QardEst: Using Quantum Machine Learning for Cardinality Estimation of Join Queries

Florian Kittelmann
Zurich University of Applied Sciences
Switzerland

Pavel Sulimov
Zurich University of Applied Sciences
Switzerland

Kurt Stockinger
Zurich University of Applied Sciences
Switzerland

ABSTRACT

Classical and learned query optimizers (LQOs) use cardinality estimations as one of the critical inputs for query planning. Thus, accurately predicting the cardinality of arbitrary queries plays a vital role in query optimization. A recent boom in novel deep learning methods stimulated not only the rise of LQOs but also contributed to the appearance of learned cardinality estimators (LCEs). However, the majority of them are based on classical neural networks, ignoring that multivariate correlations between attributes across different tables could be naturally represented via entanglements in quantum circuits.

In this paper, we introduce *QardEst - Quantum Cardinality Estimator* - a novel *quantum neural network approach* to estimate the cardinality of join queries. Our experiments conducted with a *similar number of trainable parameters* suggest that quantum neural networks executed on a quantum simulator outperform classical neural networks in terms of mean squared error as well as the q-error.

ACM Reference Format:
Florian Kittelmann, Pavel Sulimov, and Kurt Stockinger. 2024. QardEst: Using Quantum Machine Learning for Cardinality Estimation of Join Queries. In *Workshop on Quantum Computing and Quantum-Inspired Technology for Data-Intensive Systems and Applications (Q-Data '24), June 9, 2024, Santiago, AA, Chile.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3665225.3665444

1 INTRODUCTION

Cardinality estimation algorithms predict the returned numbers of rows, i.e. the result set size, of a database query without executing it. The query optimizer then uses this information about how selective different filters or joins are to choose the type of scans (e.g. index or sequential ones) or the type and order of joins for selecting an optimal query plan. Hence, the quality of cardinality estimation has a critical impact on the performance of the query optimizer [22, 23, 48].

The majority of relational database management systems use classical cardinality estimation methods. To get an understanding of single attribute "behavior" under different filters, they sample a portion of the data from a database and, based on it, evaluate either statistical characteristics like histograms of distributions [7, 9, 14, 30, 41, 47] (used, e.g. in PostgreSQL [1] and SQLServer [49]), or the number of pages of certain indexes [16, 19, 24, 25, 56] (used, e.g. in MySQL [6] and MariaDB [5]). In order to enable more complex cardinality measurements in the case of multiple attributes of the same table, additional calculations are required, e.g. calculations of functional dependencies between attributes, multivariate most-common value lists, and multivariate distinct counts [2].

Although the aforementioned methods already suffer from imperfect sampling misestimations and clusterization of single attribute errors, evaluation difficulties become even more severe when it comes to cardinality prediction for N-table joins. The problem is that, e.g., having N tables and M attributes in each table, requires to learn $O(M^N)$ correlations - so it is both about learning complicated joint distributions and computational complexity. Learned cardinalities estimators (LCEs) are exactly attempting to mitigate these issues with the help of machine learning methods. Although deep learning seems to give significant improvement compared to classical methods on datasets with more complicated data distributions and join schemas, it still requires *hundreds of millions of learnable parameters that are often hard to tune* [15].

However, quantum neural networks (QNNs) not only perform as universal function approximators, i.e., they can learn arbitrary joint non-linear correlations, the same as classical neural networks [50] but *using fewer learnable parameters*. Moreover, due to the phenomena of quantum entanglement [11], QNNs can do it with smaller amounts of trainable parameters. So, in our approach, we take the recent state-of-the-art architectures of neural networks used for cardinality estimation and replace a part of it with a quantum circuit.

The contributions of this paper are as follows:

- We develop a novel quantum neural network called *QardEst* for cardinality estimation based on an existing classical architecture, namely the multi-set convolutional network (MSCN) [21].
- We perform a detailed performance comparison of QardEst against a classical neural network baseline and show that when using a *similar number of parameters, the quantum approach executed on a quantum simulator outperforms the classical baseline.*

The source code of this paper is publicly available at: https://drive.google.com/file/d/1sqrIwCAKt2SRTCJjV-kmJ-zdbMDTLlh5/view?usp=drive_link

2 RELATED WORK

Quantum computing has recently gained traction due to the rapid development of quantum hardware [8, 36]. Typical applications in quantum computing are currently in optimization [29], machine learning [44] or in specific domains such as material science or chemistry [28].

Q-Data '24, June 9, 2024, Santiago, AA, Chile

ACM ISBN 979-8-4007-0553-3/24/06
https://doi.org/10.1145/3665225.3665444

The cardinality estimation problem for database queries has recently been tackled using mainly classical machine learning algorithms [20]. While the results of these methods are more accurate than traditional non-machine learning-based approaches, they still suffer from high training and inference costs while learning multivariate correlations [15, 48], and can mitigate these "side effects" only when using single-table statistics as an input [51]. This is still an unsolved problem, and therefore, it is uncertain if such models will prevail in real databases, especially in dynamic environments [48]. In general, foregoing approaches to learning the cardinalities could be split into query-driven [10, 21, 53] and data-driven [17, 52, 54, 55, 57] groups. While query-driven approaches learn the cardinalities directly from the SQL queries, data-driven approaches use the data in the database to learn and estimate the cardinalities [15].

Applications of quantum computing for database problems are currently arising more and more, for instance, in multiple query optimization [12, 45], as well as join ordering (e.g. using novel quantum-inspired encoding [40] or digital annealing [39]). In the sub-area of cardinality estimation, the usage of quantum technologies is meanwhile tailored to the quantum natural language processing model, predicting the query execution time and cardinality straight from SQL [46]. This approach frames cardinality estimation as a multi-class classification problem, where cardinality estimates are within 2 or 4 ranges. In contrast, our approach treats cardinality estimation as a regression problem and thus allowing more fine-grained estimates.

In our optics, an intuitive approach to solving the cardinality estimation problem using quantum computing could be to learn the histograms of the data in database tables using a quantum generative adversarial network (qGAN) [58]. Unfortunately, the advantages compared to traditional histograms and sampling methods are small until nonexistent because training a qGAN is computationally expensive.

Thus, instead of using qGANs, we suggest designing a hybrid quantum neural network approach that leverages classical neural networks and only implements part of the neural network using quantum circuits while still taking the quantum advantage. In one of the most recent LCE papers [35], it was shown how embedding machine learning heuristics into one of the fundamental LCE methods based on MSCN [21] can give a significant increase in accuracy under training procedure constraints. For our QardEst approach, we stick to using the same MSCN architecture as a baseline but extend it with novel improvements by injecting quantum elements.

3 LEARNED CARDINALITY ESTIMATION BASELINE

In this paper we compare our novel quantum cardinality estimation approach against a classical neural network baseline that is both well-understood and performs well in different cardinality estimation benchmarks. This classical baseline is called MSCN [21] - Multi-Set Convolutional Neural Network. We have chosen this approach since it is considered the state-of-the-art technique for learned query-driven cardinality estimation. Moreover, the input encoding can be straightforwardly applied also for our quantum approach with a limited number of qubits. Finally, we propose a few optimizations on the input encoding of MSCN.

We will now explain the MSCN approach in more detail.

3.1 Query and Data Encoding

Let us first discuss how to encode a query given a specific database such that a neural network can be trained for cardinality estimation. The easiest way would be to use *one-hot encoding* where each table and column that is part of a query is set to 1 - and to 0 if these tables and columns are not part of the query. The size of the encoding vector thus corresponds to the total number of tables and columns of the underlying database. In addition, the information about the distribution of the underlying data needs to be encoded along with specific query details such as joins, filters, etc.

Figure 1 shows how a specific SQL query is encoded in MSCN [21] using our own optimization. In particular, we removed the redundant table set information T_q to make the encoding more compact[1].

Now, let us analyze the encoding in more detail. Figure 1 (a) shows the query to be encoded. Figure 1 (b) shows the *join set* J_q to describe which attribute takes part in a join of the query. J_q uses a one-hot vector for the first join attribute, a separate one-hot vector for the second join attribute, etc. In our example we assume a database with 18 primary/foreign key combinations and thus 18 possible joins. Here, the attribute c.UserID (1/18) is joined with the attribute b.UserID (3/18).

Figure 1 (c) shows how the *predicate set* P_q is encoded for a database with 43 columns. P_q contains a one-hot vector for each attribute to be encoded and a one-hot vector for each of the 5 possible operators. Finally, the predicate values are encoded using simple normalization as shown in Equation 1

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{1}$$

where x is a value of the respective attribute.

3.2 Classical Neural Network

The neural network architecture of MSCN is shown in Figure 2. At the bottom we see a module for the *Join Set* as well as a module for the *Predicate Set*. At the top we see the output layer. The neural networks for the join module and the predicate module are applied for each table that is part of the join and each predicate that is part of the query. Afterwards the two neural networks are averaged, concatenated and fed to the output neural network.

