4th Symposium on Advances in Approximate Bayesian Inference, 2021 1–15

# Quantum Bayesian Neural Networks

**Anonymous Authors**
*Anonymous Institution*

## Abstract

Quantum machine learning promises great speedups over classical algorithms, but it often requires repeated computations to achieve a desired level of accuracy for its point estimates. Bayesian learning focuses more on sampling from posterior distributions than on point estimation, thus it might be more forgiving in the face of additional quantum noise. We propose a quantum algorithm for Bayesian neural network inference, drawing on recent advances in quantum deep learning, and simulate its empirical performance on several tasks. We find that already for small numbers of qubits, our algorithm approximates the true posterior well, while it does not require any repeated computations and thus fully realizes the quantum speedups.

## 1. Introduction

Quantum machine learning generally comes in two flavors: Variational quantum circuits that mimic the training of neural networks (Cerezo et al., 2020), which run on noisy intermediate-scale quantum (NISQ) devices (Preskill, 2018), and quantum algorithms aimed at replacing classical training and prediction algorithms for neural networks (Allcock et al., 2020), which run on (future) error-corrected quantum computers. The latter often use a quantum algorithm for estimating the inner product calculations that occur when training and evaluating neural networks. This inner product estimation (IPE) can evaluate inner products with lower asymptotic complexity than classical algorithms but does so at lower accuracy. When evaluating standard neural networks, this lowered accuracy in the inner product calculation has to be corrected by running the quantum algorithm multiple times to get a better estimate.

However, for Bayesian neural networks (BNNs), the goal is not to get the best point estimate of the parameters $\theta^* = \arg\max_\theta p(\mathcal{D} \mid \theta)$, as it would be in maximum-likelihood learning. Instead, one wishes to obtain $K$ samples from the Bayesian posterior over the parameters $\theta$ given the data $\mathcal{D}$, that is, $\theta_i \sim p(\theta \mid \mathcal{D}) \propto p(\theta)\, p(\mathcal{D} \mid \theta)$. These samples can then be used to approximate the posterior predictive for unseen data $\mathcal{D}^*$, that is, $p(\mathcal{D}^* \mid \mathcal{D}) = \int p(\mathcal{D}^* \mid \theta)\, p(\theta \mid \mathcal{D})\, \mathrm{d}\theta \approx \frac{1}{K} \sum_{i=1}^{K} p(\mathcal{D}^* \mid \theta_i)$.

Since these samples are noisy by stipulation, they might allow for a larger margin of error in the quantum computations. Ideally, under zero-mean quantum noise with sufficiently small variance, one might achieve meaningful results with just running one single quantum computation per sample, thus realizing the maximum possible quantum speedup. In this work, we investigate this idea empirically and demonstrate a proof of concept for BNN inference on quantum computers. We find that already for decently small numbers of qubits, our quantum algorithm approximates the true posterior well with just one computation per sample. A discussion of related work is deferred to Appendix A.

© A. Authors.

## 2. Quantum Bayesian Neural Networks

This paper focuses on reducing the asymptotic runtime of the inference and prediction in BNNs using quantum algorithms. The algorithms described in this paper are derived from the quantum deep learning algorithms introduced by Allcock et al. (2020) for feedforward neural networks. We will describe the alterations made for BNNs and their consequences in this section.

### 2.1. Quantum Inner Product Estimation

The main quantum speedups are gained in our algorithm by replacing the classical inner product $v_i^\top v_j$ of two vectors in $\mathbb{R}^d$ by its quantum estimate. To this end, we use the modified IPE algorithm, based on the work by Kerenidis et al. (2018). It achieves an asymptotic runtime of

$$\tilde{\mathcal{O}}\left(T\frac{\|v_i\|\|v_j\|}{\epsilon}\right), \tag{1}$$

where $T$ is the time to load the input vectors into a superposition quantum state, which becomes $T = \mathcal{O}(\text{polylog}(d))$ if we assume quantum random access memory (QRAM) or an equivalent quantum memory model (Kerenidis and Prakash, 2016). $\epsilon$ is an error bound on the inner product estimate, which in turn depends on the number of qubits $n$ used in the quantum phase estimation subroutine. $\tilde{\mathcal{O}}$ hides polylogarithmic factors.

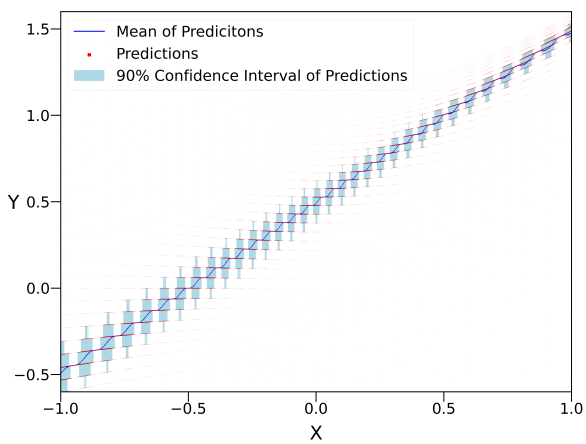
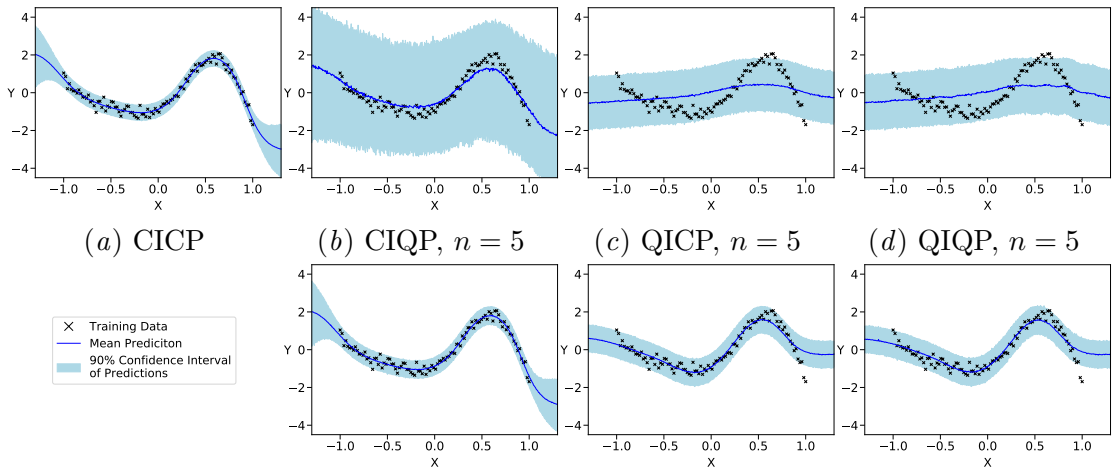



Figure 1: Noise pattern of the IPE quantum algorithm. The noise is non-isotropic and periodic.

Our modification for the usage in BNNs dispenses of median evaluation from the IPE algorithm. Instead of evaluating the inner product multiple times, we use a single estimate. Thus, the asymptotic runtime is reduced by a factor of $\log(1/\Delta)$, where $\Delta$ ensured a specific probability to attain the error $\epsilon$ in the inner product. Consequently, our IPE algorithm only has a constant probability of estimating the inner product within an error $\epsilon$.

#### 2.1.1. Quantum Noise in the Inner Product Estimation

The periodicity and non-isotropy of the noise seen in Figure 1 stems mostly from the phase estimation subroutine (see Section E.1). The representation of the inner products in the $n$ available qubits means there are only $2^n$ possible values available for the inner product estimate. Also, the IPE algorithm uses probabilistic subroutines to estimate the inner product. Thus, the best estimate in the $2^n$ possible values is not attained with certainty. For a more thorough treatment of the noise characteristics of the IPE algorithm, see Section E.




(a) CICP (b) CIQP, $n = 5$ (c) QICP, $n = 5$ (d) QIQP, $n = 5$

(e) Legend (f) CIQP, $n = 10$ (g) QICP, $n = 10$ (h) QIQP, $n = 10$

Figure 2: Linear Regression with BNN.

### 2.2. Quantum Inference Algorithm for Bayesian Neural Networks

Markov Chain Monte Carlo (MCMC) methods, such as stochastic gradient Langevin dynamics (SGLD), Hamiltonian Monte Carlo (HMC), and No-U-Turn Sampler (NUTS), are guided by gradients and thus they require backpropagation through the BNN. Current software relies on computing these gradients using automatic differentiation and Jacobian-vector products (JVPs). The JVP of an inner product contains two inner products (see Section C.2). These can be replaced with our IPE algorithm to compute estimates of the true gradient.

