

Application of Quantum Neural Networks and Variational Quantum Regression in Seismic Inversion

Zhen Liu*, Junhua Zhang. China University of Petroleum(East China)

Summary

This study explores the application of quantum neural networks (QNN) and variational quantum regression (VQR) in seismic inversion, leveraging quantum computing's advantages in superposition, entanglement, and interference. Implemented using variational quantum circuits (VQC) and tested on the Qiskit simulator, QNN captures complex seismic data relationships, while VQR focuses on regression tasks. Results demonstrate their potential in geophysical applications but highlight challenges in model generalization and scalability. As quantum computing advances beyond the NISQ era, quantum intelligence is expected to provide new insights for efficient seismic inversion.

Introduction

Quantum intelligence, an emerging field integrating quantum computing and artificial intelligence, is rapidly becoming a revolutionary approach to tackling large-scale computation and complex optimization problems (Biamonte et al., 2017). This study explores the preliminary application of quantum neural networks (QNN) and variational quantum regression (VQR) in seismic inversion. Seismic inversion is a core technique in geophysics, used to infer elastic parameters from seismic data for reservoir prediction. Traditional seismic inversion methods often require enormous computational resources to process data and construct subsurface models. As geological models grow in complexity and the demand for high-resolution imaging increases, the development of more efficient computational methods has become increasingly urgent. Quantum computing, leveraging the principles of qubit superposition, entanglement, and interference, offers powerful parallel processing capabilities, providing a groundbreaking alternative to conventional methods (Lloyd, 1996; Preskill, 2018).

In this study, we evaluate the performance of quantum neural networks (QNN) and variational quantum regression (VQR) in seismic inversion tasks. QNN, a quantum counterpart of classical neural networks, captures complex relationships in data through parameterized quantum circuits (Cerezo et al., 2021), while VQR is specifically designed for regression tasks, optimizing quantum circuits to predict target values. Both methods are based on variational quantum circuits (VQC), integrating quantum feature encoding with classical optimization techniques for model training. Experiments were conducted using the Qiskit quantum simulation framework on classical computers (Qiskit Machine Learning

Documentation, 2025), and the results indicate that QNN and VQR can effectively fit the seismic inversion process, though challenges remain in terms of model generalization and scalability. As the first attempt to introduce quantum intelligence into seismic inversion, this study demonstrates the potential advantages of this technology in geophysics while also highlighting its current limitations. With continuous advancements in quantum hardware—particularly in error correction and the transition from the noisy intermediate-scale quantum (NISQ) era to fault-tolerant quantum computing (IBM, 2016; Zhong et al., 2020)—we believe that quantum intelligence has the potential to become a powerful tool for solving geophysical inverse problems in the future.

Principles of Quantum Intelligence

Both QNN and VQR essentially rely on Variational Quantum Circuits (VQC), also known as Parameterized Quantum Circuits (PQC). The main idea is to construct quantum states using parameterized quantum gates and train the parameters using classical optimization algorithms, making them suitable for specific tasks such as regression prediction (Hadfield et al., 2019).

VQC consists of the following key components:

(1) Data Encoding Layer (Quantum Feature Encoding): Classical data $X \in R^d$ need to be mapped to quantum states, typically using amplitude encoding (Amplitude Encoding) or parameterized rotation gates (Nielsen & Chuang, 2011):

$$U_{enc}(X)|0\rangle^{\otimes n} = |\psi_{enc}(X)\rangle \quad (1)$$

Here, $U_{enc}(X)$ is the quantum gate used for data encoding.

(2) Parameterized Quantum Circuit (Variational Quantum Circuit, VQC): The variational quantum circuit consists of multiple parameterized rotation gates $R(\theta)$ and entangling gates (such as CNOT):

$$U(\theta) = \prod_{l=1}^L U_l(\theta_l) \quad (2)$$

Here, θ is the trainable parameter.

(3) Measurement and Expectation Computation: The observable $\hat{M}$ is measured, and the expectation value is computed as the predicted value (Adedoyin et al., 2018).

$$f(x; \theta) = \langle \psi_{enc}(X) | U^\dagger(\theta) \hat{M} U(\theta) | \psi_{enc}(X) \rangle \quad (3)$$

(4) Classical Optimization: The parameters θ are adjusted by optimizing the loss function $L(\theta)$:

$$\theta^* = \arg \min_{\theta} L(\theta) \quad (4)$$

Classical optimizers such as Adam and SPSA (Simultaneous Perturbation Stochastic Approximation) can be chosen.

