Neural Circuit Parameter Prediction for Efficient Quantum Data Loading

Dohun Kim, Sunghye Park and Seokhyeong Kang
Department of Electrical Engineering, Pohang University of Science and Technology, Pohang, Republic of Korea
{dohunkim, shpark96, shkang}@postech.ac.kr

***Abstract*—Quantum machine learning (QML) has demonstrated the potential to outperform classical machine learning algorithms in various fields. However, encoding classical data into quantum states, known as quantum data loading, remains a challenge. Existing methods achieve high accuracy in loading single data, but lack efficiency for large-scale data loading tasks.**

In this work, we propose *Neural Circuit Parameter Prediction*, a novel method that leverages classical deep neural networks to predict the parameters of parameterized quantum circuits directly from the input data. This approach benefits from the batch inference capability of neural networks and improves the accuracy of quantum data loading. We introduce real-valued parameterization of quantum circuits and a three-phase training strategy to further enhance training efficiency and accuracy. Experimental results on MNIST dataset show that our method achieves a 17.31% improvement in infidelity score and 108 times faster runtime compared to existing methods. Our approach provides an efficient solution for quantum data loading, enabling the practical deployment of QML algorithms on large-scale datasets.

***Index Terms*—Quantum machine learning, amplitude encoding, data loading, parameterized quantum circuits**

I. Introduction

Quantum machine learning (QML) has attracted significant attention in recent years, particularly in the context of noisy intermediate-scale quantum (NISQ) devices [1]. NISQ devices are quantum computers with limited qubit coherence time and gate fidelity, making them prone to errors [2]. QML algorithms, especially those using variational methods, are promising due to their hybrid quantum-classical structure, which reduces quantum resource demands and mitigates noise effects [3]. QML has been applied to various machine learning tasks, such as classification and generative modeling, demonstrating the potential of quantum advantages [4].

While there has been significant progress in QML research, many existing methods assume that the input data is readily available as quantum states. This often leads to an oversight of the challenges associated with data loading. Efficiently encoding classical data into quantum states is a critical step in QML, as it directly impacts the efficiency and practical advantage of QML algorithms [5]. However, preparing an arbitrary quantum state generally requires an exponential circuit depth [6] or an exponential number of ancillary qubits [7], making it impractical for high-dimensional data.

To mitigate this issue, approximation methods have been proposed which aim to trade-off between the accuracy of the quantum data loading and the quantum resources required. For example, tensor network-based methods [8], [9] approximate the quantum state using a tensor network representation, such as matrix product states (Fig. 1(a)). These methods allow for the efficient representation of quantum states with a small number of parameters, making them suitable for high-dimensional data. However, these methods require complex tensor decomposition [8] and iterative optimization [9] for each data point, adding to the computational burden. Moreover, tensor networks operations makes it difficult to leverage batch inference, which is essential for machine learning tasks.

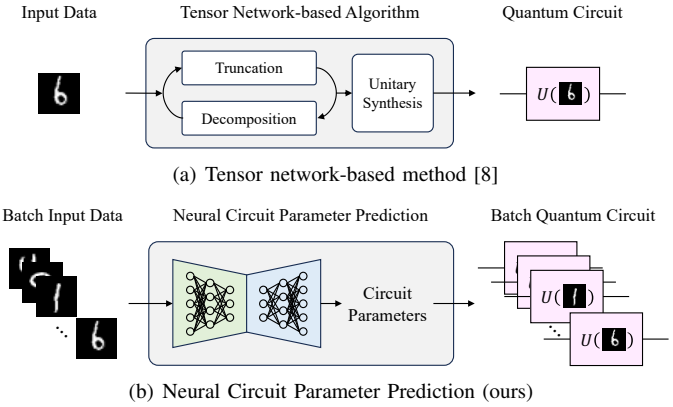


Fig. 1: Example of quantum data loading: (a) Previous method [8] based on tensor network, including time-consuming tensor network operations for each data point. (b) Proposed method using neural networks to directly predict circuit parameters, eliminating iterative process and enabling efficient batch inference.

In this paper, we introduce a novel method for quantum data loading using machine learning, termed *Neural Circuit Parameter Prediction* (Fig. 1(b)). Our method directly predicts the parameters of the quantum circuit from input data, leveraging the fast inference speed of neural networks and improved accuracy. We also investigate the impact of each component and discuss their advantages, demonstrating the efficiency of our approach in facilitating practical quantum advantage in machine learning tasks. Our contributions can be summarized as follows:

- We introduce an optimization problem formulation of the quantum data loading process using a parametrized quantum circuit. Then, we approximate the optimal function which maps raw data to circuit parameters with neural network.
- We propose *Neural Circuit Parameter Prediction* framework that predicts parameters directly from input data, by translating data to low-dimensional latent features, then reconstructing them as parameters. To further enhance the accuracy and training efficiency, we introduce real-valued gate parameterization and a three-step training strategy.