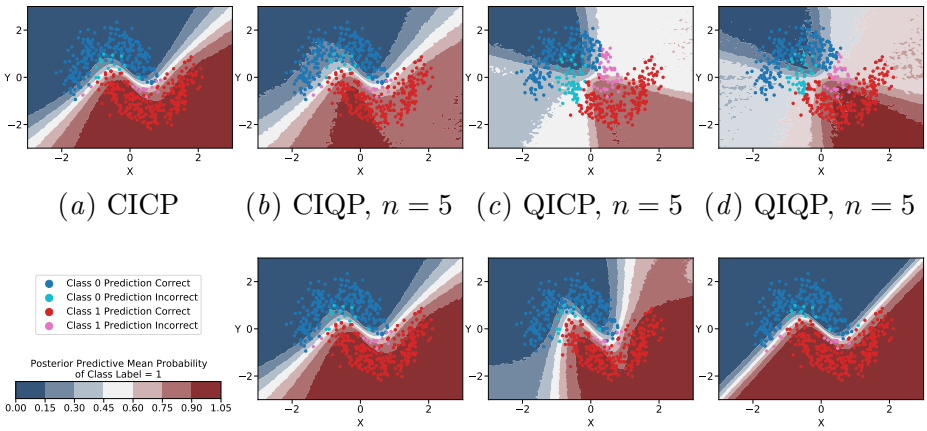
Our quantum inference algorithm differs only in, for the asymptotic runtime, negligible parts from the quantum training algorithm described by Allcock et al. (2020). For a single backpropagation, $\mathcal{O}(\Omega)$ inner products have to be calculated, where $\Omega$ is the number of neurons in the network. If we draw $K$ samples from our posterior and our training dataset $\mathcal{D}$ has a cardinality of $N = |\mathcal{D}|$, we need to calculate $KN$ backpropagations. Thus our quantum inference algorithm for BNNs has an asymptotic runtime of

$$\tilde{\mathcal{O}}\left((KN)^{1.5}\Omega\frac{1}{\epsilon}R\right), \tag{2}$$

where $R$ is a variable defined in Equation (4) and can be expected to be reasonably small for practical problems. The error component $\frac{1}{\epsilon}$ of the IPE depends on the number of qubits $n$ used in the phase estimation subroutine. For a small qubit number $n$, $\frac{1}{\epsilon}$ is also small. The additional factor of $\sqrt{KN}$ is a computational overhead of storing the weight matrices implicitly (see Section 2.2.1).

A classical inference would incur a runtime of $\mathcal{O}(KNP)$, where $P$ is the number of weights inside the neural network and for a fully-connected BNN is proportional to $\Omega^2$. The quantum algorithm has an advantage over the classical algorithm if $\sqrt{KN} \ll \Omega$, that is, for large networks.

#### 2.2.1. Low-Rank Initialization and Implicit Storage of Weight Matrices




$(a)$ CICP $(b)$ CIQP, $n = 5$ $(c)$ QICP, $n = 5$ $(d)$ QIQP, $n = 5$

$(e)$ Legend $(f)$ CIQP, $n = 10$ $(g)$ QICP, $n = 10$ $(h)$ QIQP, $n = 10$

Figure 3: Binary Classification with BNN.

After a backpropagation through the network, the updated weight matrices need to be stored. QRAM allows for fast load times into a quantum state, but storing is linear in the input size. Thus, if we were to store the weight matrices explicitly, we would incur a runtime of $\mathcal{O}(\Omega^2)$ per backpropagation. This would negate the speedup seen in Section 2.2. Allcock et al. (2020) propose to solve this problem via low-rank initialization and implicit storage of the weight matrices.

As a caveat, it should be noted that the low-rank initialization of the weight matrices needs to be compatible with the initialization using the prior $p(\theta)$. Moreover, the algorithm for implicit storage of the weight matrices is dependent on the inference algorithm and its feasibility needs to be evaluated on a case-by-case basis.

For our simulations, we operate in a full-rank prior regime. However, we also present simulation results that show that if we move to a low-rank prior regime, the results are still viable (see Appendix F). We do assume that the sampler allows for implicit storage of the weight matrices. It will however be an exciting direction for future work to study how these requirements could be relaxed.

### 2.3. Quantum Prediction Algorithm for Bayesian Neural Networks

The prediction algorithm presented in this paper follows the same outline as the evaluation algorithm in Allcock et al. (2020), but with the modified IPE introduced in Section 2.1. To get all predictions, the modified evaluation algorithm is executed $KM$ times, where $K$ is the number of weight samples drawn from the posterior and $M$ is the cardinality of the prediction dataset. The quantum prediction algorithm for a BNN has an asymptotic runtime of

$$\tilde{\mathcal{O}}\left(K^{1.5}\sqrt{N}M\Omega\frac{1}{\epsilon}R_e\right), \tag{3}$$

Here, the additional factor of $\sqrt{KN}$ is a consequence of storing the weight matrices implicitly during the quantum inference algorithm. If we were to use classical inference, this factor would disappear. Again, $R_e$ is a variable defined in Equation (5) and can be expected to be reasonably small for practical problems. The rest of the runtime analysis is analogous to the one in Section 2.2.

A classical algorithm for evaluating a BNN with the same inputs will have an asymptotic complexity of $\tilde{\mathcal{O}}(KMP)$. Similarly to the case of training, the quantum algorithm thus provides a speedup over the classical algorithm if $\sqrt{KM} \ll \Omega$, that is, for large networks. If classical inference is used, the speedup occurs unconditionally.

## 3. Results

We provide results for a linear regression task and a binary classification task. The BNN we use in both tasks has two hidden layers with five neurons each. The results are obtained using a simulation of the IPE algorithm on a classical computer. We vary the number of qubits ($n$) used in the phase estimation algorithm for the IPE procedure. We expect a higher accuracy on the inner product estimate for a larger number of qubits.

For both tasks, we compare the fully classical algorithm (i.e., classical inference and classical prediction, *CICP*), classical inference with quantum prediction (*CIQP*), quantum inference with classical prediction (*QICP*), and quantum inference with quantum prediction (*QIQP*). While the QIQP setting promises the largest speedups, the other settings can also be interesting in certain applications. For instance, CIQP could be used when a predictive model is only trained once (offline), but then used for prediction repeatedly in real-time (online).

### 3.1. Linear Regression

We see in Figure 2 that our expectation of greater precision for a higher qubit number $n$ holds. For $n = 10$ qubits, the results are already comparable to the completely classical case. We also notice that quantum prediction seems to be more resilient to fewer qubits than quantum inference. Specifically, if we compare the subfigures Figure 2($f$) and Figure 2($g$), the difference becomes apparent. Further results are shown in Figure 8 in the Appendix.

In Figure 5 in the Appendix, we see the difference between low-rank initialization (rank 3) and full-rank initialization (rank 5). We observe that the low-rank initialization still delivers acceptable results for 10 qubits when using quantum prediction. We also see that quantum inference is more susceptible to low-rank initialization. Increasing the qubit number likely makes the low-rank initialization almost equivalent to full-rank initialization because higher qubit number simulations approach the classical results. When looking at the completely classical simulations in Figure 5($a$) and Figure 5($e$), we notice that low- and full-rank initialization are comparable.

### 3.2. Binary Classification

The binary classification in Figure 3 confirms the results of the linear regression task. Again, there is greater precision with higher qubit numbers and quantum prediction is more resilient to fewer qubits than quantum inference. These observations also qualitatively hold for the

predictive uncertainties on this task, as shown in Figure 4 in the Appendix. Further results can be found Figure 9 and Figure 10.

Both Figure 6 and Figure 7 in the Appendix confirm the observations regarding low-rank initialization. Here too, the low-rank initialization seems to provide sufficient results, especially if one would increase the number of qubits for the phase estimation.

### 3.3. UCI Datasets

In Figure 11 (in Appendix F), we see the result of simulating the linear regression tasks in the four UCI datasets on our version of a quantum BNN. The datasets are split 20 times into different training and prediction sets. The mean of the log-likelihood and the standard error are then used to create the plots. We simulate the linear regression for five different qubit numbers and 2000 weight samples of the posterior.