An example of how the MSCN processes a query is given in Figure 1.

4 QUANTUM CARDINALITY ESTIMATION

In this section, we first describe some foundations of quantum computing. Afterwards, we introduce our novel approach to quantum cardinality estimation.

[1] Since the original join set J_q contains already information about the used tables of SQL, our more compact encoding does not lose any information.

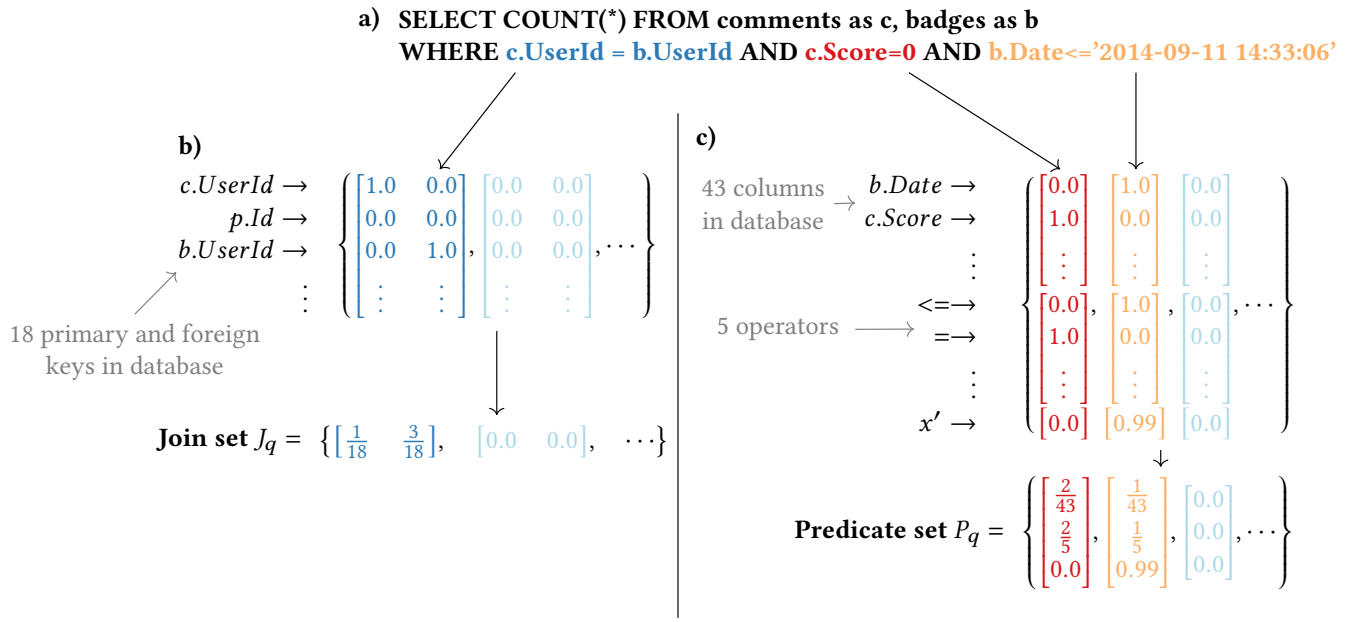

Figure 1: Query and data encoding as input for a neural network to estimate the cardinality of a query. a) SQL query to be encoded. b) Join set J_q with zero-padding encoded as a vector. The size of the vector corresponds to the number of possible PK/FK joins in the database. The light blue matrixes refer to potential PK/FK-joins that are not used in the example query. c) Predicate set P_q encoded as 3 vectors to represent (1) the query attribute, (2) the query operator and (3) the value of the query attribute. The light blue matrixes refer to attributes not used in the example queries.

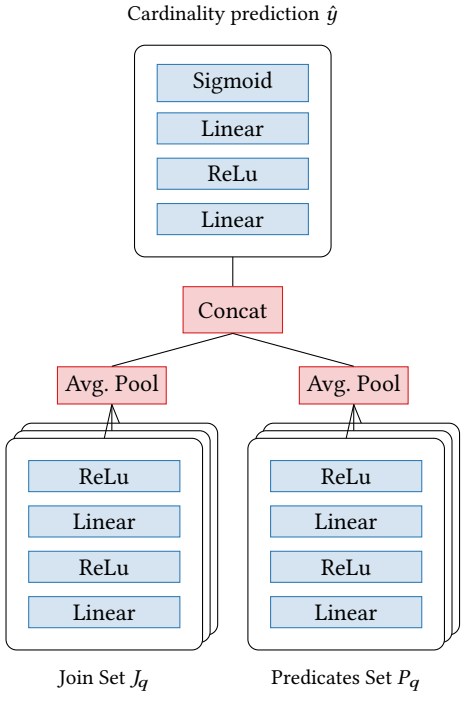

Figure 2: Multi-set convolutional network model architecture as defined in [21] with the absence of the table set.

4.1 Quantum Computing Foundations

Quantum computing is based on physical effects described in quantum mechanics, and the research area already has produced some promising quantum algorithms with significant speed-up compared to classical computers (e.g. the factorization of prime numbers [43]).

Superposition and *entanglement* are the main physical effects, which are used in quantum computing. They bring completely different characteristics to the development of algorithms.

Qubits are used as the smallest computing unit, which can be visualized in a Bloch sphere as shown in Figure 3. The vector showing to the north pool is state $|1\rangle$, while state $|0\rangle$ is shown on the south pool. All the other parts of the Bloch sphere are states in *superposition* of $|0\rangle$ and $|1\rangle$.

By rotations about the x-, y- and z-axis, the qubit can be put in another state. When a measurement is executed, the quantum state is destroyed and collapses to either a 0 or 1. If the quantum algorithm is executed multiple times, a probability distribution $P(x)$ of the states $|0\rangle$ and $|1\rangle$ can be observed, which is physically described by the square of the quantum state $|\psi\rangle$ as shown in Equation 2.

$$P(x) = |\psi\rangle^2 \tag{2}$$

One execution of the algorithm is typically called *shot*, and therefore, the number of *shots* needs to be configured high enough to obtain a good quality of the probability distribution $P(x)$. In quantum mechanics, the probability distribution $P(x)$ describes the probability of finding the particle (e.g. photon or electron) in a certain position. Therefore, the probability characteristics in quantum

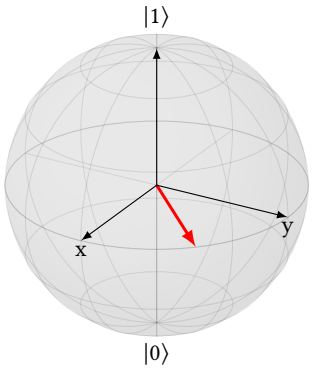

Figure 3: Bloch sphere of a single qubit in a superposition of the states $|0\rangle$ and $|1\rangle$. Because the vector in red points to the equator of the Bloch sphere, the probability is 50% to receive a resulting 1 and 50% to receive a resulting 0 on a measurement of the qubit.

computing are derived from the wave-particle duality in quantum mechanics.

Entanglement is the second essential physical effect, which is used in quantum computing. When one of the two entangled qubits is measured, the result of the measurement of the second qubit is known without needing to measure the second qubit. They are entangled, and therefore, colloquially speaking, they share information. Entangled qubits can not be visualized anymore in a Bloch sphere.

4.2 Quantum Neural Network - High Level View

The architecture of our quantum approach QCardEst has the same principal structure as the classical MSCN described in Section 3.2. However, the classical neural network layers are replaced with quantum neural networks (see Figure 4).

Let us now analyze the specifics of the quantum neural network architecture from bottom to top. The inputs of the quantum circuits are rescaled to the range of $[0.0, 2 \cdot \pi - 0.05]$ to cover the whole Bloch sphere as shown in Figure 4. Afterwards, the quantum circuits are executed. We will describe the internals of these circuits in Section 4.3.

Next, we measure the probability distribution $P(x)$ of the quantum circuits using 100 shots. The output of the 100 shots for the Join Set as well as the Predicate Set are combined using average pooling (Avg. Pool). Finally, the results of the Join Set are concatenated (Concat) with the results of the Predicate Set and fed to the output quantum neural network. The input values for the output quantum neural network are therefore again rescaled to the range of $[0.0, 2 \cdot \pi - 0.05]$ and its output is then fed to one single perceptron followed by a sigmoid function to obtain values in the range of $[0.0, 1.0]$.