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No.2019R1A5A1027055). This research was supported by 'Creation of the quantum information science R&D ecosystem (based on human resources)' through the NRF funded by the MSIT (No.2021M3H3A1036573). This research was supported by 'Quantum Information Science R&D Ecosystem Creation' through the NRF funded by the MSIT (No. 2020M3H3A1110365).

978-3-9826741-0-0/DATE25/© 2025 EDAA

- Evaluated on the MNIST dataset [10], our method achieves a 17.31% improvement in accuracy and 108 times speedup in runtime compared to existing methods, demonstrating the efficiency of our approach.

The rest of this paper is organized as follows: Section II provides backgrounds and motivations. Section III describes the proposed method for quantum data loading. Section IV presents the experimental results and ablation studies. Finally, we conclude in Section V.

II. Background and Motivations

A. Quantum Data Loading

Quantum data loading is a crucial process in quantum computing, involving the encoding of classical data into quantum states. This enables quantum computers to process and analyze classical data, which is particularly essential for QML applications. In these applications, classical data must be transformed into quantum states for quantum algorithms to operate effectively. This work focuses on the amplitude encoding method, a standard technique for initial state preparation in QML applications.

Amplitude encoding directly maps a vector of classical data to the amplitudes of a quantum state. Initially, the classical data are vectorized into a flat vector and then normalized so that the sum of the absolute squared amplitudes is equal to one. The amplitude encoding of a real or complex vector $\mathbf{x} = (x_0, x_1, \ldots, x_{2^n-1})$ into a n-qubit state $|\psi(\mathbf{x})\rangle$ is expressed as:

$$|\psi(\mathbf{x})\rangle = \frac{1}{\|\mathbf{x}\|} \sum_{i=0}^{2^n-1} x_i |i\rangle, \text{ where } \|\mathbf{x}\| = \sqrt{\mathbf{x}^\dagger \mathbf{x}}, \quad (1)$$

where $|i\rangle$ represents the computational basis state corresponding to the binary representation of i.

In the context of image data, images are zero-padded to make the number of pixels a power of two. For instance, a grayscale image with dimensions $[w, h]$ is zero-padded to $[W, H]$ where W and H are the smallest powers of two greater than w and h, respectively. The image is then vectorized and normalized to form a quantum state of $\log_2(WH)$ qubits.

B. Circuit Implementations of Quantum Data Loading

To implement amplitude encoding on quantum computers, it is essential to construct gate-based circuits that prepare quantum states corresponding to classical data. The traditional approach, the Möttönen method [6], prepares a quantum state by executing a series of uniformly controlled rotation gates capable of manipulating individual elements of the state vector. However, this method requires an exponential number of gates, making it impractical for NISQ devices. An alternative approach is the optimal depth method [7], which prepares a quantum state with a depth of $O(n)$, where n is the number of qubits. Despite its reduced depth, this method requires $O(2^n)$ ancilla qubits, which remains impractical for near-term devices with limited resources.

To address these limitations, approximation methods have been developed to balance the precision of quantum data loading with the required resources. One such method is the tensor network-based approach, which approximates the quantum state using a tensor network representation. Ran *et al*. [8] employed a matrix product state (MPS), which is a one-dimensional tensor network, to represent the quantum state. This method iteratively truncates and disentangles the MPS, then constructs the quantum circuit by unitarizing and stacking the truncated MPS. Furthermore, Rudolph *et al*. [9] proposed gate-by-gate optimization on top of the MPS-based method to further improve fidelity.

However, these methods focus on loading a single data point at a time with high accuracy, which is not suitable for large-scale data loading tasks in QML. Therefore, we aim to develop a more efficient method that can handle large datasets with batch execution while maintaining high accuracy in data loading.

C. Parameterized Quantum Circuits (PQCs)

PQCs are a class of circuits characterized by adjustable parameters, which is suitable for quantum neural networks [11]. These circuits are typically constructed using rotation gates parameterized by angles, allowing them to represent a wide range of quantum operations. The optimization of these parameters is achieved by evaluating and minimizing a cost function, which can be defined as the expectation value of an observable.