Both the Boston dataset in Figure 11($a$) and the Concrete dataset in Figure 11($b$) show us expected results. The precision of the classical predictions increases with the number of hidden neurons. With the Boston dataset, the model stops improving after 10 hidden neurons, while the Concrete dataset shows improvements even after 20 hidden neurons per layer. As expected, the accuracy of the model increases with the number of qubits until it almost reaches classical prediction capabilities with 13 qubits.

For the Energy dataset in Figure 11($d$), the behavior seems unexpected. The quantum prediction with the smallest number of neurons performs best. A possible explanation for this behavior is that the accuracy in the classical predictions does not increase with a larger number of hidden layer neurons. This suggests that the larger models do not capture more features of the data, while still incurring more inner products. The larger number of inner product calculations might lead to a larger overall error of these models when using quantum IPE.

The Wine dataset in Figure 11($e$) seems to punish larger models even while using classical prediction. Thus it does not seem surprising that the quantum prediction also performs worse with the larger models.

## 4. Conclusion

We have shown that quantum deep learning techniques can be fruitfully combined with Bayesian neural networks. In contrast to the standard point estimation setting, when sampling from Bayesian posteriors, one can achieve high fidelity of the samples even without repeating the quantum computations, already at small numbers of qubits. The promised speedups of the quantum algorithms can thus be realized to their full extent in the Bayesian learning setting. In future work, it will be exciting to extend these studies to more realistic prediction tasks and potentially speed up the inference even further through the use of quantum MCMC techniques.

## References

Jonathan Allcock, Chang-Yu Hsieh, Iordanis Kerenidis, and Shengyu Zhang. Quantum algorithms for feedforward neural networks. *ACM Transactions on Quantum Computing*, 1(1):1–24, 2020.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*, 2015.

Gilles Brassard, Peter Hoyer, Michele Mosca, and Alain Tapp. Quantum amplitude amplification and estimation. *Contemporary Mathematics*, 305:53–74, 2002.

Marco Cerezo, Andrew Arrasmith, Ryan Babbush, Simon C Benjamin, Suguru Endo, Keisuke Fujii, Jarrod R McClean, Kosuke Mitarai, Xiao Yuan, Lukasz Cincio, et al. Variational quantum algorithms. *arXiv preprint arXiv:2012.09265*, 2020.

Kamil Ciosek, Vincent Fortuin, Ryota Tomioka, Katja Hofmann, and Richard Turner. Conservative uncertainty estimation by fitting prior networks. In *International Conference on Learning Representations*, 2019.

Francesco D'Angelo and Vincent Fortuin. Repulsive deep ensembles are bayesian. *arXiv preprint arXiv:2106.11642*, 2021.

Francesco D'Angelo, Vincent Fortuin, and Florian Wenzel. On stein variational neural network ensembles. *arXiv preprint arXiv:2106.10760*, 2021.

Erik Daxberger, Eric Nalisnick, James Urquhart Allingham, Javier Antorán, and José Miguel Hernández-Lobato. Expressive yet tractable Bayesian deep learning via subnetwork inference. *arXiv preprint arXiv:2010.14689*, 2020.

Michael W Dusenberry, Ghassen Jerfel, Yeming Wen, Yi-an Ma, Jasper Snoek, Katherine Heller, Balaji Lakshminarayanan, and Dustin Tran. Efficient and scalable Bayesian neural nets with rank-1 factors. *arXiv preprint arXiv:2005.07186*, 2020.

Vincent Fortuin. Priors in bayesian deep learning: A review. *arXiv preprint arXiv:2105.06868*, 2021.

Vincent Fortuin, Adrià Garriga-Alonso, Mark van der Wilk, and Laurence Aitchison. Bnnpriors: A library for bayesian neural network inference with different prior distributions. *Software Impacts*, page 100079, 2021a.

Vincent Fortuin, Adrià Garriga-Alonso, Florian Wenzel, Gunnar Rätsch, Richard Turner, Mark van der Wilk, and Laurence Aitchison. Bayesian neural network priors revisited. *arXiv preprint arXiv:2102.06571*, 2021b.

Adrià Garriga-Alonso and Vincent Fortuin. Exact langevin dynamics with stochastic gradients. *arXiv preprint arXiv:2102.01691*, 2021.

Aram W Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. *Physical review letters*, 103(15):150502, 2009.

José Miguel Hernández-Lobato and Ryan Adams. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *International Conference on Machine Learning*, pages 1861–1869. PMLR, 2015.

Matthew D Hoffman and Andrew Gelman. The no-u-turn sampler: adaptively setting path lengths in hamiltonian monte carlo. *J. Mach. Learn. Res.*, 15(1):1593–1623, 2014.

Xinyu Hu, Paul Szerlip, Theofanis Karaletsos, and Rohit Singh. Applying svgd to bayesian neural networks for cyclical time-series prediction and inference. *arXiv preprint arXiv:1901.05906*, 2019.

Alexander Immer, Matthias Bauer, Vincent Fortuin, Gunnar Rätsch, and Mohammad Emtiyaz Khan. Scalable marginal likelihood estimation for model selection in deep learning. *arXiv preprint arXiv:2104.04975*, 2021a.

Alexander Immer, Maciej Korzepa, and Matthias Bauer. Improving predictions of bayesian neural nets via local linearization. In *International Conference on Artificial Intelligence and Statistics*, pages 703–711. PMLR, 2021b.

Pavel Izmailov, Sharad Vikram, Matthew D Hoffman, and Andrew Gordon Wilson. What are bayesian neural network posteriors really like? *arXiv preprint arXiv:2104.14421*, 2021.

Iordanis Kerenidis and Anupam Prakash. Quantum recommendation systems. *arXiv preprint arXiv:1603.08675*, 2016.

Iordanis Kerenidis and Anupam Prakash. Quantum gradient descent for linear systems and least squares. *Physical Review A*, 101(2):022316, 2020.

Iordanis Kerenidis, Jonas Landman, Alessandro Luongo, and Anupam Prakash. q-means: A quantum algorithm for unsupervised machine learning, 2018.

Iordanis Kerenidis, Jonas Landman, and Anupam Prakash. Quantum algorithms for deep convolutional neural networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.

Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. Quantum principal component analysis. *Nature Physics*, 10(9):631–633, 2014.

David J.C. MacKay. A practical Bayesian framework for backpropagation networks. *Neural computation*, 4(3):448–472, 1992.

Ashley Montanaro. Quantum speedup of monte carlo methods. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 471(2181):20150301, 2015.

Radford M. Neal. Bayesian training of backpropagation networks by the Hybrid Monte Carlo method. Technical report, University of Toronto, 1992.

Radford M Neal et al. Mcmc using hamiltonian dynamics. *Handbook of markov chain monte carlo*, 2(11):2, 2011.

Michael A Nielsen and Isaac Chuang. Quantum computation and quantum information, 2002.

Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, David Sculley, Sebastian Nowozin, Joshua Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? Evaluating predictive uncertainty under dataset shift. In *Advances in Neural Information Processing Systems*, pages 13991–14002, 2019.

John Preskill. Quantum computing in the nisq era and beyond. *Quantum*, 2:79, 2018.

Jakub Swiatkowski, Kevin Roth, Bastiaan S Veeling, Linh Tran, Joshua V Dillon, Stephan Mandt, Jasper Snoek, Tim Salimans, Rodolphe Jenatton, and Sebastian Nowozin. The k-tied normal distribution: A compact parameterization of Gaussian mean field posteriors in Bayesian neural networks. *arXiv preprint arXiv:2002.02655*, 2020.

Ziyu Wang, Tongzheng Ren, Jun Zhu, and Bo Zhang. Function space particle optimization for bayesian neural networks. In *International Conference on Learning Representations*, 2018.

Max Welling and Yee Whye Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, pages 681–688, 2011.

Florian Wenzel, Kevin Roth, Bastiaan S Veeling, Jakub Świ atkowski, Linh Tran, Stephan Mandt, Jasper Snoek, Tim Salimans, Rodolphe Jenatton, and Sebastian Nowozin. How good is the Bayes posterior in deep neural networks really? In *International Conference on Machine Learning*, 2020.

Yusen Wu, Chao-hua Yu, Sujuan Qin, Qiaoyan Wen, and Fei Gao. Bayesian machine learning for boltzmann machine in quantum-enhanced feature spaces. *arXiv preprint arXiv:1912.10857*, 2019.