4.3 Quantum Circuits - Detailed View

Now, let us analyze the quantum circuits in more detail. The parameterized quantum circuits (PQC) of the quantum neural network are shown in Figure 5. The first circuit encodes the Join Set and uses

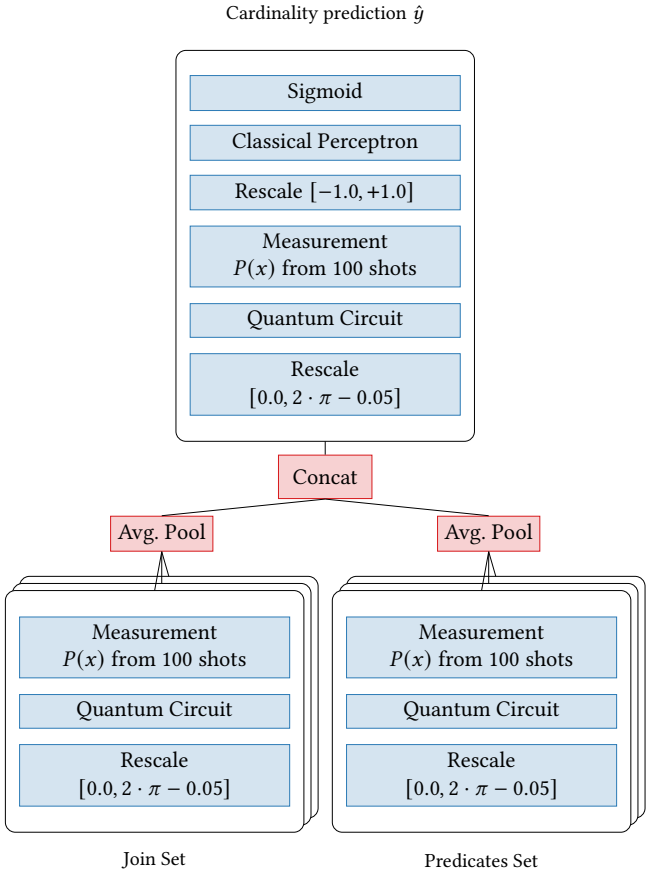

Figure 4: Multi-set convolutional network architecture for the quantum neural network.

angle encoding of the parameters x_i followed by rotation blocks with trainable parameters θ_i. To represent the analogous of multiple layers in a neural network, we have chosen the parameters k and l to vary the depth of the quantum circuit. By performing multiple repetitions with the parameter l, the input part of the quantum circuit gets applied multiple times inside the circuit with the same inputs x_i, while the trainable parameters θ_i are not shared when the parameter k is increased.

The second circuit uses angle encoding to encode the Predicate Set by two R_z rotation blocks, followed by a controlled R_y gate. We designed the trainable part with the parameters θ_i, l and k equal to the first circuit. On both circuits, the measurement is executed on both qubits, which results in two probability distributions of 4 states.

In the last circuit, the architecture of the output layer is shown, which uses 3 qubits to encode the concatenated probability distribution values using 8 rotation blocks. The trainable part of the output layer uses full entanglement.

In general, this design of the circuits was chosen, because it seemed to be an ideal approach to encode the whole data while the numbers of qubits can be kept small. This results in a feasible training time of about 10 days for 500 epochs using the parameters

$l = 4$ and $k = 2$. We discuss more details in the experiment section. Note that we avoided using an R_x gate after a Hadmard gate because it does not encode the data into the Bloch sphere. Therefore, we chose R_z rotation blocks in the input encoding because this ensures the data is encoded in the phase of the qubit.

4.4 Discussion about Nonlinearity and Output Encoding

For fulfilling the Universal Approximation Theorem [18], nonlinearities play a major role in approximating complex functions. While classical neural networks use activation functions to add nonlinearities, the best approach in quantum neural networks remains unclear. Because recent papers propose a redundancy of the input encoding in the circuits [13, 26, 38] for a good approximation of functions, the parameter l was introduced to repeat the input layer of the circuit.

Regarding output encoding, quantum circuits are predestined to solve classification tasks because of their output as a probability distribution. Unfortunately, this output brings challenges for solving regression tasks by e.g. using the expected value $E[x]$ of the probability distribution $P(x)$. The reason is that on a random initialized PQC the chances in a Bloch sphere are quite small to predict exactly the value 0.0 or 1.0. In our experiments, we observed that values close to 0.0 and 1.0 were somewhat challenging to reach for a quantum neural network. Therefore, we decided to use values from the probability distribution $P(x)$ and input them in a single perceptron to set aside this restriction.

5 EXPERIMENTS

We now describe the experiments for evaluating the effectiveness of QCardEst - our novel quantum cardinality estimation algorithm. The main research questions are (1) to find out if this *quantum algorithm can outperform* the classical counterpart and (2) to see how the *performance per trainable parameter* compares to the classical neural network.

5.1 Experimental Setup

Dataset: For all our experiments we use the STATS-CEB dataset [15] - a commonly used benchmark for evaluating cardinality estimation algorithms. Because the query file of the STATS-CEB dataset did not contain any cardinality information, we imported the data of the STATS-CEB database into our local MySQL database and executed all SQL queries to retrieve their cardinalities. The final dataset contains 2,627 SQL queries and their cardinalities without duplicated SQL queries.

Because the cardinalities of the SQL queries in the STATS-CEB dataset contain outliers, the labels were rescaled (y_r) by a logarithm as shown in Equation 3:

$$y_r = log_{10}(C^T) \tag{3}$$

In other words, we normalized the true cardinalities C^T as shown in Equation 5 by y_{max} (see Equation 4), which is the highest rescaled cardinality y_r in the dataset.

$$y_{max} = \max(y_r) = 11.05 \tag{4}$$

$$y = \frac{y_r}{y_{max}} \tag{5}$$

The value $y_{max} = 11.05$ was observed in the dataset and is used to calculate the estimated cardinality C^E from the neural network prediction $\hat{y}$ after inference (see Equation 6).

$$C^E = 10^{\hat{y} \cdot y_{max}} = 10^{11.05 \cdot \hat{y}} \tag{6}$$

The advantage of choosing logarithmic rescaling is that high cardinality outliers are more compressed while patterns in small cardinalities are preserved. However, a disadvantage is that when predicting the result set size of SQL queries with high cardinalities, the error in the neural network prediction $\hat{y}$ leads to an exponential increase of the error for the estimated cardinality C^E (see Equation 6).

Evaluation Metric: For every experimental run, we applied a random train/validation split of 80%/20% on the dataset and trained the samples using batches. While in the classical neural network, we used a batch size of 64, we evaluated the optimal batch size for the quantum model (see details in Section 5.2).

After a successful forward pass of a single batch, the mean squared error (MSE) as shown in Equation 7 is calculated and the gradients for all the parameters are computed in the backward pass of the neural network.

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y_i})^2 \tag{7}$$

For the quantum neural network, the parameter-shift rule [37] was used to evaluate analytically the gradients with respect to the parameters and quantum circuits inputs.

The quantum neural networks described in Section 4.2 were evaluated using the l parameter with values of $2, 3, 4$ and the k parameter with values of $1, 2$. Each combination was run 5 times (having different train/validation split) to obtain a mean value μ and standard deviation σ of the MSE loss after 500 epochs.

While the trainings were computed using 100 shots for each quantum circuit, for the evaluations of the models 1,000 shots were used. By using a higher shot size, the values were better reproducible when repeating the evaluations.

From the best training result of the 5 runs, the q-error (see Equation 8) was evaluated from the reconstructed C^E (see Equation 6) to quantitatively compare the performance of the model as a cardinality estimation method.

$$\text{Q-Error} = \max\left(\frac{C^E}{C^T}, \frac{C^T}{C^E}\right) \tag{8}$$

Note that for cardinality estimation the q-error is a commonly used metric because it penalizes both overestimation and underestimation of the true cardinality C^T [15].

Software: Every quantum circuit in the forward and backward pass was configured to run with 100 shots on the Statevector Aer-Simulator [4, 33] to obtain the resulting probability distribution and AMSGrad [34] was used to process the optimization steps after each batch computation. The quantum circuits were implemented using Qiskit [33] combined with the TorchConnector from the Qiskit