QNN (Quantum Neural Network) simulates the behavior of classical neural networks through variational quantum circuits and can be used for regression tasks. Its main features are as follows:

- (1) It uses multi-layer variational quantum circuits, where each layer is similar to the 'hidden layer' of a classical neural network.
- (2) After the quantum state undergoes multiple transformations, the measurement results are used to compute the loss function and optimize the circuit parameters.
- (3) QNN can be purely quantum or hybrid quantum-classical (HQC).

The training process of QNN is as follows:

- (1) Input data mapping: The input data xxx is transformed into a quantum state through quantum feature mapping (Equation 1).

- (2) Variational quantum circuit computation:

$$|\psi_{out}(x; \theta)\rangle = U(\theta)|\psi_{enc}(x)\rangle \quad (5)$$

- (3) Measurement and Regression Output:

$$y_{pred} = f(x; \theta) = \langle \psi_{out}(x; \theta) | \hat{M} | \psi_{out}(x; \theta) \rangle \quad (6)$$

- (4) Compute the loss function (e.g., Mean Squared Error, MSE):

$$L(\theta) = \frac{1}{N} \sum_{i=1}^N (y_{true,i} - y_{pred,i})^2 \quad (7)$$

- (5) Optimize parameters: Classical optimization methods are used to update the parameters (Cerezo et al., 2021):

$$\theta \leftarrow \theta - \eta \nabla_{\theta} L(\theta) \quad (8)$$

VQR is a variational quantum circuit specifically designed for regression tasks, with the goal of finding a quantum circuit such that its measurement results match the target regression values. Compared to QNN, VQR focuses more on:

- (1) The design of the quantum feature map to ensure nonlinear mapping capability.
- (2) The optimization of the quantum circuit architecture to make the output values align better with the target regression distribution.

The training process of VQR is similar to that of QNN, but its objective is more focused on regression fitting:

- (1) Input data encoding (Equation 1).
- (2) Variational quantum circuit processing (Equation 5).
- (3) Measurement of the regression output:

$$y_{pred} = \sum_k \omega_k \langle \psi_{out}(x; \theta) | \hat{M}_k | \psi_{out}(x; \theta) \rangle \quad (9)$$

Where ω_k is the linear weight, and $\hat{M}_k$ represents the different measurement operators.

- (4) Compute the regression loss (Equation 7).
- (5) Optimize the variational parameters (Equation 8) (Preskill, 2018).

Experimental Results on Real Data

Figure 1 shows the data and labels used for training QNN and VQR. The data comes from a synthetic record created by convolving data from the Marmousi2 model with a 15Hz central frequency Ricker wavelet, and then normalizing it. The labels are derived from the inversion of the P-wave velocity data using a Physics-Informed Neural Network (PINN). During label creation, it was considered that the actual P-wave velocity variations in the Marmousi2 model were too sharp and would hinder the training of QNN and VQR, so the inversion results were used for training.

Figure 2 illustrates the QNN architecture, depicting the quantum state flow. The circuit begins with an input passing through a classical rotation gate R , followed by a $U2$ gate with parameters $(0, \pi)$ for phase transformations. Next, rotation gates R_y and R_z apply learnable parameters $\theta[0]$ to $\theta[3]$, dynamically adjusted during training. These operations transform the quantum state on the Bloch sphere, extracting features. The right side represents the output, demonstrating how QNN maps classical inputs to quantum space, processes features via quantum operations, and generates outputs—leveraging quantum superposition, entanglement, and interference for efficient data processing.

Figure 3 shows the convergence curve of QNN training. The objective function value drops rapidly in the first few iterations and stabilizes after the second iteration, indicating that the model quickly learns in the early stages and converges to a relatively low error level. This suggests that the optimization process of QNN is efficient, allowing the model to fit the training data rapidly. The convergence curve is smooth without significant oscillations, demonstrating a stable training process without issues such as overfitting or gradient explosion.

Figure 4 compares QNN prediction results. Figure 4a represents the training set, while the Figure 4b represents the test set. In the training set, the predicted values (red dashed line) closely match the actual values (black solid line), though some deviations exist in certain depth intervals, potentially due to the distribution of training data or the model's generalization ability. In the test set, the agreement between predicted and actual values slightly declines, indicating that the model performs slightly worse on unseen data. However, the overall trend and key features remain consistent. This suggests that QNN possesses a certain level of generalization capability but requires further optimization in finer details.