In addition to their use as quantum neural networks, PQCs can also function as quantum data loaders. In this context, each data point has its own parameters, which are optimized to maximize the similarity between the quantum state and the corresponding classical data point [9]. However, this optimization process, performed for each individual data point, is computationally intensive for large datasets and is not feasible for unseen data points. To address this limitation, our work leverages classical deep neural networks to generalize the prediction of the parameters of PQCs, facilitating efficient quantum data loading.

D. Autoencoder

The autoencoder is a type of neural network architecture that learns to compress and reconstruct data, effectively extracting important features from the input [12]. An autoencoder consists of an encoder and a decoder, where the encoder compresses the input data into a low-dimensional latent vector, and the decoder reconstructs the input data from the latent vector. Autoencoders are commonly used for feature extraction [13] and representation learning [14]. In this work, autoencoders are used to extract latent features from classical data, which are then fed into a neural predictor to generate the parameters of a quantum circuit for data loading.

III. Proposed Methods

In this Section, we describe the proposed method for efficient quantum data loading. First, we formulate the problem of quantum data loading. Then, we introduce the overall framework of the proposed method, followed by a structure of the PQC used in our experiment. Finally, we discuss the training process of our model.

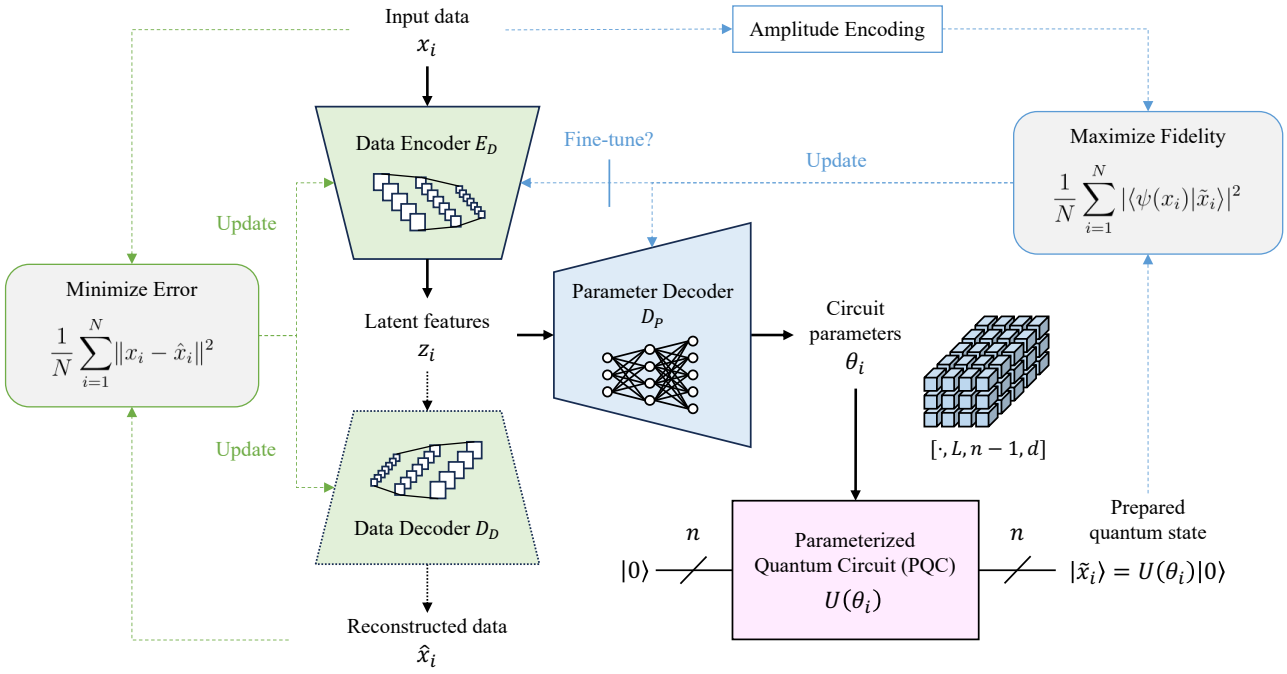


Fig. 2: Overall framework of proposed *Neural Circuit Parameter Prediction*. The data encoder E_D extracts latent features z from the raw data x, which are then used to predict a set of parameters θ for the parameter decoder D_P. E_D and D_P are trained with three steps: (i) autoencoder pre-training of E_D, (ii) prediction training of D_P, and (iii) fine-tuning of E_D and D_P. The quantum state is prepared using the parameterized quantum circuit U with the predicted parameter set θ, where BS is batch size and n, L, and d are the number of qubits, layers, and parameters per gate, respectively.