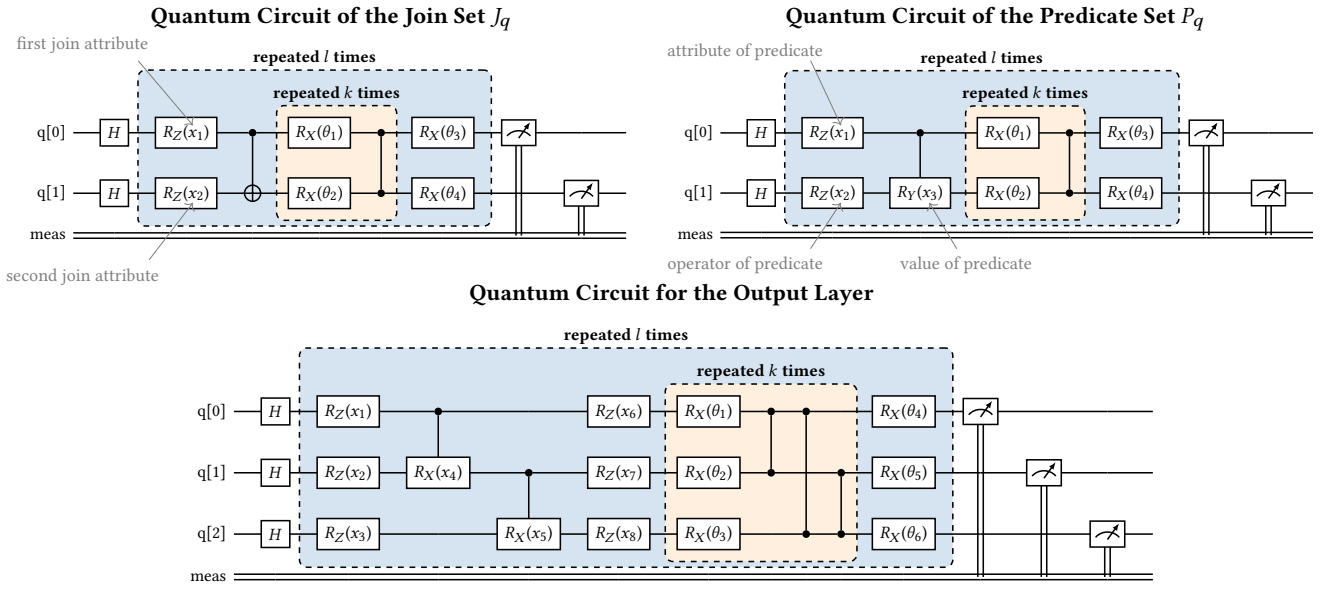

Figure 5: Quantum circuits for the Join Set J_q, the Predicate Set P_q and the output layer. The variables x_i indicate the input values, while the θ_is are the trainable parameters of the quantum neural network.

Machine Learning library [3, 33] to perform the training procedure with PyTorch [31].

Hardware: Our experiments showed that simulations of small quantum circuits were faster computed on a CPU rather than a GPU. Therefore, we executed the cardinality estimation experiments either on a *MacBook Pro, Apple M2 Max CPU with 32 GB RAM*, a *Virtual Linux Server with 16 Intel VCPUs and 64 GB RAM* or a *Linux Server with 2 Intel Xeon Platinum 8270 CPUs (26 cores each) with 384 GB RAM*. However, since in our experiments are focused on cardinality estimations and not on query execution times, the results are not affected by the hardware.

lr	$batchsize = 32$		$batchsize = 64$	
	Train Loss in 10^{-3}	Val Loss in 10^{-3}	Train Loss in 10^{-3}	Val Loss in 10^{-3}
10^{-1}	8.22	7.76	6.99	7.97
10^{-2}	**6.52**	**6.64**	**5.59**	**5.62**
10^{-3}	5.97	7.10	7.22	5.53
10^{-4}	7.56	8.26	8.73	8.97

Table 1: Mean squared error (MSE) after 500 epochs of training one QardEst model.

5.2 Hyperparameter Tuning

Firstly, we evaluated the optimal hyperparameters for the quantum neural network before analyzing the performance of the models.

The best hyperparameters for a certain batch size and learning rate lr were evaluated on the architecture using $l = 3$ and $k = 1$. The models were trained using batch sizes of 32 and 64 with learning rates of 10^{-1}, 10^{-2}, 10^{-3} and 10^{-4}. Each of these combinations was trained once. The MSE losses of the train and validation dataset after 500 epochs are shown in Table 1.

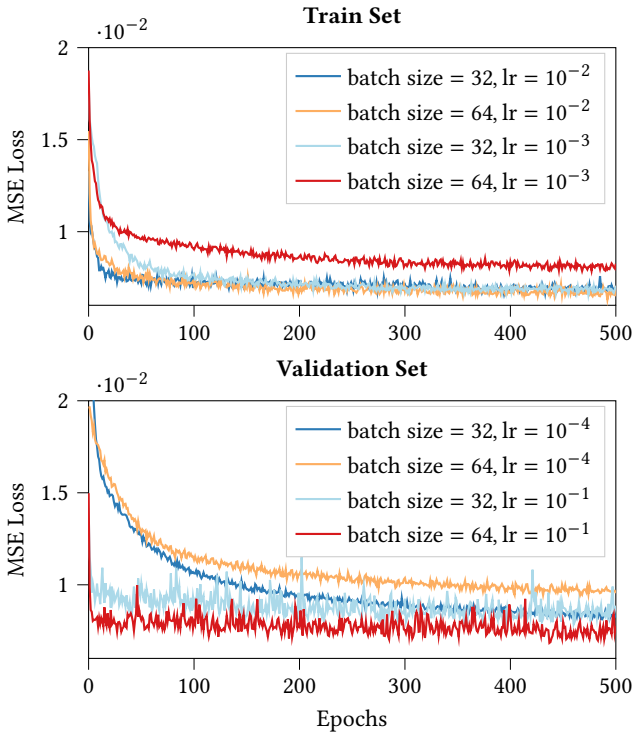

Figure 6: QardEst MSE loss with hyperparameter-tuning for the train set (top) and the validation set (bottom).

Table 1 shows that a learning rate $lr = 10^{-2}$ has the best results with batch size = 32 and batch size = 64. In general, smaller learning rates lr seem to have a better MSE loss with a batch size = 32 (see Table 1 and Figure 6). Therefore, pragmatically it was decided to choose the hyperparameters $lr = 10^{-2}$ and batch size = 32 for further evaluations, even if the hyperparameters $lr = 10^{-2}$, batch size = 64 showed the best results after 500 epochs. Furthermore, qualitatively we observed that the models converge faster with a batch size = 32 (see Figure 6).

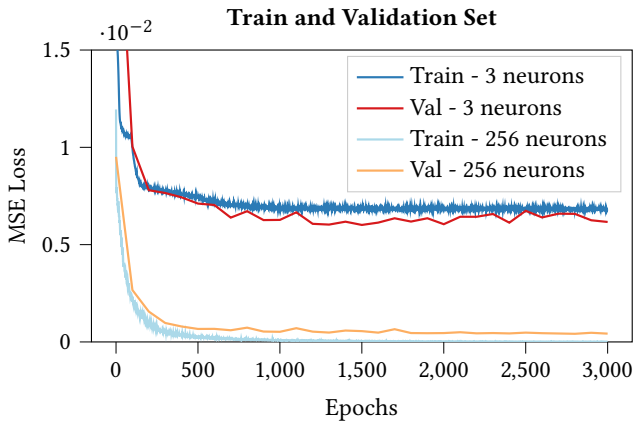

Figure 7: Train and validation MSE loss using the classical multi-set convolutional network architecture (MSCN) with 3 respectively 256 neurons in each hidden layer.

5.3 Performance of the Classical Neural Network - The Baseline

Let us first analyze the performance of the classical neural network that we consider as the classical baseline. We evaluated the multi-set convolutional network (MSCN) architecture, including our adjustments by training the model with the hyperparameters batch size = 64 and $lr = 10^{-3}$. Figure 7 shows the results with 3 and 256 neurons inside of each linear layer. We can observe that the models converge after about 100 and 500 epochs, respectively.

5.4 Performance of the Quantum Neural Network

We now describe the performance of our quantum neural network and compare it against the performance of the classical neural network baseline. In the first set of experiments we chose the classical neural network such that it has the same number of parameters as the quantum neural network.

On the classical neural network with 3 neurons in each hidden layer, an average MSE loss of $9.27 \cdot 10^{-3}$ was observed on the training dataset (see Table 2). The smallest quantum neural network ($l = 2, k = 1$) has an average train MSE loss of $6.86 \cdot 10^{-3}$, which is an improvement of **26.0%**. On the validation dataset, the quantum neural network has an average MSE loss of $7.04 \cdot 10^{-3}$ and the classical neural network an average MSE loss of $9.57 \cdot 10^{-3}$, which is an improvement of **26.44%**.

These results indicate that a **quantum neural network can improve the MSE loss using fewer trainable parameters**: the quantum neural network with $l = 2$ and $k = 1$ holds 37 and the classical neural network with 3 neurons has 70 trainable parameters. The quantum neural network with $l = 4$ and $k = 2$ has 93 trainable parameters and shows the best result of the quantum models with **an average train MSE loss of** $5.68 \cdot 10^{-3}$ and **an average validation MSE loss of** $6.25 \cdot 10^{-3}$. Nevertheless, the classical neural network with 256 neurons in each hidden layer still achieved a better mean MSE loss with $0.29 \cdot 10^{-3}$ on the training dataset and $0.83 \cdot 10^{-3}$ on the validation dataset, but it consists of 264,961 trainable parameters, i.e. it has 2,849 times more parameters than the largest quantum neural network.