Figure 5 shows the VQR network structure diagram. The input quantum state is first encoded through R_y which maps the classical data x into the quantum space. Subsequently,

the quantum state enters the "EfficientSU2" module, a highly efficient parameterized quantum circuit containing the parameters $\theta[0]$, $\theta[1]$, $\theta[2]$, $\theta[3]$, which are adjusted through classical optimization algorithms to enhance model performance. The entire architecture clearly presents the data processing flow from input to output.

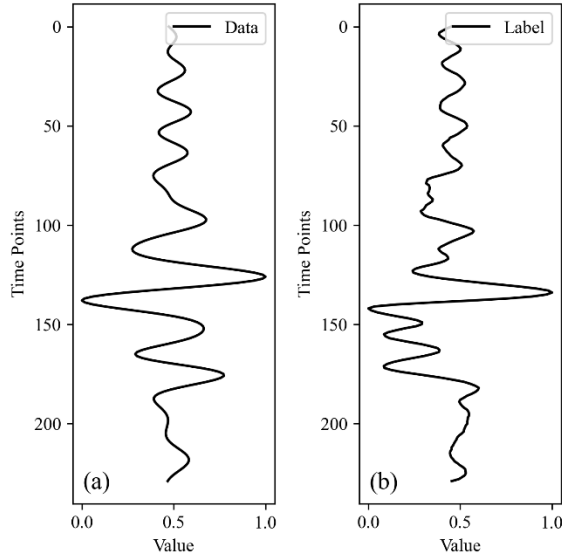


Figure 1: Training data(a) and training label(b).

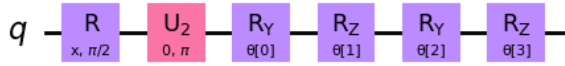


Figure 2: QNN network architecture.

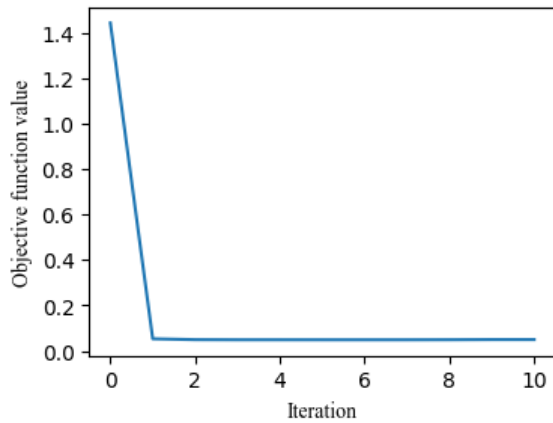


Figure 3: Convergence curve for QNN training.

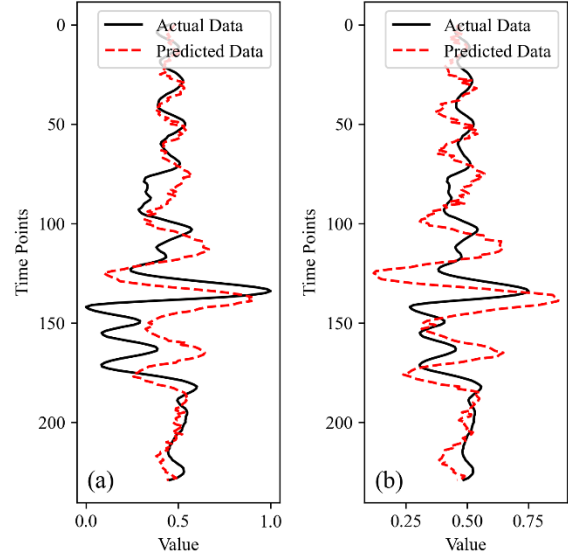


Figure 4: Comparison of QNN prediction results. (a) Training dataset. (b) Test dataset.



Figure 5: VQR network structure.

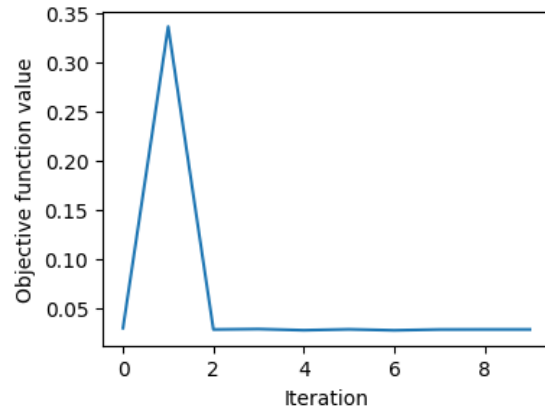


Figure 6: VQR training convergence curve.