Zhikuan Zhao, Alejandro Pozas-Kerstjens, Patrick Rebentrost, and Peter Wittek. Bayesian deep learning on a quantum computer. *Quantum Machine Intelligence*, 1(1):41–51, 2019.

## Appendix A. Related Work

**Bayesian neural networks.** Bayesian neural networks (MacKay, 1992; Neal, 1992) have gained popularity recently (Wenzel et al., 2020; Fortuin et al., 2021a,b). They provide many benefits compared to their non-Bayesian counterparts, including calibrated uncertainties (Ovadia et al., 2019), principled inclusion of prior knowledge (Fortuin, 2021), and automatic model selection (Immer et al., 2021a). While there has been work on BNN inference using variational methods (Hernández-Lobato and Adams, 2015; Blundell et al., 2015; Swiatkowski et al., 2020; Dusenberry et al., 2020), Laplace approximation (Daxberger et al., 2020; Immer et al., 2021b), and particle-based methods (Wang et al., 2018; Hu et al., 2019; Ciosek et al., 2019; D'Angelo et al., 2021; D'Angelo and Fortuin, 2021), the gold-standard inference methods are still MCMC methods (Izmailov et al., 2021), such as SGLD (Welling and Teh, 2011), GG-MC (Garriga-Alonso and Fortuin, 2021), HMC (Neal et al., 2011), and NUTS (Hoffman and Gelman, 2014). We use NUTS sampling, but our algorithm is readily extensible to other inference settings.

**Quantum machine learning.** Quantum machine learning has gained a lot of interest in recent years, providing hope to enhance machine learning on a fault-tolerant universal quantum computer. It builds on several fundamental algorithms such as linear system solving (Harrow et al., 2009), optimization (Kerenidis and Prakash, 2020), recommendation systems (Kerenidis and Prakash, 2016), dimensionality reduction (Lloyd et al., 2014), and many more. The quantum algorithms for neural networks by Allcock et al. (2020) and Kerenidis et al. (2020) were inspired by Kerenidis et al. (2018), who defined a fast quantum inner product estimation algorithm. We also build on this algorithm in our work. Recently, quantum Bayesian machine learning such as Gaussian Processes (Zhao et al., 2019) or Boltzmann machines (Wu et al., 2019) have also been explored. Moreover, Quantum Monte Carlo algorithms, such as the ones by Montanaro (2015), could be an interesting direction for use in BNNs in future work.

## Appendix B. Source Code and Data

The source code and additional data from the simulations presented in this paper is on GitHub: `ANONYMIZED URL`.

## Appendix C. Method details

### C.1. $R$ Terms in Quantum Inference and Prediction Algorithm

The $R$ terms appearing in the runtime of both quantum algorithms are a new phenomenon, not observed in classical algorithms. The variable $R$ in Equation (2) is defined as

$$
\begin{aligned}
R &= R_a + R_\delta + R_W, \\
R_a &= \frac{1}{KN(\Omega - n_1)} \sum_{k,n} \sum_{\ell=2}^{L} \sum_{j=1}^{n_\ell} \|X^{[k,\ell,j]}\|_F \|a^{k,n,\ell-1}\|, \\
R_\delta &= \frac{1}{KN(\Omega - n_1)} \sum_{k,n} \sum_{\ell=1}^{L-1} \sum_{j=1}^{n_\ell} \|\tilde{X}^{[k,\ell+1,j]}\|_F \|\delta^{k,n,\ell+1}\|, \\
R_W &= \frac{1}{KN(\Omega - n_1)} \sum_{\ell=2}^{L} \sum_{j=1}^{n_\ell} \left( \frac{\|X^{[k,\ell,j]}\|_F}{\|W_j^{k,\ell}\|} + \frac{\|\tilde{X}^{[k,\ell,j]}\|_F}{\|(W^{k,\ell})_j^\top\|} \right),
\end{aligned}
\tag{4}
$$

where $n_\ell$ is the number of neurons in the $\ell$-th layer, the BNN consists of $L$ layers, the weight matrix $W^\ell$ is associated between layers $\ell - 1$ and $\ell$, $W_j^{k,\ell}$ is the $k$-th sample of the $j$-th row of the weight matrix $W^\ell$, $X^{[k,\ell,j]}$ is the implicitly stored version of $W_j^{k,\ell}$, $a^{k,n,\ell}$ is the output for the $n$-th datapoint of the $\ell$-th layer using the $k$-th weight sample, $\delta^{k,n,\ell}$ is the output for the $n$-th datapoint of the $\ell$-th layer using the $k$-th weight sample in the backward pass and $\|\|_F$ is the Frobenius norm.

The variable $R_e$ used in Equation (3) is defined as

$$
R_e = \frac{1}{(\Omega - n_1)} \sum_{\ell=2}^{L} \sum_{j=1}^{n_\ell} \|W_j^\ell\| \|a^{\ell-1}\|. \tag{5}
$$

As argued by Allcock et al. (2020), both these values are expected to be small for practical parameter regimes, which we also expect to hold for BNNs.

### C.2. Jacobian-Vector Product of an Inner Product

The JVP of an inner product contains itself two inner products between vectors:

$$
\nabla(v_i^\top v_j) \cdot (t_1, t_2) = v_j \cdot t_1 + v_i \cdot t_2, \tag{6}
$$

where $t_1$ and $t_2$ are the tangent vectors. In our simulation, we replace the exact calculation of these inner products with the estimate of the IPE routine. This gives us an estimate of the JVP (and thus of the gradient), instead of the true JVP value. This allows for a faster runtime of the backpropagation algorithm.

## Appendix D. Preliminaries in Quantum Computing

We present a succinct broad-audience quantum information background necessary for this work. See the book by Nielsen and Chuang (2002) for a detailed course.

**Qubits:** In classical computing, a bit can be either 0 or 1. From a quantum information perspective, a quantum bit or *qubit* can be in state $|0\rangle$ or $|1\rangle$. We use the *braket* notation $|\cdot\rangle$ to specify the quantum nature of the bit. The qubits can be in superposition of both states $\alpha|0\rangle + \beta|1\rangle$ where $\alpha, \beta \in \mathbb{C}$ such that $|\alpha|^2 + |\beta|^2 = 1$. The coefficients $\alpha$ and $\beta$ are

called *amplitudes*. The probabilities of observing either 0 or 1 when *measuring* the qubit are linked to the amplitudes:

$$p(0) = |\alpha|^2, \quad p(1) = |\beta|^2 \tag{7}$$

As quantum physics teaches us, any superposition is possible before the measurement, which gives special abilities in terms of computation. With $n$ qubits, $2^n$ possible binary combinations (e.g. $|01\cdots1001\rangle$) can exist simultaneously, each with its own amplitude.

A $n$ qubits system can be represented as a normalized vector in a $2^n$ dimensional Hilbert space. A multi-qubit system is called a quantum *register*. If $|p\rangle$ and $|q\rangle$ are two quantum states or quantum registers, the whole system can be represented as a tensor product $|p\rangle\otimes|q\rangle$, also written as $|p\rangle\,|q\rangle$ or $|p,q\rangle$.

**Quantum Computation:** As logical gates in classical circuits, qubits or quantum registers are processed using quantum gates. A quantum gate is a *unitary* mapping in the Hilbert space, preserving the unit norm of the quantum state vector. Therefore, a quantum gate acting on $n$ qubits is a matrix $U \in \mathbb{C}^{2^n}$ such that $UU^\dagger = U^\dagger U = I$, with $U^\dagger$ being the adjoint, or conjugate transpose, of $U$.

Common single qubit gates include the Hadamard gate $\frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}$ that maps $|0\rangle \mapsto \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and $|1\rangle \mapsto \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$, creating a quantum superposition, the NOT gate $\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}$ that permutes $|0\rangle$ and $|1\rangle$, or $R_y$ rotation gate parametrized by an angle $\theta$, given by $\begin{pmatrix}\cos(\theta/2) & -\sin(\theta/2)\\ \sin(\theta/2) & \cos(\theta/2)\end{pmatrix}$.

Common two-qubits gates include the CNOT gate $\begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 1 & 0\end{pmatrix}$ which is a NOT gate applied on the second qubit only if the first one is in state $|1\rangle$.