We now analyze how well the classical and the quantum neural networks *predict the cardinalities* of the queries. Figures 8 and 9 show the true cardinalities (labels y) and the predicted cardinalities ($\hat{y}$). The results are sorted by the labels y for each SQL query. We can observe that the classical neural network with 256 neurons in each hidden layer learns the cardinalities from the SQL queries qualitatively well. The quantum neural network ($l = 4, k = 2$), in comparison, approximates the function, but has much higher errors to the real cardinalities. Therefore, the quantum model is feasible to approximate the complex function for cardinality estimation but still needs to be improved to obtain the same results as the classical neural network with 256 neurons in each hidden layer.

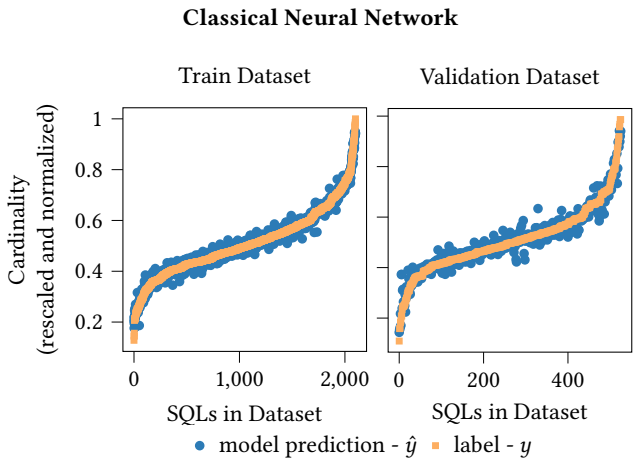

Figure 8: True cardinalities y and predicted cardinalities $\hat{y}$ of the classical neural network with 256 neurons in each hidden layer (after 500 epochs).

6 ENHANCEMENT OF THE QUANTUM NEURAL NETWORK ARCHITECTURE

The previous results showed the possibility for the quantum neural network to outperform the classical neural network when both use a similar number of parameters. Nevertheless, a classical neural network with 256 neurons in each hidden layer has still the best performance. Therefore, it raises the question of how the quantum

Quantum Neural Network									Classical Neural Network				
	k=1				k=2								
	Train in 10^{-3}		Val in 10^{-3}		Train in 10^{-3}		Val in 10^{-3}			Train in 10^{-3}		Val in 10^{-3}	
# layers (#params)	μ	σ	μ	σ	μ	σ	μ	σ	# neurons (#params)	μ	σ	μ	σ
$l = 2, (37; 51)$	6.86	0.379	7.04	0.773	6.64	0.405	6.44	0.342	$n = 3, (70)$	9.27	2.946	9.57	3.361
$l = 3, (51; 72)$	6.30	0.412	6.04	0.514	5.52	0.167	5.56	0.432	$n = 4, (109)$	7.18	0.243	7.62	0.440
$l = 4, (65; \underline{93})$	5.32	0.311	6.48	0.710	**4.58**	**0.145**	**5.35**	**0.587**	$n = 8, (345)$	6.15	0.604	6.84	0.264
									$n = 16, (1, 201)$	5.33	0.744	6.05	0.712
									$n = 32, (\underline{4, 449})$	2.81	0.290	3.48	0.332
									$n = 256, (264, 961)$	0.29	0.036	0.83	0.115

Table 2: MSE loss of the quantum neural network trained N=5 times after 500 epochs and comparison to a classical neural network. #neurons gives the number of classical neurons per hidden layer in the classical MSCN approach shown in Figure 2. #params indicates the number of trainable parameters of the respective model. The underlined parameters show when the best quantum neural network is surpassed by the classical counterpart, i.e. 93 vs 4,449.

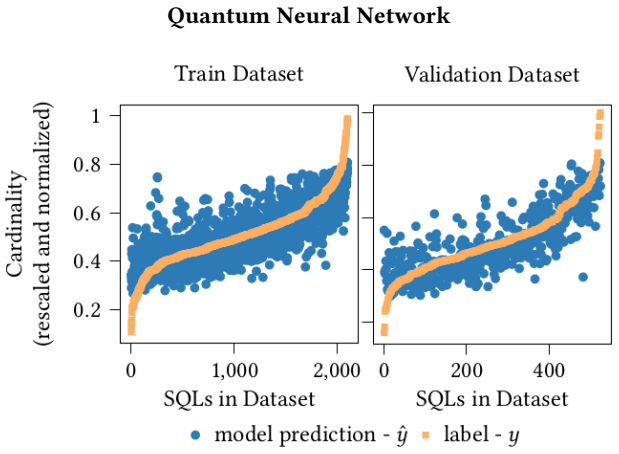

Figure 9: True cardinalities y and predicted cardinalities $\hat{y}$ of the quantum neural network with $l = 4, k = 2$ (after 500 epochs).

neural network can be extended to perform better. Firstly, the results already indicate, that an increase of parameters k and l improves the results. In particular, we experimented with the following approach to enhance the quantum neural network model architecture further:

Increasing degrees of freedom in quantum neural network: So far, we only used R_x rotation blocks in the parameterizable quantum circuit (PQC). Because a qubit can be visualized on the Bloch sphere, it can be assumed that a PQC performs better if rotations about all the axes are used. Therefore, we used a model with circuits as shown in Figure 5. However, instead of R_x rotation blocks, we experimented with a sequence of R_x-R_y-R_z and another model with a sequence of R_x-R_y rotation blocks. Theoretically, with 2 degrees of freedom (DoF) the whole Bloch sphere can be explored.

Removing measurements after the Join Set J_q and Predicate Set P_q circuits and creating a combined circuit: A measurement destroys the quantum state and therefore information is lost. Hence, we developed and evaluated a combined quantum model as shown in Figure 10. This new model can be applied to combine the Predicate Set and Join Set. If the two sets have different lengths, we used zero padding and in the output of the combined model we applied an average pool, followed by a single perceptron and a sigmoid activation function.

Spreading input encoding on more qubits using entanglements: We extended the circuits for the Join Set J_q and Predicate Set P_q to 4 qubits, while the output circuit was extended to 8 qubits. The input information is spread by entanglements to all the qubits and we trained a model using the parameters $l = 2$ and $k = 1$ with different learning rates lr.

Figure 11 shows, that this way of extension does not improve the results and the training can not improve the minima as the model with 2 qubits. It seems, that at about a train MSE loss of $1.6 \cdot 10^{-2}$ the models with 4 qubits reach a plateau and improve quite slowly from there.

Barren plateaus, which are majorly influenced by the chosen entanglement strategy and the largeness of the circuit, are areas in the loss function where no clear direction for optimization exists, and usually, it happens at random initialization of the model [26, 27]. Research within this area indicates that quantum convolutional neural networks are immune [26, 32], while dissipative quantum neural networks suffer from barren plateau [26, 42]. Our experiments using a dissipative quantum neural network and a quantum convolution neural network show, that the dissipative model can reach similar performances to the original quantum neural network with $l = 2, k = 1$ on two qubits (see Figure 12). However, the quantum convolutional neural network shows quite a bad performance. Therefore, we can not conclude that we observed a barren plateau because it contradicts the results in the literature.

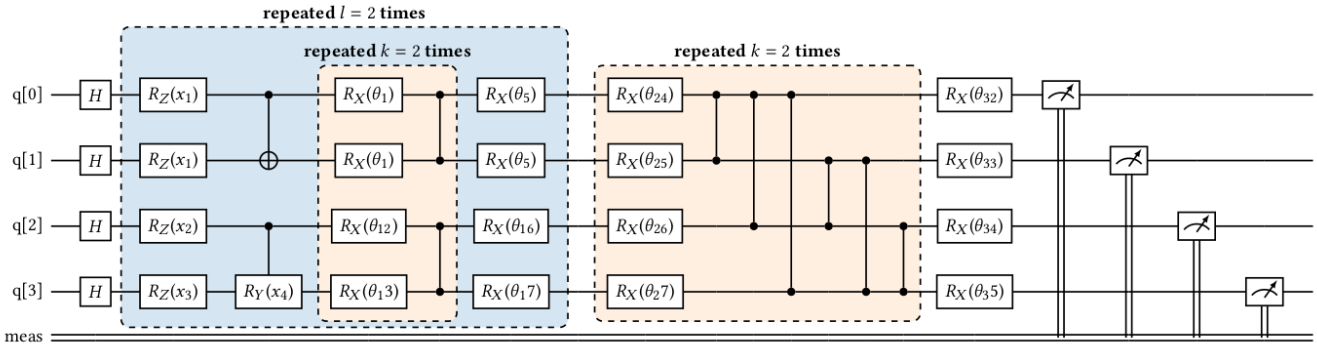

Figure 10: Quantum circuit of the combined model.