A. Problem Formulation

Consider a classical dataset $\mathcal{D} = \{x_1, x_2, \ldots, x_N\}$, where each x_i represents a classical data point and N denotes the total number of data points. The objective is to accurately encode data x_i into quantum state $|x_i\rangle$. This encoding is achieved by implementing a data-specific quantum circuit U_{x_i}, starting from the initial state $|0\rangle$. The resulting quantum state $U_{x_i}|0\rangle$ should closely approximate the ideal quantum state $|x_i\rangle$ in terms of fidelity. Fidelity is a measure of the similarity between two quantum states. Specifically, the fidelity between two pure states $|a\rangle$ and $|b\rangle$ is defined as $F(a,b) := |\langle a|b\rangle|^2$.

The circuit U_{x_i} is parameterized by a set of parameters θ_i, which determine the quantum state loaded by the circuit. The goal is to find the mapping $f : x \mapsto \theta$ that predicts the parameters $\theta_i := f(x_i)$ for all x_i in $\mathcal{D}$ that maximize the fidelity between the amplitude-encoded state $|\psi(x_i)\rangle$ and loaded state $U(\theta_i)|0\rangle$. The problem can be formulated as follows:

$$f^* = \arg\max_f \frac{1}{N}\sum_{i=1}^{N} |\langle\psi(x_i)|U(f(x_i))|0\rangle|^2, \qquad (2)$$

where f^* represents the optimal mapping. To solve this problem, we propose a machine learning-based approach, where the optimal mapping f^* is approximated by a neural network.

B. Overall Framework

The overall structure of the proposed method is illustrated in Fig. 2. The framework consists of two main components: the data encoder E_D and the parameter decoder D_P. The data encoder E_D extracts low-dimensional latent features z from the input data x. Subsequently, the parameter decoder D_P utilizes these features to predict a set of parameters θ. The parameter decoder D_P is implemented as a feed-forward neural network with multiple hidden layers. Finally, these predicted parameters θ are inserted into the PQC U, which prepares the desired quantum state $U(\theta)|0\rangle$.

The data encoder E_D can be implemented using various methods, such as autoencoders or principal component analysis (PCA). As depicted in Fig. 2, we employed a convolutional autoencoder as the data encoder, which has been demonstrated effectiveness for image data [15]. Additionally, we investigated the impact of different implementation methods on the performance of data loading.

C. PQC Architecture

In the proposed method, the PQC U works as a function rather than a trainable component. The structure of the circuit U is predefined and does not change during the training process. Any structure can be used for circuit U, depending on the specific task of quantum data loading. In our experiment, we employed a linear architecture for circuit U, as shown in Fig. 3, in order to compare with the previous work [8]. The n-qubit circuit consists of L layers, where each layer contains $n-1$ two-qubit gates.

We employed a two-qubit special unitary ($\mathrm{SU}(4)$) gate G as the fundamental component of the circuit U due to its superior trainability compared to other decomposed gates [16]. The $\mathrm{SU}(4)$ gate G is parameterized by 15 real-valued parameters θ. These parameters correspond to the two-qubit Pauli words except for the identity, i.e., $P = \{IX, IY, IZ, XI, \ldots, ZY, ZZ\}$, where I denotes the identity matrix, and X, Y, and Z denote the Pauli matrices. The gate G is defined as follows:

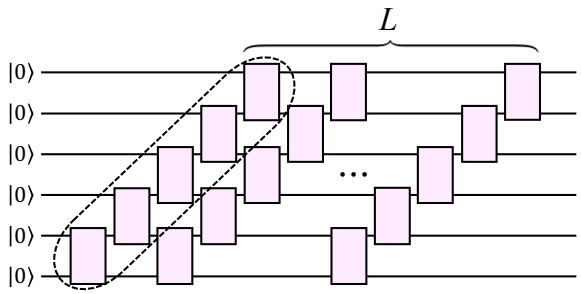


Fig. 3: Architecture of the PQC used in our experiment. We employed the same linear circuit architecture from the previous method proposed by Ran *et al*. [8] for comparison. This structure effectively approximates the higher dimensional matrix product state (MPS) by stacking multiple layers L.

$$G(\theta) = \exp \sum_{m=1}^{|P|} i\theta_m P_m. \tag{3}$$

For image data specifically, it is beneficial to use real-valued quantum gates, as images are inherently real-valued. Therefore, we propose a modification to the gate G to maintain the quantum state as real-valued throughout the circuit. This modification involves using only six out of 15 parameters for each gate, specifically those corresponding to the Pauli words in the set $P = \{IY, YI, XY, YX, YZ, ZY\}$. This adjustment ensures that the quantum state remains real-valued throughout the circuit, which is advantageous for image loading tasks.