The main advantage of quantum gates is their ability to be applied to a superposition of inputs. Indeed, given a gate $U$ such that $U\,|x\rangle \mapsto |f(x)\rangle$, we can apply it to all possible combinations of $x$ at once $U(\frac{1}{C}\sum_x |x\rangle) \mapsto \frac{1}{C}\sum_x |f(x)\rangle$.

## Appendix E. Error Analysis of the Quantum Inner Product Estimation Algorithm

The inner product estimation routine takes as input two vectors as quantum states $|v_i\rangle$ and $|v_j\rangle$. It outputs an estimate of the inner product $\langle v_i|v_j\rangle$. The vectors are amplitude encoded, meaning that $|v_i\rangle$ is defined as

$$|v_i\rangle = \sum_{l=0}^{d} v_{i,l}\,|l\rangle\,, \tag{8}$$

where $v_{i,l}$ is the $l$-th element of the vector $v_i$ and for the dimension $d = 2^n$ has to hold, where $n$ is the number of qubits of state $|v_i\rangle$. Note, that amplitude encoding enforces

that the vectors $v_i$ and $v_j$ are normalized. Thus also for the IPE the input vectors are normalized. Their norms are stored during the quantum inference and prediction algorithms to unnormalize the inner product estimate after the IPE computation. This means that the error of the IPE is also multiplied by the norms $\|v_i\|$ and $\|v_j\|$.

IPE calculates the inner product estimate by preparing a state $|\psi\rangle$. $|\psi\rangle$ has an amplitude on one of the measurable basis states that is proportional to the inner product $\langle v_i|v_j\rangle$. It then uses amplitude estimation, which was introduced by Brassard et al. (2002), as a subroutine to compute the value of this amplitude. Amplitude estimation uses phase estimation as a subroutine, so we will first look at phase estimation and cascade the error backward.

### E.1. Phase Estimation

Phase estimation receives as input the following quantum state:

$$\frac{1}{\sqrt{2^n}} \sum_{y=0}^{2^n-1} e^{2\pi i \omega y} |y\rangle, \tag{9}$$

where $n$ is the number of qubits. The desired output is a good estimate of the phase parameter $\omega$.

In general, the output of the phase estimation algorithm will be a superposition

$$|\tilde{\omega}\rangle = \sum_x \alpha_x(\omega)|x\rangle \tag{10}$$

of all possible integer states $|x\rangle$, where $x \in \{0, \ldots, 2^n - 1\}$. We are interested in the amplitudes $|\alpha_x(\omega)|^2$ which define the output distribution of the phase estimation algorithm.

Let $b$ be the integer in the range 0 to $2^n - 1$ such that $\frac{b}{2^n} = 0.b_1 \ldots b_n$ is the best $n$ bit approximation of $\omega$ which is less than $\omega$. Then, the difference $\delta \equiv \omega - \frac{b}{2^n}$ satisfies $0 \leq \omega \leq 2^{-n}$. Applying the phase estimation algorithm, also known as the inverse Quantum Fourier Transform, yields the state

$$\frac{1}{2^n} \sum_{k,l=0}^{2^n-1} e^{\frac{-2\pi i k l}{2^n}} e^{2\pi i \omega k} |l\rangle \tag{11}$$

The amplitude $\alpha_l$ of the state $|(b+l)(\text{mod } 2^n)\rangle$ is

$$\begin{aligned} \alpha_l &= \frac{1}{2^n} \sum_{k=0}^{2^n-1} \left( e^{2\pi i(\omega - (b+l)/2^n)} \right)^k \\ &= \frac{1}{2^n} \left( \frac{1 - e^{2\pi i(2^n \omega - (b+l))}}{1 - e^{2\pi i(\omega - (b+l)/2^n)}} \right) \\ &= \frac{1}{2^n} \left( \frac{1 - e^{2\pi i(2^n \delta - l)}}{1 - e^{2\pi i(\delta - l/2^n)}} \right) \end{aligned} \tag{12}$$

Here, the second equality stems from the closed-form formula for the geometric series.

The probability to measure the integer $b + l \mod 2^n$ is

$$\begin{aligned} |\alpha_l|^2 &= \frac{1}{2^{2n}} \left| \left( \frac{1 - e^{2\pi i (2^n \delta - l)}}{1 - e^{2\pi i (\delta - l/2^n)}} \right) \right|^2 \\ &= \frac{1}{2^{2n}} \left| \frac{2\sin(\pi(2^n\delta - l))}{2\sin(\pi(\delta - l/2^n))} \right|^2 \\ &= \frac{1}{2^{2n}} \frac{\sin^2(\pi(2^n\delta - l))}{\sin^2(\pi(\delta - l/2^n))}, \end{aligned} \tag{13}$$

where we used the fact that $|1 - e^{2ix}|^2 = 4|\sin(x)|^2$. Note that the distribution in Equation (13) depends (through the variable $\delta$) on the phase $\omega$, which is the variable that the phase estimation algorithm is trying to estimate. This means that it will not be possible to predict the exact output distribution of the phase estimation algorithm, since that would require knowledge of $\omega$.

### E.2. Amplitude Estimation

The amplitude of the state $|\psi\rangle$ given by the IPE as an input to the amplitude estimation algorithm is

$$a = \frac{1}{2} \left( \frac{\langle v_i | v_j \rangle}{\|v_i\| \, \|v_j\|} + 1 \right) \tag{14}$$

Phase estimation can obtain an estimate of $\omega$ for an amplitude of the form $\sin^2(\pi\omega)$. Thus the phase we try to estimate using phase estimation is

$$\overline{\omega} = \frac{1}{\pi} \arcsin(\sqrt{a}) \tag{15}$$

Using the results from Section E.1, we know that the phase estimate will be

$$\omega_l = \frac{(b + l) \mod 2^n}{2^n}, l \in \{0, 1, \ldots, 2^n - 1\}, \tag{16}$$

where $b = \text{argmin}_i |\frac{i}{2^n} - \overline{\omega}|, i \in \{0, 1, \ldots, 2^n\}$, with a probability of

$$p_l = \frac{1}{2^{2n}} \frac{\sin^2(\pi(2^n\delta - l))}{\sin^2(\pi(\delta - l/2^n))}, \delta = \overline{\omega} - b/2^n. \tag{17}$$

The output of the amplitude estimation will be

$$a_l = \sin^2(\pi\omega_l) \tag{18}$$

and the final output of the IPE algorithm is

$$(2a_l - 1)\|v_i\|\|v_j\|. \tag{19}$$

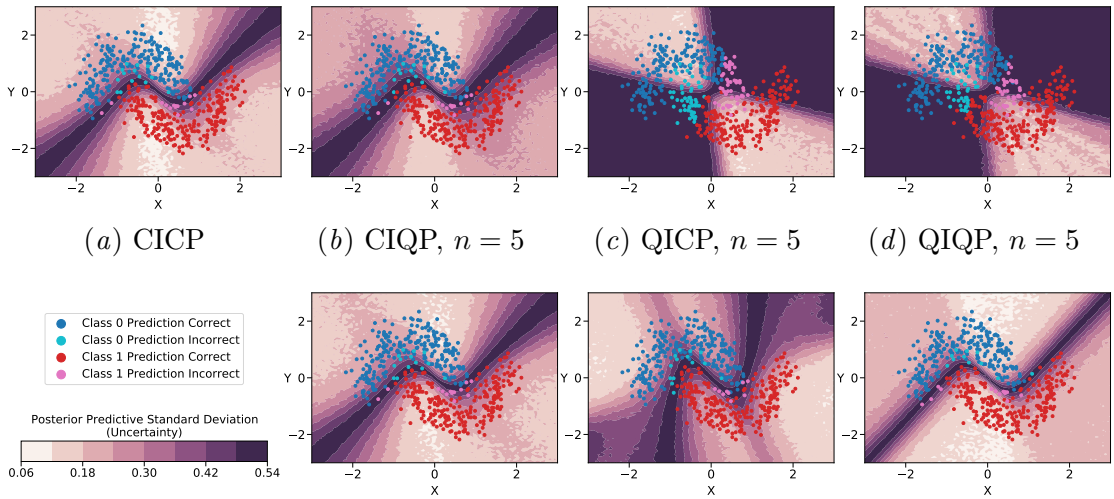
as an estimate of the inner product $\langle v_i | v_j \rangle$ between the vectors $v_i$ and $v_j$.

