/10.1007/s42354-019-0164-0>
18. Nielsen, M.A., Chuang, I.L.: *Quantum computation and quantum information*. Cambridge University Press, Cambridge (2008)
19. Nobuyuki, M., Haruhiko, N., Isokawa, T.: Qubit neural network: its performance and applications. *Neural Process. Lett.* **3**(22), 277–290 (2005). <https://doi.org/10.4018/978-1-60566-214-5.ch013>
20. O’Malley, P.J.J., Babbush, R., Kivlichan, I.D., Romero, J., McClean, J.R., Barends, R., Kelly, J., Roushan, P., Tranter, A., Ding, N., Campbell, B., Chen, Y., Chen, Z., Chiaro, B., Dunsworth, A., Fowler, A.G., Jeffrey, E., Lucero, E., Megrant, A., Mutus, J.Y., Neeley, M., Neill, C., Quintana, C., Sank, D., Vainsencher, A., Wenner, J., White, T.C., Coveney, P.V., Love, P.J., Neven, H., Aspuru-Guzik, A., Martinis, J.M.: Scalable quantum simulation of molecular energies. *Phys. Rev. X* **6**, 031007 (2016). <https://doi.org/10.1103/PhysRevX.6.031007>
21. Perdomo-Ortiz, A., Benedetti, M., Realpe-Gómez, J., Biswas, R.: Opportunities and challenges for quantum-assisted machine learning in near-term quantum computers. *Quantum Sci Technol* **3**(3), 030502 (2018). <https://doi.org/10.1088/2058-9565/aab859>
22. Peruzzo, A., McClean, J., Shadbolt, P., Yung, M.H., Zhou, X.Q., Love, P.J., Aspuru-Guzik, A., O’Brien, J.L.: A variational eigenvalue solver on a photonic quantum processor. *Nat Commun* (2014). <https://doi.org/10.1038/ncomms5213>
23. Preskill, J.: Quantum computing in the NISQ era and beyond. *Quantum* **2**, 79 (2018). <https://doi.org/10.22331/q-2018-08-06-79>
24. QuTech: (2019). <https://www.quantum-inspire.com/>. Accessed 13 May 2019
25. Reberntrost, P., Bromley, T.R., Weedbrook, C., Lloyd, S.: Quantum hopfield neural network. *Phys. Rev. A* **98**, 042308 (2018). <https://doi.org/10.1103/PhysRevA.98.042308>

26. Romero, J., Babbush, R., McClean, J.R., Hempel, C., Love, P.J., Aspuru-Guzik, A.: Strategies for quantum computing molecular energies using the unitary coupled cluster ansatz. *Quantum Sci. Technol.* **4**(1), 014008 (2018). <https://doi.org/10.1088/2058-9565/aad3e4>
27. Roth, D.: On the hardness of approximate reasoning. *Artif. Intell.* **82**(1–2), 273–302 (1996)
28. Schuld, M., Sinayskiy, I., Petruccione, F.: An introduction to quantum machine learning. *Contemp. Phys.* **56**(2), 172–185 (2015). <https://doi.org/10.1080/00107514.2014.964942>
29. Schuld, M., Sinayskiy, I., Petruccione, F.: Prediction by linear regression on a quantum computer. *Phys. Rev. A* **94**, 022342 (2016). <https://doi.org/10.1103/PhysRevA.94.022342>
30. Wiebe, N., Kapoor, A., Svore, K.M.: Quantum deep learning. [arXiv:1412.3489](https://arxiv.org/abs/1412.3489) [quant-ph] (2014)
31. Yoo, S., Bang, J., Lee, C., Lee, J.: A quantum speedup in machine learning: finding a N-bit Boolean function for a classification. *New J. Phys.* **16**(10), 103014 (2014). <https://doi.org/10.1088/1367-2630/16/10/103014>

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.