D. Training Process

The training process of the proposed model involves three phases: (i) *pre-training* of the data encoder E_D, (ii) *prediction training* of the parameter decoder D_P, and (iii) *fine-tuning* of the entire model.

In the *pre-training* phase, the data encoder E_D is trained using an autoencoder methodology, where an auxiliary decoder D_D is introduced to facilitate the training of the encoder E_D. The autoencoder is optimized to minimize the mean squared error (MSE) between the input data x and the reconstructed data $D_D(E_D(x))$, as expressed by the following equation:

$$\min_{E_D, D_D} \frac{1}{N} \sum_{i=1}^{N} \|x_i - D_D(E_D(x_i))\|^2. \tag{4}$$

This optimization process allows the encoder to learn a latent representation $z = E_D(x)$ of the input data x that encapsulates the essential information necessary for recovering the data. The pre-training helps achieving enhanced performance and reducing the number of updates required in the subsequent training phases. It is important to note that alternative feature extraction techniques, such as PCA for simpler datasets or publicly available pre-trained models for more complex datasets, can be beneficial in improving model performance.

In the *prediction training* phase, the parameter decoder D_P is trained to predict circuit parameters $\theta = D_P(z)$, which achieves accurate data loading. The decoder D_P is trained without updating the data encoder E_D to avoid forgetting the learned knowledge. The training objective is to maximize the reconstruction fidelity, i.e., the fidelity between the loaded state $U(\theta)\,|0\rangle$ and the amplitude encoded state $|\psi(x)\rangle$, which is formulated as follows:

$$\max_{D_P} \frac{1}{N} \sum_{i=1}^{N} |\langle \psi(x_i) | U(D_P(E_D(x_i))) | 0 \rangle|^2. \tag{5}$$

In the *fine-tuning* phase, the entire network is fine-tuned to optimize the data loading fidelity. Both the data encoder E_D and the parameter decoder D_P are incrementally updated with a small learning rate to optimize them for the target task. This process aims to further improve reconstruction performance by addressing the suboptimality of the pre-trained encoder E_D. The fine-tuning process is formulated as follows:

$$\max_{E_D, D_P} \frac{1}{N} \sum_{i=1}^{N} |\langle \psi(x_i) | U(D_P(E_D(x_i))) | 0 \rangle|^2. \tag{6}$$

Here, the objective function remains the same as the Eq. (5), but the data encoder E_D and the parameter decoder D_P are optimized simultaneously. This three-phase training process allows the model to achieve high reconstruction fidelity while maintaining the efficiency of quantum data loading.

TABLE I: Implementation details of the models in the experiment.

Layers	Output Shape	Parameters
Data Encoder E_D		
Input	[·, 28, 28, 1]	-
Conv2D (shape=3x3, pad=1, stride=2)	[·, 14, 14, 16]	160
Conv2D (shape=3x3, pad=1, stride=2)	[·, 7, 7, 32]	4,640
Conv2D (shape=7x7)	[·, 1, 1, $\dim(z)$]	100,416
Reshape	[·, $\dim(z)$]	-
Parameter Decoder D_P		
Input	[·, $\dim(z)$]	-
Linear (in=$\dim(z)$, out=256)	[·, 256]	16,640
Linear (in=256, out=512)	[·, 512]	131,584
Linear (in=512, out=$54L$)	[·, $54\times L$]	$27{,}702\times L$
Reshape	[·, L, 9, 6]	-