## Appendix F. Additional results

The additional results from the simulation of both the linear regression and binary classification task can be found in this section. Figure 4 shows the standard deviation of the BNN in the binary classification task.

Figure 5, Figure 6 and Figure 7 show a comparison between low-rank and full-rank initialization of the linear regression and binary classification task.

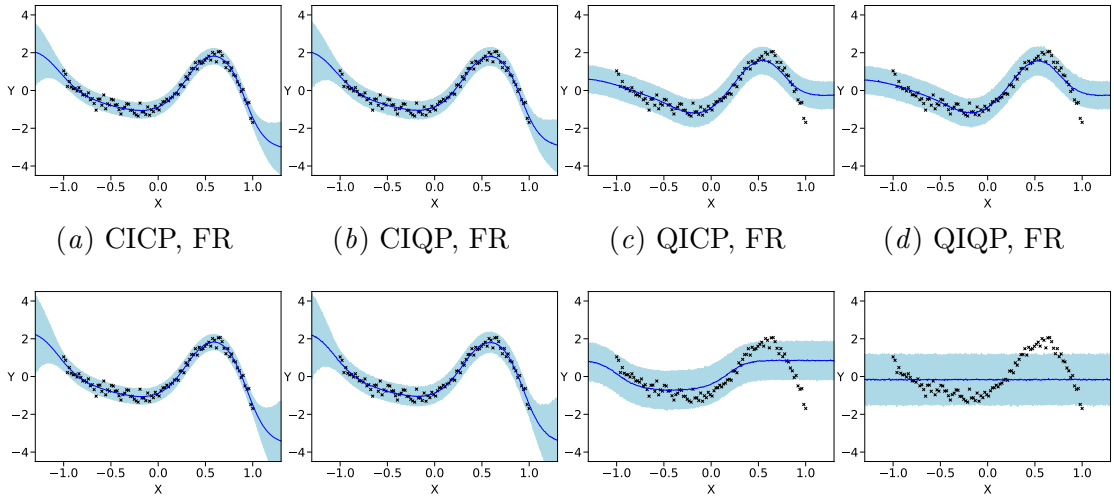
Figure 8 are further results for the linear regression task for a larger set of qubits. Figure 9 and Figure 10 show the mean prediction and standard deviation of the binary classification task respectively, both for a larger set of qubits.




$(a)$ CICP $(b)$ CIQP, $n = 5$ $(c)$ QICP, $n = 5$ $(d)$ QIQP, $n = 5$

$(e)$ Legend $(f)$ CIQP, $n = 10$ $(g)$ QICP, $n = 10$ $(h)$ QIQP, $n = 10$

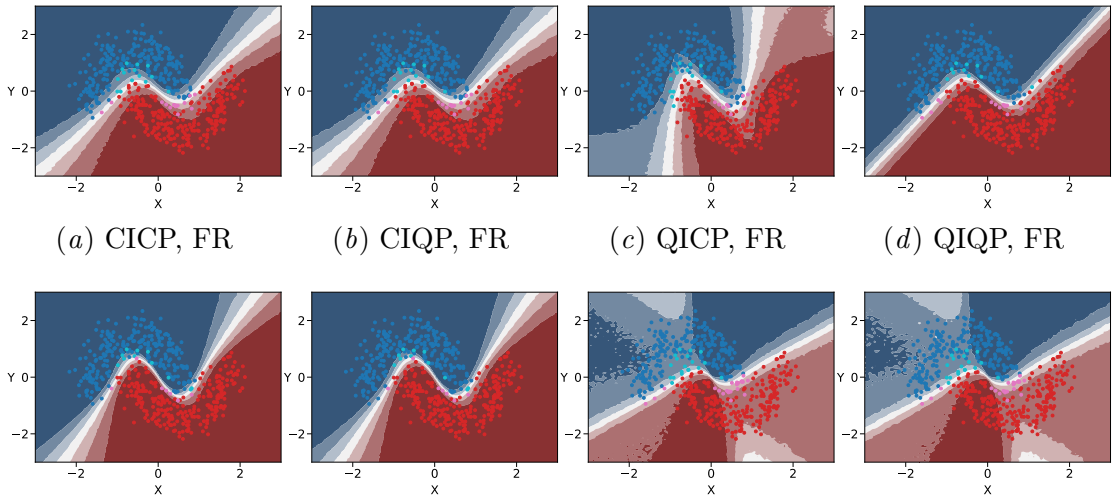
Figure 4: Posterior Predictive Standard Deviation of Binary Classification with BNN: $C$ and $Q$ stand for *Classical* and *Quantum* respectively. $I$ and $P$ stand for *Inference* and *Prediction*. The Figure shows the expected increase in accuracy for higher qubit numbers $n$.




(a) CICP, FR (b) CIQP, FR (c) QICP, FR (d) QIQP, FR

(e) CICP, LR (f) CIQP, LR (g) QICP, LR (h) QIQP, LR

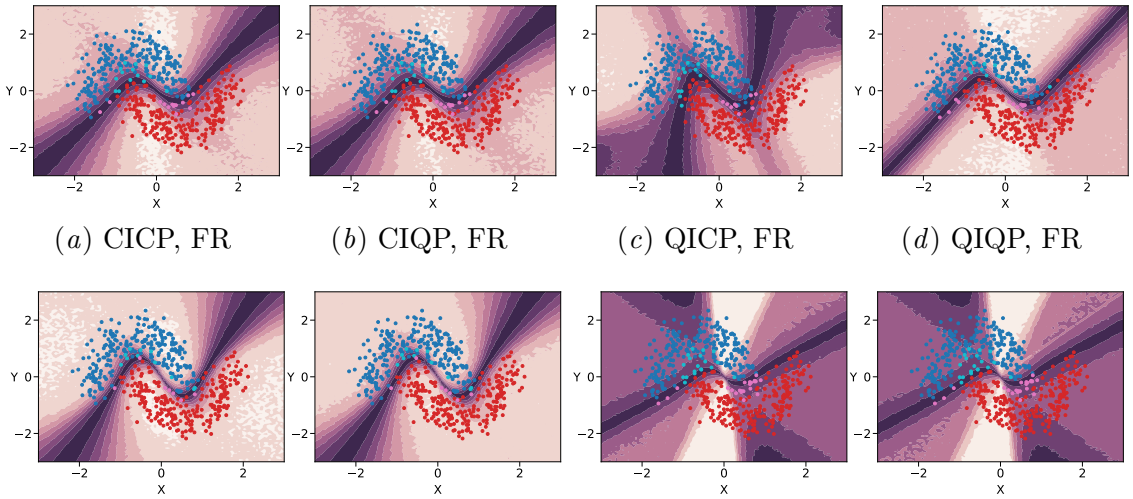
Figure 5: Comparison between Full-Rank Initialization (Rank 5) and Low-Rank Initialization (Rank 3) for the Linear Regression Task on a BNN for 10 qubits. FR stands for Full-Rank Initialization and LR stands for Low-Rank Initialization.