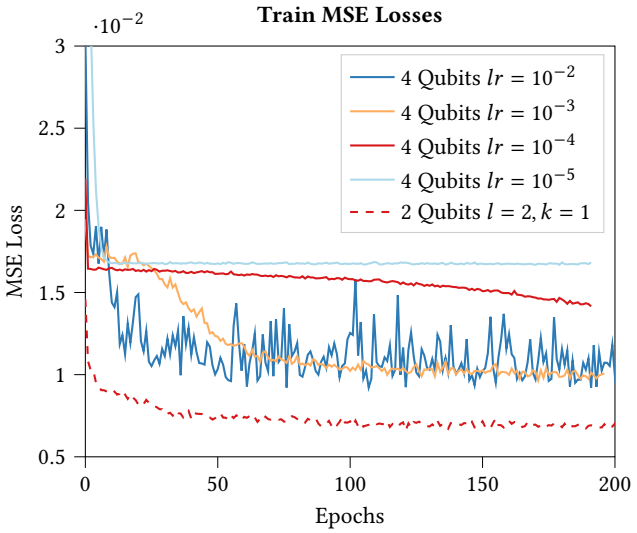

Figure 11: MSE losses of the quantum neural network model with 4 qubits.

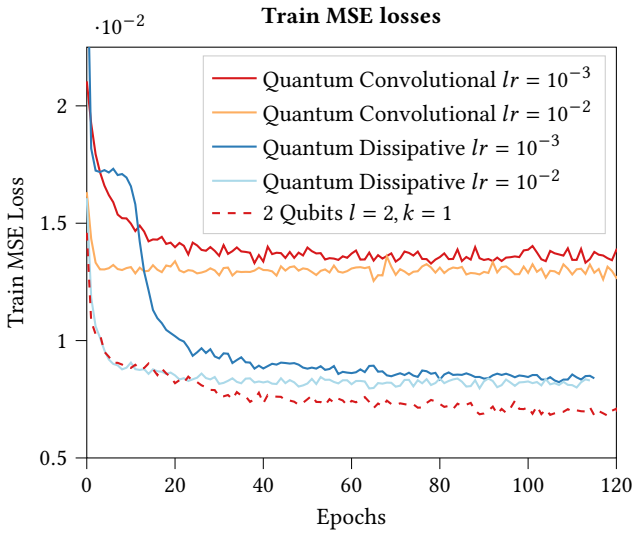

Figure 12: MSE losses of the quantum dissipative neural network and the quantum convolutional neural network.

7 OVERALL PERFORMANCE COMPARISON

We will now compare the overall performance of quantum neural networks with classical neural networks as a function of the number of parameters used by each model.

7.1 MSE Loss

Figure 13 shows the mean MSE losses for the train and validation dataset depending on the number of trainable parameters in the model. Let us focus our attention on the MSE loss of the validation dataset (right side of Figure 13). We observe that the classical neural network using 3 neurons in hidden layers with 70 trainable parameters has a mean MSE validation loss of $9.57 \cdot 10^{-3}$, while the quantum neural network using $l = 3, k = 2$ with 72 trainable parameters has an MSE Loss of $5.56 \cdot 10^{-3}$, which is an improvement of 41.9%. We can also see that the best quantum neural network is the one with $l = 4, k = 2$. In other words, **the best quantum neural network has only 93 parameters and outperforms the classical neural networks up to 16 neurons having 1,201 parameters.**

Quantum neural networks with an increased l tend to have better MSE losses compared to a model with the same number of trainable parameters but a lower l parameter. For instance, the model $l = 3, k = 1$ has an improved validation MSE loss of 6.21% compared to the model with $l = 2, k = 2$ and the same number of trainable parameters.

Due to the overfitting of the model with $l = 4, k = 1$, the same tendency can not be observed for the models with $l = 4, k = 1$ and $l = 3, k = 2$. Generally, it can be said that models having $l = 4$ tend to overfit more than the other models.

Furthermore, the model with R_x-R_y sequences compared with the $l = 2, k = 1$ model containing just R_x rotation blocks improves the average validation MSE loss by 16.8% from $7.04 \cdot 10^{-3}$ to $5.86 \cdot 10^{-3}$. However, appending an additional R_z rotation block just increases the validation MSE loss by 4.22% to $6.11 \cdot 10^{-3}$. This could be explained by the fact, that two different rotation blocks

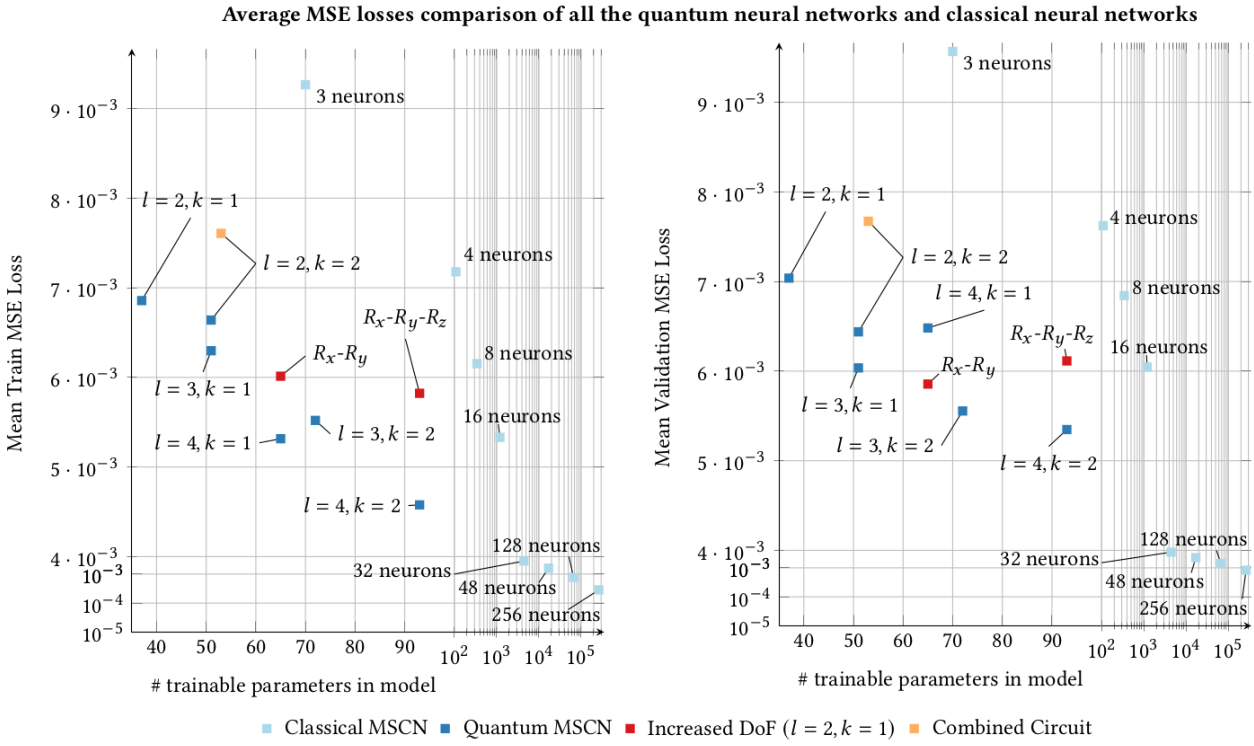

Figure 13: Mean MSE loss (N=5) on the training dataset (left) and validation dataset (right) of quantum neural networks and classical neural networks plotted as a function of the trainable parameters. The scale of the mean MSE loss (y-axis) is logarithmic until a value of $4 \cdot 10^{-3}$, afterwards, the scale is linear. In contrast, the scale of the trainable parameters (x-axis) is linear until a value of 10^2, afterwards, the scale is logarithmic.

are already enough degrees of freedom to reach every point in the Bloch sphere.

7.2 Q-Error

Let us conclude our analysis with an evaluation of the q-error for all models. For our experiments we evaluated the q-errors of the best quantum model $l = 4, k = 2$ based on 5 different runs of train/validation splits. We also evaluated the q-errors of the best classical neural networks with up to 256 neurons on the same train/validation splits as the quantum model. Afterwards, we calculated the 50%-, 90%- and 99%-percentiles.

In Table 3 we can see that the quantum neural network with $l = 4, k = 2$ achieves better q-error results on the 50%-, 90%- and 99%-percentiles than the classical neural network with 3, 4 and partially 8 neurons in each hidden layer. Starting from 16 neurons, the classical neural network shows lower q-errors.

Q-Errors

Model	Train Dataset 50%	Train Dataset 90%	Train Dataset 99%	Validation Dataset 50%	Validation Dataset 90%	Validation Dataset 99%
Classical Neural Networks						
3 neurons	3.12	36.67	775.69	3.09	26.90	400.10
4 neurons	3.40	32.53	455.15	3.28	20.88	349.14
8 neurons	2.60	21.96	629.29	2.55	17.11	164.65
16 neurons	2.14	12.35	132.30	2.11	10.52	73.83
256 neurons	1.26	1.91	3.44	1.40	2.77	9.57
Quantum Neural Networks						
$l = 4, k = 2$	2.74	13.85	170.21	2.74	15.35	189.37

Table 3: Q-error percentile comparison of the best quantum neural network to classical neural networks.