IV. Experimental Results

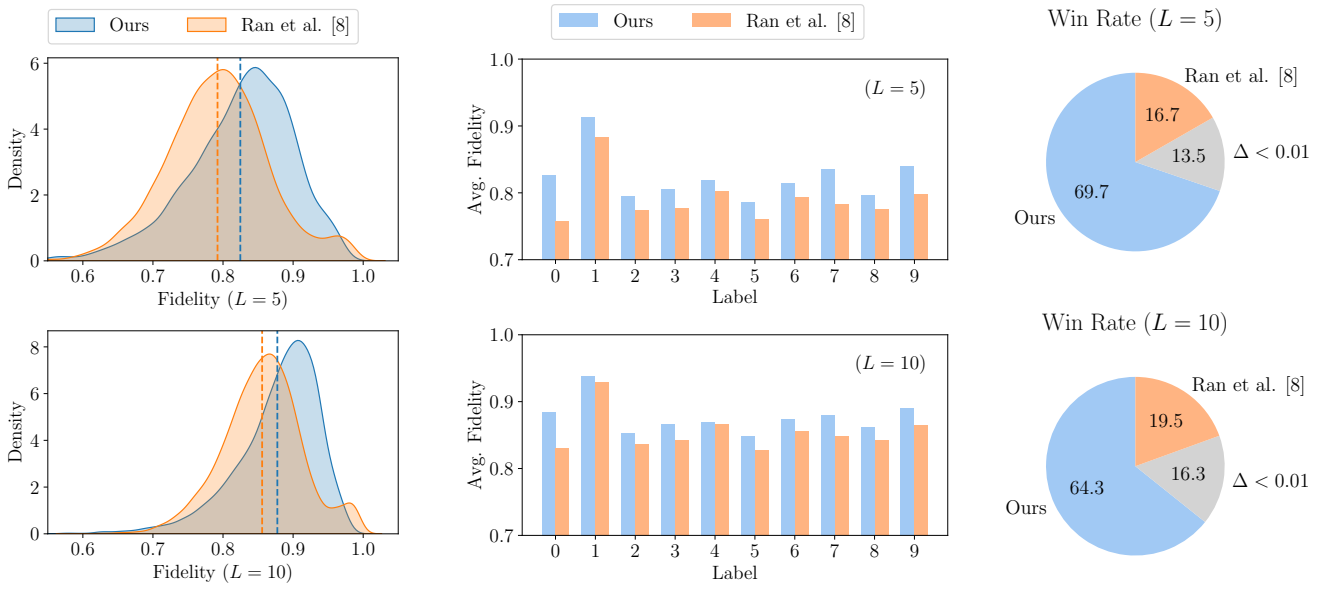
In this Section, we present the implementation details and the experimental results of the proposed method, followed by the ablation studies.

A. Experimental Setup

We conducted experiments on the MNIST dataset [10], which consists of 60,000 training images and 10,000 test images of handwritten digits. We used a quantum circuit with $L = 5, 10$ layers for the prediction target of the parameter decoder. We evaluated the reconstruction fidelity and runtime performance of our model. We used Pennylane [17] for implementing and simulating quantum circuits, and PyTorch [18] for training the model The experiments were conducted on a CentOS Linux 7.9 system with an Intel Xeon Gold 6132 CPU with 56 cores at 2.6 GHz and 256 GB RAM.

B. Model Implementation

Table I shows the implementation details of the proposed models employed in the experiment. The data encoder E_D is a convolutional neural network comprising three hidden



(a) Distribution of reconstruction fidelity

(b) Average fidelity for each label

(c) Win rate of each method

Fig. 4: Comparison of reconstruction performance on the MNIST Dataset [10]. (a) Kernel density estimation of the reconstruction fidelity distribution, with the vertical line indicating the mean value. (b) Average reconstruction fidelity for each label, demonstrating that the proposed method consistently outperforms the MPS-based method [8]. (c) Win rate of the proposed method over [8], showing higher fidelity for the majority of data points.

layers, designed to transform 28×28 grayscale images into one-dimensional vector z. Each layer is followed by a ReLU activation, except for the final layer. The data decoder D_D is built with the same architecture as the data encoder E_D, but in reverse order, replacing the convolution layers with transposed convolution layers. The parameter decoder D_P is a feed-forward neural network with three hidden layers. Each layer is followed by a SiLU activation [19], except for the final layer. The decoder output is reshaped into a $L \times (n-1) \times 6$ tensor, corresponding to the n-qubit circuit with L layers of real-valued gates G, as discussed in Section III-C. In Table I, the values for the output shape and the number of parameters are specified with $n = 10$ which is the minimum number of qubits required to encode 28×28 images.

C. Reconstruction Fidelity

Fig. 4 illustrates the comparison of reconstruction fidelity between the proposed method and the MPS-based method [8]. The fidelity distribution indicates that the proposed method achieves a higher average fidelity across all test set data (Fig. 4(a)). To further evaluate performance, we calculated the average fidelity for each label from 0 to 9 (Fig. 4(b)). The proposed method consistently outperforms the MPS-based method for all labels, demonstrating that the average fidelity is not biased towards specific classes. Additionally, the win rate for each data point is illustrated in Fig. 4(c), where the gray area represents instances where both methods have a fidelity difference of less than 0.01. This result shows that the proposed method achieves higher fidelity for over 64.3% and 69.7% of data points for $L = 5$ and $L = 10$, respectively. These results highlight the superior data loading accuracy while using the same amount of quantum resources.