(a) CICP, FR (b) CIQP, FR (c) QICP, FR (d) QIQP, FR

(e) CICP, LR (f) CIQP, LR (g) QICP, LR (h) QIQP, LR

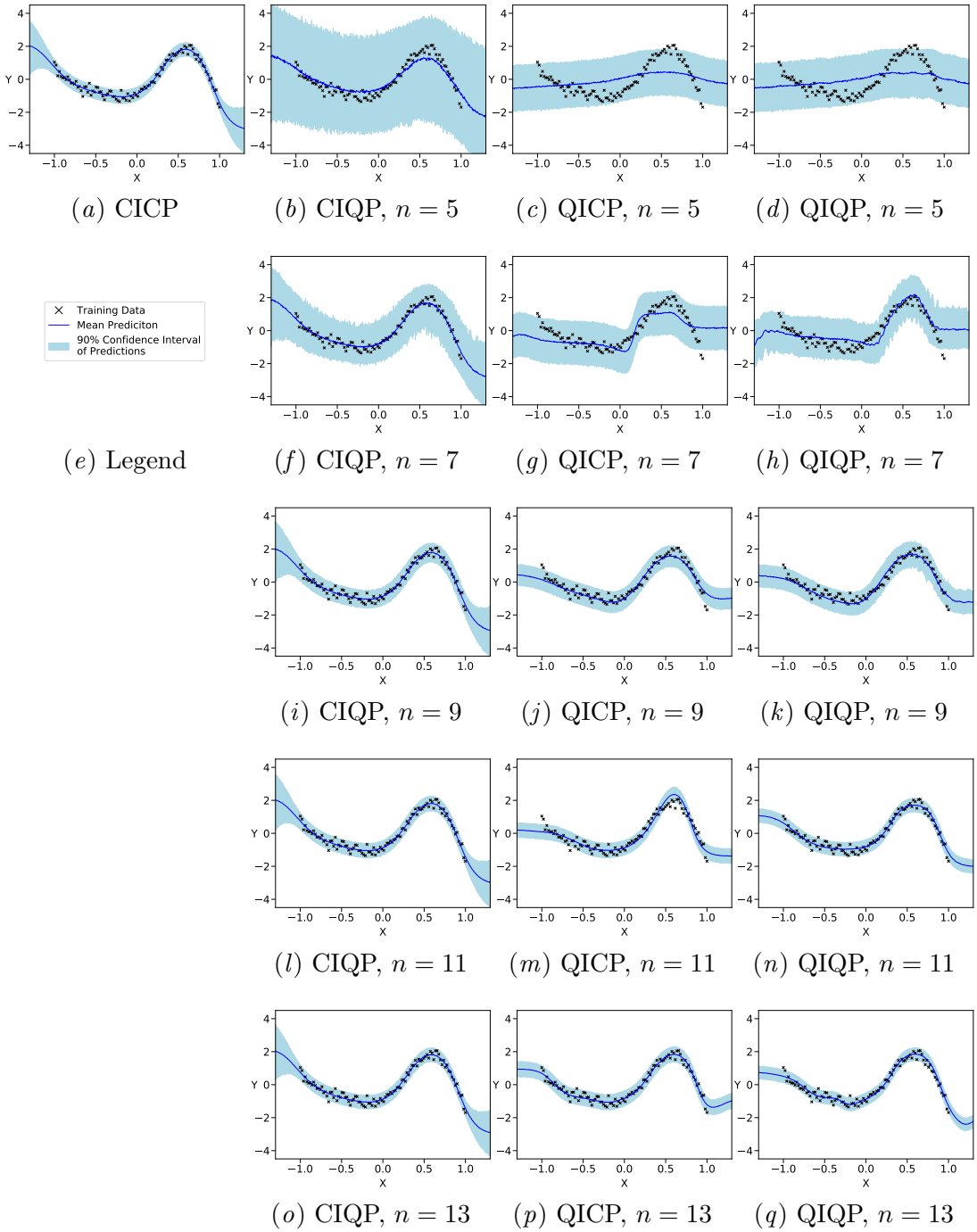
Figure 6: Comparison between Full-Rank Initialization (Rank 5) and Low-Rank Initialization (Rank 3) for the Binary Classification Task on a BNN for 10 qubits. FR stands for Full-Rank Initialization and LR stands for Low-Rank Initialization.




(*a*) CICP, FR (*b*) CIQP, FR (*c*) QICP, FR (*d*) QIQP, FR

(*e*) CICP, LR (*f*) CIQP, LR (*g*) QICP, LR (*h*) QIQP, LR

Figure 7: Comparison between Full-Rank Initialization (Rank 5) and Low-Rank Initialization (Rank 3) for the Posterior Predictive Standard Deviation of Binary Classification with BNN for 10 qubits. FR stands for Full-Rank Initialization and LR stands for Low-Rank Initialization.




(a) CICP (b) CIQP, $n = 5$ (c) QICP, $n = 5$ (d) QIQP, $n = 5$

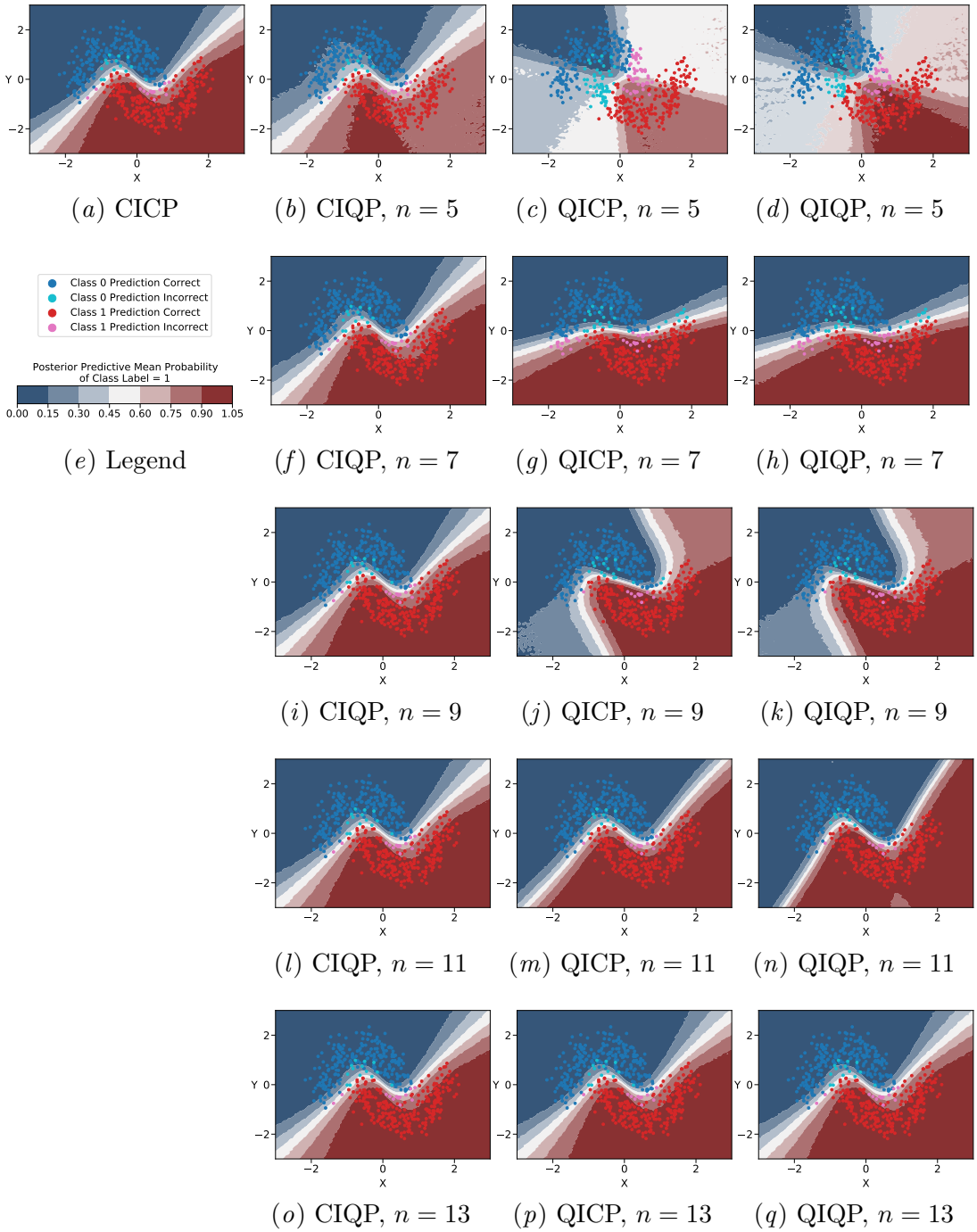
(e) Legend (f) CIQP, $n = 7$ (g) QICP, $n = 7$ (h) QIQP, $n = 7$

(i) CIQP, $n = 9$ (j) QICP, $n = 9$ (k) QIQP, $n = 9$

(l) CIQP, $n = 11$ (m) QICP, $n = 11$ (n) QIQP, $n = 11$

(o) CIQP, $n = 13$ (p) QICP, $n = 13$ (q) QIQP, $n = 13$

Figure 8: Additional Results for Linear Regression with BNN: $C$ and $Q$ stand for *Classical* and *Quantum* respectively. $I$ and $P$ stand for *Inference* and *Prediction*. The Figure shows the expected increase in accuracy for higher qubit numbers $n$.