8 CONCLUSIONS

In this paper we introduced QardEst - a novel quantum natural network to estimate the cardinality of join queries. QardEst is based on the multi-set convolutional network architecture [21] and replaces parts of the layers with quantum circuits. Our experiments demonstrate that QardEst executed on a quantum simulator outperforms

classical neural networks with up to 16 neurons but uses about 13 times fewer parameters than the classical counterpart.

However, it is noteworthy that classical neural networks with 256 neurons in each hidden layer outperform all assessed quantum neural networks. If quantum neural networks should match the performance of a classical neural network with 256 neurons, it would require a quantum neural network with an equivalent number of trainable parameters. Unfortunately, simulating a quantum neural network with 264, 961 trainable parameters within a reasonable timeframe is currently unfeasible given the hardware we had for our experiments. Nevertheless, the study revealed that the performance of quantum neural networks can be improved through the re-uploading of the input encodings and the adoption of a dissipative design for the quantum circuits.

Interesting future research directions are to implement the quantum neural networks on real quantum hardware and perform extensive scalability experiments to better understand both the theoretical as well as practical limitations of quantum neural network architectures for cardinality estimation.

ACKNOWLEDGMENTS

The project has received funding from the Swiss National Science Foundation under grant number 1921052.

REFERENCES

[1] PostgreSQL Documentation 2023. *76.1. Row Estimation Examples*. PostgreSQL Documentation. https://www.postgresql.org/docs/16/row-estimation-examples.html

[2] PostgreSQL Documentation 2023. *76.2. Multivariate Statistics Examples*. PostgreSQL Documentation. https://www.postgresql.org/docs/16/multivariate-statistics-examples.html

[3] Qiskit Community 2023. *Qiskit-Community/Qiskit-Machine-Learning*. Qiskit Community. https://github.com/qiskit-community/qiskit-machine-learning

[4] Qiskit 2023. *Qiskit/Qiskit-Aer*. Qiskit. https://github.com/Qiskit/qiskit-aer

[5] MariaDB KnowledgeBase 2024. *InnoDB Persistent Statistics*. MariaDB KnowledgeBase. https://mariadb.com/kb/en/innodb-persistent-statistics/

[6] 2024. *MySQL :: MySQL 8.0 Reference Manual :: 15.8.10.1 Configuring Persistent Optimizer Statistics Parameters*. https://dev.mysql.com/doc/refman/8.0/en/innodb-persistent-stats.html

[7] Nicolas Bruno, Surajit Chaudhuri, and Luis Gravano. 2001. STHoles: A Multidimensional Workload-Aware Histogram. *ACM SIGMOD Record* 30, 2 (2001), 211–222. https://doi.org/10.1145/376284.375686

[8] Davide Castelvecchi. 2023. IBM quantum computer passes calculation milestone. *Nature* 618 (06 2023). https://doi.org/10.1038/d41586-023-01965-3

[9] Amol Deshpande, Minos Garofalakis, and Rajeev Rastogi. 2001. Independence Is Good: Dependency-Based Histogram Synopses for High-Dimensional Data. *ACM SIGMOD Record* 30, 2 (2001), 199–210. https://doi.org/10.1145/376284.375685

[10] Anshuman Dutt, Chi Wang, Azade Nazi, Srikanth Kandula, Vivek Narasayya, and Surajit Chaudhuri. 2019. Selectivity Estimation for Range Predicates Using Lightweight Models. *Proceedings of the VLDB Endowment* 12, 9 (2019), 1044–1057. https://doi.org/10.14778/3329772.3329780

[11] A. Einstein, B. Podolsky, and N. Rosen. 1935. Can Quantum-Mechanical Description of Physical Reality Be Considered Complete? *Phys. Rev.* 47 (May 1935), 777–780. Issue 10. https://doi.org/10.1103/PhysRev.47.777

[12] Tobias Fankhauser, Marc E Solèr, Rudolf M Füchslin, and Kurt Stockinger. 2023. Multiple Query Optimization Using a Gate-Based Quantum Computer. *IEEE Access* (2023).

[13] Francisco Javier Gil Vidal and Dirk Oliver Theis. 2020. Input Redundancy for Parameterized Quantum Circuits. *Frontiers in Physics* 8 (2020), 297. https://doi.org/10.3389/fphy.2020.00297

[14] Dimitrios Gunopulos, George Kollios, Vassilis J. Tsotras, and Carlotta Domeniconi. 2005. Selectivity Estimators for Multidimensional Range Queries over Real Attributes. *The VLDB Journal* 14, 2 (2005), 137–154. https://doi.org/10.1007/s00778-003-0090-4

[15] Yuxing Han, Ziniu Wu, Peizhi Wu, Rong Zhu, Jingyi Yang, Liang Wei Tan, Kai Zeng, Gao Cong, Yanzhao Qin, Andreas Pfadler, Zhengping Qian, Jingren Zhou, Jiangneng Li, and Bin Cui. 2021. Cardinality Estimation in DBMS: A Comprehensive Benchmark Evaluation. *Proceedings of the VLDB Endowment* 15, 4 (2021), 752–765. https://doi.org/10.14778/3503585.3503586

[16] Max Heimel, Martin Kiefer, and Volker Markl. 2015. Self-Tuning, GPU-Accelerated Kernel Density Models for Multidimensional Selectivity Estimation. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data* (Melbourne Victoria Australia). ACM, 1477–1492. https://doi.org/10.1145/2723372.2749438

[17] Benjamin Hilprecht, Andreas Schmidt, Moritz Kulessa, Alejandro Molina, Kristian Kersting, and Carsten Binnig. 2020. DeepDB: Learn from Data, Not from Queries! *Proceedings of the VLDB Endowment* 13, 7 (2020), 992–1005. https://doi.org/10.14778/3384345.3384349

[18] Kurt Hornik. 1991. Approximation Capabilities of Multilayer Feedforward Networks. *Neural Networks* 4, 2 (1991), 251–257. https://doi.org/10.1016/0893-6080(91)90009-T

[19] Martin Kiefer, Max Heimel, Sebastian Breß, and Volker Markl. 2017. Estimating Join Selectivities Using Bandwidth-Optimized Kernel Density Models. *Proceedings of the VLDB Endowment* 10, 13 (2017), 2085–2096. https://doi.org/10.14778/3151106.3151112

[20] Kyoungmin Kim, Jisung Jung, In Seo, Wook-Shin Han, Kangwoo Choi, and Jaehyok Chong. 2022. Learned Cardinality Estimation: An In-depth Study. In *Proceedings of the 2022 International Conference on Management of Data* (Philadelphia, PA, USA) *(SIGMOD '22)*. Association for Computing Machinery, New York, NY, USA, 1214–1227. https://doi.org/10.1145/3514221.3526154

[21] Andreas Kipf, Thomas Kipf, Bernhard Radke, Viktor Leis, Peter Boncz, and Alfons Kemper. 2018. Learned Cardinalities: Estimating Correlated Joins with Deep Learning. *CIDR* (2018). https://doi.org/10.48550/ARXIV.1809.00677

[22] Claude Lehmann, Pavel Sulimov, and Kurt Stockinger. 2024. Is Your Learned Query Optimizer Behaving As You Expect? A Machine Learning Perspective. *Proc. VLDB Endow. 17(7): 1565-1577* (2024).

[23] Viktor Leis, Andrey Gubichev, Atanas Mirchev, Peter Boncz, Alfons Kemper, and Thomas Neumann. 2015. How Good Are Query Optimizers, Really? *Proceedings of the VLDB Endowment* 9, 3 (2015), 204–215. https://doi.org/10.14778/2850583.2850594

[24] Viktor Leis, Bernhard Radke, Andrey Gubichev, Alfons Kemper, and Thomas Neumann. 2017. Cardinality Estimation Done Right: Index-based Join Sampling.. In *CIDR*.