TABLE II: Runtime comparison on the MNIST dataset. The runtime measured as the average value over the entire test dataset with 10 runs, presented in milliseconds along with standard deviation.

Methods	Runtime ($L = 5$)	Runtime ($L = 10$)
Ran *et al.* [8] (PyTorch [18])	36.9274 ± 0.8102 ($\times 108.39$)	65.5558 ± 1.2208 ($\times 187.57$)
Ran *et al.* [8] (Quimb [20])	42.5838 ± 2.4115 ($\times 124.99$)	83.8750 ± 4.5530 ($\times 239.99$)
Ours	0.3407 ± 0.0097	0.3495 ± 0.0126
Ours (batch size 100)	1.6366 ± 0.1849	1.7145 ± 0.1698

D. Runtime Comparison

To compare the runtime, we implemented the MPS-based method [8] using both PyTorch [18] and Quimb [20]. This dual implementation ensures a fair and comprehensive comparison, accounting for potential performance variations due to different libraries. We measured the time required to load a single image over the entire test dataset for both methods. The runtime was calculated as the average value over 10 runs. Table II indicate that the proposed method is 108 times faster than the MPS-based method for $L = 5$ and 187 times faster for $L = 10$. This significant improvement in runtime demonstrates the efficiency of the proposed method in handling large-scale quantum data loading tasks. Additionally, the runtime for $L = 5$ and $L = 10$ is similar in our method, whereas the runtime for $L = 10$ doubled in the MPS-based method. This is because the proposed method does not iterate over the layers but infers the parameters of the entire circuit at once. Moreover, we can further improve the runtime efficiency by leveraging the batch execution capability of the proposed method. For $L = 5$, processing a batch of 100 takes only 4.8 times longer, achieving a per-data-point speedup of $20\times$.

TABLE III: Ablation study for the pre-training and fine-tuning steps.

Pre-training	Fine-tuning	Infidelity ($L = 5$)	Infidelity ($L = 10$)
-	-	0.1566	0.1385
✓	-	0.0977	0.0691
✓	✓	**0.0932**	**0.0641**
Ran *et al.* [8]		0.1128	0.0771

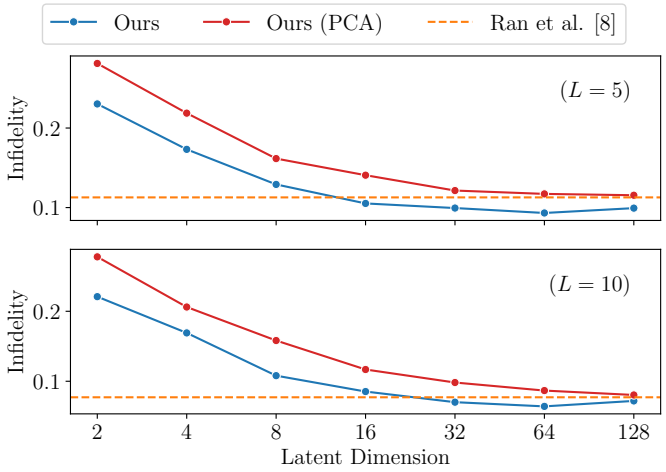


Fig. 5: Scaling of the infidelity scores for the latent dimension from 2 to 128 is presented, along with the results for the case using PCA as the data encoder.

E. Ablation Study

1) Training Phases: To assess the impact of pre-training and fine-tuning on the performance of the proposed method, we conducted an ablation study. We compared infidelity scores of the proposed method with and without these steps. The infidelity score is defined as $1 - \frac{1}{N}\sum_{i=1}^{N} |\langle x_i|\tilde{x}_i\rangle|$, where x_i is the ideal amplitude-encoded state and $\tilde{x}_i$ is the loaded state. We employed the infidelity score without the square operation, following the convention in [9]. The experimental results indicate that the proposed method achieved the lowest infidelity scores when both steps were applied (Table III). The pre-training step significantly enhanced the performance of the model, reducing the infidelity by 37.6% for $L = 5$ and 50.1% for $L = 10$, outperforming the MPS-based method [8]. The fine-tuning step further improved performance, reducing the infidelity by an additional 2.9% for $L = 5$ and 3.6% for $L = 10$. These results underscores the importance of both pre-training and fine-tuning in achieving high fidelity in quantum data loading tasks.