(a) CICP (b) CIQP, $n = 5$ (c) QICP, $n = 5$ (d) QIQP, $n = 5$

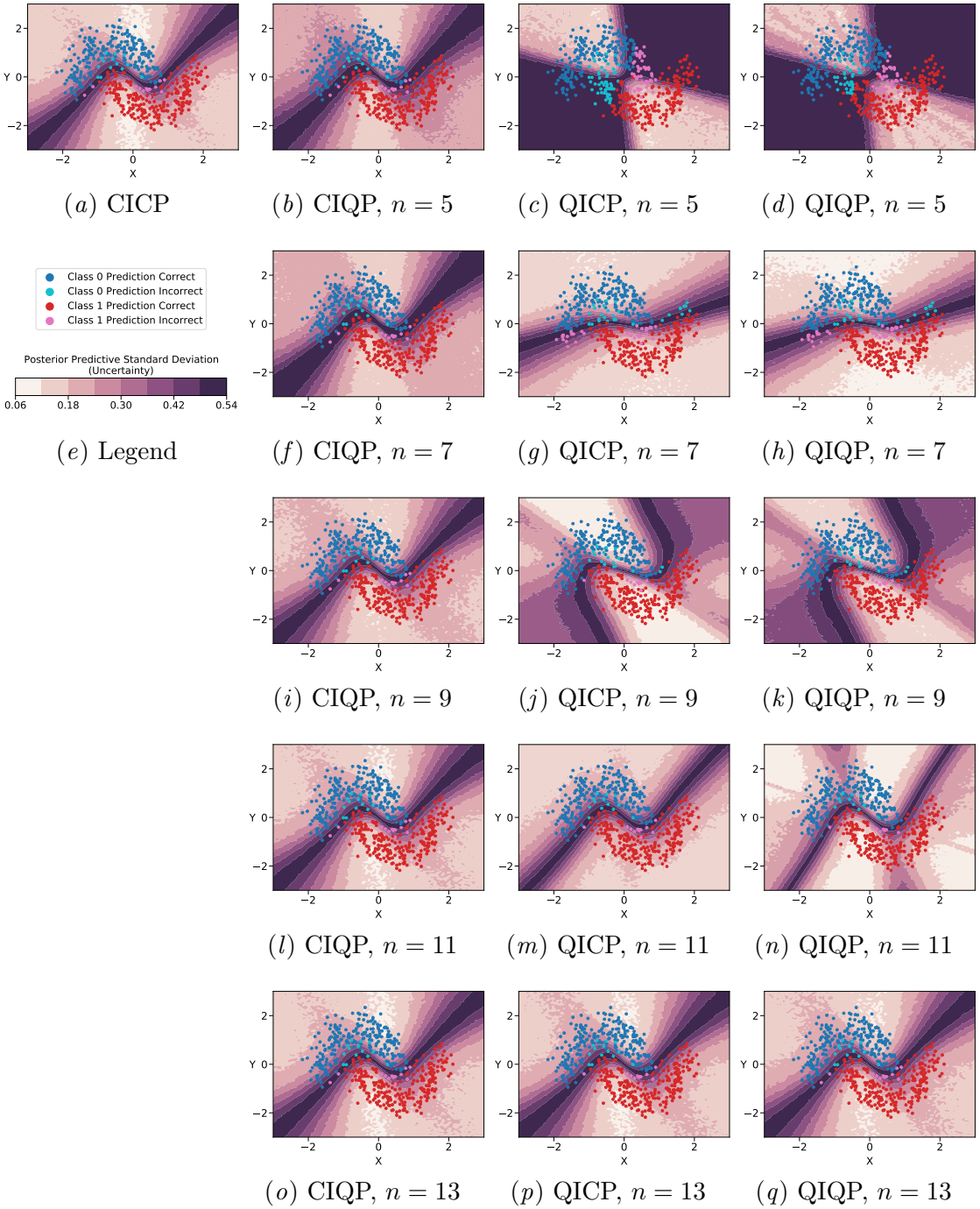
(e) Legend (f) CIQP, $n = 7$ (g) QICP, $n = 7$ (h) QIQP, $n = 7$

(i) CIQP, $n = 9$ (j) QICP, $n = 9$ (k) QIQP, $n = 9$

(l) CIQP, $n = 11$ (m) QICP, $n = 11$ (n) QIQP, $n = 11$

(o) CIQP, $n = 13$ (p) QICP, $n = 13$ (q) QIQP, $n = 13$

Figure 9: Additional Results for Binary Classification with BNN: $C$ and $Q$ stand for *Classical* and *Quantum* respectively. $I$ and $P$ stand for *Inference* and *Prediction*. The Figure shows the expected increase in accuracy for higher qubit numbers $n$.




(a) CICP (b) CIQP, $n = 5$ (c) QICP, $n = 5$ (d) QIQP, $n = 5$

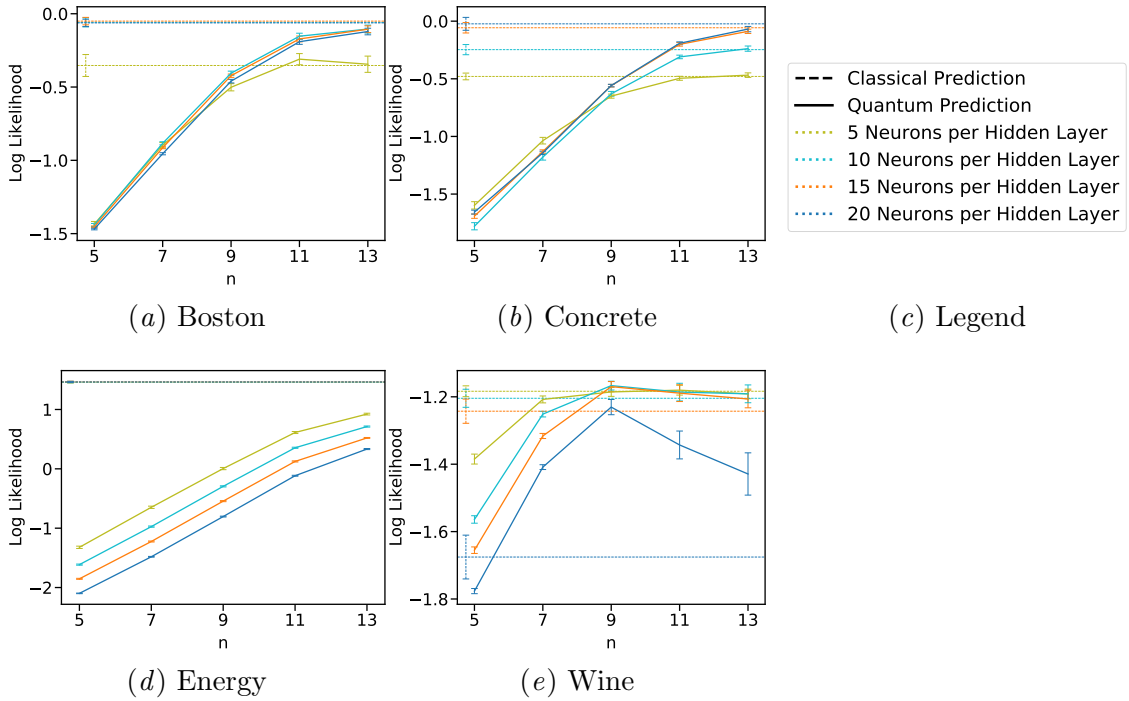
(e) Legend (f) CIQP, $n = 7$ (g) QICP, $n = 7$ (h) QIQP, $n = 7$

(i) CIQP, $n = 9$ (j) QICP, $n = 9$ (k) QIQP, $n = 9$

(l) CIQP, $n = 11$ (m) QICP, $n = 11$ (n) QIQP, $n = 11$

(o) CIQP, $n = 13$ (p) QICP, $n = 13$ (q) QIQP, $n = 13$

Figure 10: Additional Results for Posterior Predictive Standard Deviation of Binary Classification with BNN: $C$ and $Q$ stand for *Classical* and *Quantum* respectively. $I$ and $P$ stand for *Inference* and *Prediction*. The Figure shows the expected increase in accuracy for higher qubit numbers $n$.




$(a)$ Boston $(b)$ Concrete $(c)$ Legend

$(d)$ Energy $(e)$ Wine

Figure 11: UCI Data Regression with a BNN Using Quantum IPE. It shows the log-likelihood of classical prediction and quantum prediction for different qubit numbers $n$. Note that the y-axis is at different scales in the subplots.