[25] Feifei Li, Bin Wu, Ke Yi, and Zhuoyue Zhao. 2016. Wander Join: Online Aggregation via Random Walks. In *Proceedings of the 2016 International Conference on Management of Data* (San Francisco California USA). ACM, 615–629. https://doi.org/10.1145/2882903.2915235

[26] S. Mangini, F. Tacchino, D. Gerace, D. Bajoni, and C. Macchiavello. 2021. Quantum Computing Models for Artificial Neural Networks. *Europhysics Letters* 134, 1 (2021), 10002. https://doi.org/10.1209/0295-5075/134/10002

[27] Jarrod R. McClean, Sergio Boixo, Vadim N. Smelyanskiy, Ryan Babbush, and Hartmut Neven. 2018. Barren Plateaus in Quantum Neural Network Training Landscapes. *Nature Communications* 9, 1 (2018), 4812. https://doi.org/10.1038/s41467-018-07090-4

[28] D.A. McQuarrie. 2007. *Quantum Chemistry*. Viva Books Private Limited. https://books.google.es/books?id=2VLpQwAACAAJ

[29] Nikolaj Moll, Panagiotis Barkoutsos, Lev S Bishop, Jerry M Chow, Andrew Cross, Daniel J Egger, Stefan Filipp, Andreas Fuhrer, Jay M Gambetta, Marc Ganzhorn, Abhinav Kandala, Antonio Mezzacapo, Peter Müller, Walter Riess, Gian Salis, John Smolin, Ivano Tavernelli, and Kristan Temme. 2018. Quantum optimization using variational algorithms on near-term quantum devices. *Quantum Science and Technology* 3, 3 (jun 2018), 030503. https://doi.org/10.1088/2058-9565/aab822

[30] M. Muralikrishna and David J. DeWitt. 1988. Equi-Depth Multidimensional Histograms. *ACM SIGMOD Record* 17, 3 (1988), 28–36. https://doi.org/10.1145/971701.50205

[31] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, Soumith Chintala, H. Wallach, H. Larochelle, A. Beygelzimer, prefix=d'Alchéuseprefix=true family=Buc, given=F., E. Fox, and R. Garnett. 2019. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. Curran Associates, Inc. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[32] Arthur Pesah, M. Cerezo, Samson Wang, Tyler Volkoff, Andrew T. Sornborger, and Patrick J. Coles. 2021. Absence of Barren Plateaus in Quantum Convolutional Neural Networks. *Physical Review X* 11, 4 (2021), 041011. https://doi.org/10.1103/PhysRevX.11.041011

[33] Qiskit contributors. 2023. Qiskit: An Open-Source Framework for Quantum Computing. https://doi.org/10.5281/zenodo.2573505

[34] Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. 2019. On the Convergence of Adam and Beyond. (2019). https://doi.org/10.48550/ARXIV.1904.09237

[35] Silvan Reiner and Michael Grossniklaus. 2024. Sample-Efficient Cardinality Estimation Using Geometric Deep Learning. *Proc. VLDB Endow.* 17, 4 (mar 2024), 740–752. https://doi.org/10.14778/3636218.3636229

[36] Alessandro Romito. 2023. Quantum hardware measures up to the challenge. *Nature Physics* 19 (06 2023). https://doi.org/10.1038/s41567-023-02090-8

[37] Maria Schuld, Ville Bergholm, Christian Gogolin, Josh Izaac, and Nathan Killoran. 2019. Evaluating Analytic Gradients on Quantum Hardware. *Physical Review A* 99, 3 (2019), 032331. https://doi.org/10.1103/PhysRevA.99.032331 arXiv:1811.11184 [quant-ph]

[38] Maria Schuld, Ryan Sweke, and Johannes Jakob Meyer. 2021. Effect of Data Encoding on the Expressive Power of Variational Quantum-Machine-Learning Models. *Physical Review A* 103, 3 (2021), 032430. https://doi.org/10.1103/PhysRevA.103.032430

[39] Manuel Schönberger, Immanuel Trummer, and Wolfgang Mauerer. 2023. Quantum-Inspired Digital Annealing for Join Ordering. In *Proceedings of the VLDB Endowment*, Vol. 17. https://doi.org/10.14778/3632093.3632112

[40] Manuel Schönberger, Immanuel Trummer, and Wolfgang Mauerer. 2023. Quantum Optimisation of General Join Trees. In *Proceedings of the International Workshop on Quantum Data Science and Management (QDSM '23)*.

[41] P. Griffiths Selinger, M. M. Astrahan, D. D. Chamberlin, R. A. Lorie, and T. G. Price. 1979. Access Path Selection in a Relational Database Management System. In *Proceedings of the 1979 ACM SIGMOD International Conference on Management of Data - SIGMOD '79* (Boston, Massachusetts). ACM Press, 23. https://doi.org/10.1145/582095.582099

[42] Kunal Sharma, M. Cerezo, Lukasz Cincio, and Patrick J. Coles. 2022. Trainability of Dissipative Perceptron-Based Quantum Neural Networks. *Physical Review Letters* 128, 18 (2022), 180505. https://doi.org/10.1103/PhysRevLett.128.180505

[43] Peter W. Shor. 1999. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM Rev.* 41, 2 (1999), 303–332. https://doi.org/10.1137/S0036144598347011

[44] Ricardo Daniel Monteiro Simões, Patrick Huber, Nicola Meier, Nikita Smailov, Rudolf M Füchslin, and Kurt Stockinger. 2023. Experimental evaluation of quantum machine learning algorithms. *IEEE Access* 11 (2023), 6197–6208.

[45] Immanuel Trummer and Christoph Koch. 2016. Multiple Query Optimization on the D-Wave 2X Adiabatic Quantum Computer. *Proceedings of the VLDB Endowment* 9, 9 (2016).

[46] Valter Uotila. 2023. SQL2Circuits: Estimating Metrics for SQL Queries with A Quantum Natural Language Processing Method. arXiv:2306.08529 [cs.DB]

[47] Hai Wang and Kenneth C. Sevcik. 2003. A Multi-Dimensional Histogram for Selectivity Estimation and Fast Approximate Query Answering. In *Proceedings of the 2003 Conference of the Centre for Advanced Studies on Collaborative Research* (Toronto, Ontario, Canada) *(CASCON '03)*. IBM Press, 328–342.

[48] Xiaoying Wang, Changbo Qu, Weiyuan Wu, Jiannan Wang, and Qingqing Zhou. 2021. Are We Ready for Learned Cardinality Estimation? *Proceedings of the VLDB Endowment* 14, 9 (2021), 1640–1654. https://doi.org/10.14778/3461535.3461552

[49] WilliamDAssafMSFT. 2024. *Cardinality Estimation (SQL Server) - SQL Server*. https://learn.microsoft.com/en-us/sql/relational-databases/performance/cardinality-estimation-sql-server?view=sql-server-ver16

[50] Yadong Wu, Juan Yao, Pengfei Zhang, and Hui Zhai. 2021. Expressivity of quantum neural networks. *Physical Review Research* 3, 3 (Aug. 2021). https://doi.org/10.1103/physrevresearch.3.l032049

[51] Ziniu Wu, Parimarjan Negi, Mohammad Alizadeh, Tim Kraska, and Samuel Madden. 2023. FactorJoin: A New Cardinality Estimation Framework for Join Queries. *Proc. ACM Manag. Data* 1, 1, Article 41 (may 2023), 27 pages. https://doi.org/10.1145/3588721

[52] Ziniu Wu, Amir Shaikhha, Rong Zhu, Kai Zeng, Yuxing Han, and Jingren Zhou. 2020. BayesCard: Revitilizing Bayesian Frameworks for Cardinality Estimation. (2020). https://doi.org/10.48550/ARXIV.2012.14743

[53] Ziniu Wu, Pei Yu, Peilun Yang, Rong Zhu, Yuxing Han, Yaliang Li, Defu Lian, Kai Zeng, and Jingren Zhou. 2021. A Unified Transferable Model for ML-Enhanced DBMS. (2021). https://doi.org/10.48550/ARXIV.2105.02418

[54] Zongheng Yang, Amog Kamsetty, Sifei Luan, Eric Liang, Yan Duan, Xi Chen, and Ion Stoica. 2020. NeuroCard: One Cardinality Estimator for All Tables. *Proceedings of the VLDB Endowment* 14, 1 (2020), 61–73. https://doi.org/10.14778/3421424.3421432

[55] Zongheng Yang, Eric Liang, Amog Kamsetty, Chenggang Wu, Yan Duan, Xi Chen, Pieter Abbeel, Joseph M. Hellerstein, Sanjay Krishnan, and Ion Stoica. 2019. Deep Unsupervised Cardinality Estimation. *Proceedings of the VLDB Endowment* 13, 3 (2019), 279–292. https://doi.org/10.14778/3368289.3368294

[56] Zhuoyue Zhao, Robert Christensen, Feifei Li, Xiao Hu, and Ke Yi. 2018. Random Sampling over Joins Revisited. In *Proceedings of the 2018 International Conference on Management of Data* (Houston TX USA). ACM, 1525–1539. https://doi.org/10.1145/3183713.3183739

[57] Rong Zhu, Ziniu Wu, Yuxing Han, Kai Zeng, Andreas Pfadler, Zhengping Qian, Jingren Zhou, and Bin Cui. 2021. FLAT: Fast, Lightweight and Accurate Method for Cardinality Estimation. *Proceedings of the VLDB Endowment* 14, 9 (2021), 1489–1502. https://doi.org/10.14778/3461535.3461539

[58] Christa Zoufal, Aurélien Lucchi, and Stefan Woerner. 2019. Quantum Generative Adversarial Networks for Learning and Loading Random Distributions. *npj Quantum Information* 5, 1 (2019), 103. https://doi.org/10.1038/s41534-019-0223-2