2) Latent Dimension Scaling: We analyzed the influence of the latent dimension on the proposed method. The latent dimension was varied from 2 to 128, and the infidelity scores of the model were evaluated for 30 epochs of training. Fig. 5 indicate that infidelity scores improve with an increase in the latent dimension. It suggests that the model benefits from a higher-dimensional latent space. However, the model begins to underfit as the latent dimension exceeds 64. This occurs because the model requires more training steps due to the increase in the number of parameters with the latent dimension. These results suggest that 64 is the optimal choice of latent dimension for training efficiency.

3) Feature Extraction Methods: As discussed in Section III-D, the data encoder E_D can be implemented using other feature extraction methods, such as PCA. From Fig. 5, we observed that the PCA encoder does not experience the same underfitting issue as the autoencoder because it has fewer parameters. However, the performance of the PCA encoder is significantly lower than the autoencoder. This suggests that PCA can be a viable alternative for intermediate-sized datasets, offering a good balance between performance and efficiency.

V. Conclusion

In this paper, we proposed a novel quantum data loading method using machine learning, termed *Neural Circuit Parameter Prediction*. Our method directly predicts the parameters of a quantum circuit from input data to improve runtime and accuracy. We introduced a parameterization strategy for real-valued data and a three-phase training process including autoencoder pre-training and fine-tuning to enhance the data loading accuracy. Our method outperformed the previous state-of-the-art method, demonstrating a 17.31% improvement in infidelity score and 108 times faster runtime for the MNIST dataset. Our results highlight the efficiency of our approach in facilitating practical advantage for QML tasks.

References

[1] J. Biamonte *et al.*, "Quantum machine learning," *Nature*, vol. 549, no. 7671, pp. 195–202, 2017.

[2] K. Bharti *et al.*, "Noisy intermediate-scale quantum algorithms," *Reviews of Modern Physics*, vol. 94, no. 1, p. 015004, 2022.

[3] J. R. McClean *et al.*, "The theory of variational hybrid quantum-classical algorithms," *New Journal of Physics*, vol. 18, no. 2, p. 023023, 2016.

[4] M. Hibat-Allah *et al.*, "A framework for demonstrating practical quantum advantage: comparing quantum against classical generative models," *Communications Physics*, vol. 7, no. 1, p. 68, 2024.

[5] M. Schuld *et al.*, "Effect of data encoding on the expressive power of variational quantum-machine-learning models," *Physical Review A*, vol. 103, no. 3, p. 032430, 2021.

[6] M. Mottonen *et al.*, "Transformation of quantum states using uniformly controlled rotations," *arXiv preprint quant-ph/0407010*, 2004.

[7] X.-M. Zhang *et al.*, "Quantum state preparation with optimal circuit depth: Implementations and applications," *Physical Review Letters*, vol. 129, no. 23, p. 230504, 2022.

[8] S.-J. Ran, "Encoding of matrix product states into quantum circuits of one-and two-qubit gates," *Physical Review A*, vol. 101, no. 3, p. 032310, 2020.

[9] M. S. Rudolph *et al.*, "Decomposition of matrix product states into shallow quantum circuits," *Quantum Science and Technology*, vol. 9, no. 1, p. 015012, 2023.

[10] Y. LeCun *et al.*, "Mnist handwritten digit database," *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, vol. 2, 2010.

[11] M. Benedetti *et al.*, "Parameterized quantum circuits as machine learning models," *Quantum Science and Technology*, vol. 4, no. 4, p. 043001, 2019.

[12] M. A. Kramer, "Nonlinear principal component analysis using autoassociative neural networks," *AIChE journal*, vol. 37, no. 2, pp. 233–243, 1991.

[13] P. Vincent *et al.*, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th international conference on Machine learning*, 2008, pp. 1096–1103.

[14] P. Vincent *et al.*, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion." *Journal of machine learning research*, vol. 11, no. 12, 2010.

[15] J. Masci *et al.*, "Stacked convolutional auto-encoders for hierarchical feature extraction," in *Proceedings of the 21th International Conference on Artificial Neural Networks - Volume Part I*, 2011, pp. 52–59.

[16] R. Wiersema *et al.*, "Here comes the su (n): multivariate quantum gates and gradients," *Quantum*, vol. 8, p. 1275, 2024.

[17] V. Bergholm *et al.*, "Pennylane: Automatic differentiation of hybrid quantum-classical computations," *arXiv preprint arXiv:1811.04968*, 2018.

[18] A. Paszke *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.

[19] P. Ramachandran *et al.*, "Searching for activation functions," *arXiv preprint arXiv:1710.05941*, 2017.

[20] J. Gray, "quimb: a python library for quantum information and many-body calculations," *Journal of Open Source Software*, vol. 3, no. 29, p. 819, 2018.