Figure 6 shows the variation of the objective function value with the number of iterations. In the early stages of the iterations, the objective function value rises rapidly, peaks between the first and second iterations, and then drops sharply, stabilizing at a lower level in subsequent iterations. This indicates that during the training process, the algorithm makes significant adjustments to the model parameters in the

early stage, causing large fluctuations in the objective function value. As the iterations progress, the algorithm gradually converges, and the objective function value stabilizes, suggesting that the model reached a relatively stable state in the later stages of training.

Figure 7a shows the training set, and Figure 7b shows the test set. Both graphs display a comparison of actual data and predicted data. From the graphs, it is evident that, whether in the training or test set, the predicted data (red dashed line) and the actual data (black solid line) share a similar trend, but some deviations still exist. In the training set, the predicted data fits the actual data well in certain areas but deviates in some highly fluctuating regions. Similarly, in the test set, there is a comparable situation, suggesting that while the model can capture the general trend of the data, there is room for improvement in precision. There may be some overfitting or underfitting, and further optimization of the model parameters or structure may be required.

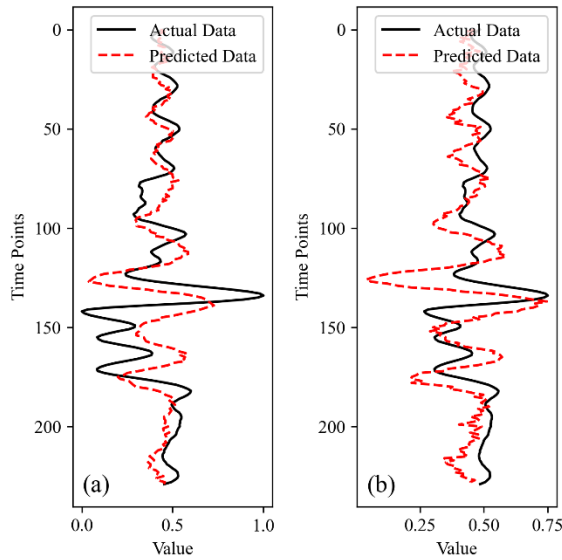


Figure 7: Comparison of VQR prediction results. (a) Training dataset. (b) Test dataset.

Overall, as shown in Figure 3, QNN's objective function value starts at about 1.4, drops rapidly, and stabilizes below 0.1 around the second iteration, indicating fast and stable convergence. Figure 6 shows that VQR's objective function

value starts lower at 0.05, rises quickly to 0.35, peaks, and then drops, stabilizing around the second iteration. Compared to QNN, VQR exhibits more fluctuation before convergence.

Figure 4a (QNN training data) and Figure 7a (VQR training data) both show a reasonable fit between predicted (red dashed line) and actual values (black solid line). QNN captures details well but deviates in regions with large fluctuations, while VQR aligns with the overall trend but shows discrepancies in certain areas.

For test data (Figures 4b and 7b), both QNN and VQR follow the general trend but show deviations at key turning points and highly fluctuating regions. QNN's performance declines slightly compared to its training results, while VQR maintains stable performance but with some local errors.

In summary, QNN converges faster and more smoothly, while VQR exhibits a more complex training process. Both capture data trends but require further optimization for improved precision, with no significant advantage observed between them in prediction performance.

Conclusion

Experimental results show that both QNN and VQR exhibit strong fitting capabilities and some generalization ability in regression tasks, though prediction accuracy can be improved. QNN converges rapidly, stabilizing after a few iterations, indicating efficient and stable optimization. It fits training data well but shows slight performance degradation on the test set, highlighting the need for better generalization. VQR has a more complex convergence pattern, with initial fluctuations before stabilization. While its final accuracy is comparable to QNN, deviations appear in regions with large local variations. Overall, QNN excels in convergence speed and stability, while VQR effectively captures trends, with no clear superiority between the two.

This study pioneers the application of QNN and VQR in seismic inversion, demonstrating the feasibility of quantum intelligence despite current hardware limitations. While conducted on classical simulations, the results highlight the potential of quantum computing for efficient seismic inversion. Future research should refine model structures, optimization algorithms, and parameter tuning to enhance their performance.