Trabalho

Título em Português:	Framework Computacional para Implementação e Análise de Redes Neurais Quânticas Aplicadas à Classificação de Portas Lógicas
Título em Inglês:	a computational framework for the implementation and analysis of quantum neural networks applied to logic gate classification
Autor:	Pedro Calligaris Delbem
Instituição:	Universidade de São Paulo
Unidade:	Instituto de Física de São Carlos
Orientador:	Filippo Giovanni Ghiglieno
Área de Pesquisa / SubÁrea:	Áreas Clássicas de Fenomenologia e suas Aplicações
Agência Financiadora:	FUSP - Fundação Universidade de São Paulo

Framework Computacional para Implementação e Análise de Redes Neurais Quânticas Aplicadas à Classificação de Portas Lógicas

Pedro C. Delbem, Lucca Rodrigues Cunha,
Bernardo Maia Coelho, Rodrigo Silva de Almeida

pedrodelbem@usp.br

Orientador: Prof. Dr. Alexandre Delbem e Prof. Dr. Filippo Ghiglieno

Universidade de São Paulo (USP)

25 de Agosto de 2025

Introdução

A área de aprendizagem de máquina, e em particular as redes neurais artificiais, revolucionou inúmeros campos da ciência e tecnologia. A unidade fundamental dessas redes é o neurônio artificial, ou perceptron, um modelo matemático concebido para imitar o seu análogo biológico. No entanto, já em 1969, Minsky e Papert demonstraram que um único perceptron é inerentemente limitado a resolver problemas linearmente separáveis, sendo incapaz de aprender funções básicas como o "OU exclusivo" (XOR) [3]. Esta limitação impulsionou o desenvolvimento de redes com múltiplas camadas (deep learning), que, embora poderosas, implicam um elevado custo computacional.

A computação quântica apresenta um paradigma fundamentalmente novo para o processamento de informação. A sua unidade básica, o *qubit*, difere de um bit clássico de forma crucial: enquanto um bit assume estados discretos de 0 ou 1, um qubit pode existir numa superposição linear de ambos os estados simultaneamente. Esta capacidade permite que os algoritmos quânticos explorem um espaço computacional fundamentalmente diferente.

Neste contexto, as Redes Neurais Quânticas (QNNs) surgem como um campo promissor, que busca unir o poder da computação quântica com a arquitetura das redes neurais. Uma QNN codifica informações em parâmetros de circuitos quânticos — tipicamente, os ângulos de rotação de portas quânticas. Uma das vantagens teóricas mais notáveis é que um único neurônio quântico pode, em princípio, resolver problemas não linearmente separáveis como o XOR, uma tarefa que requer múltiplas camadas no domínio clássico [1].

Este trabalho apresenta um framework computacional desenvolvido em Python para a implementação, treinamento e análise sistemática de arquiteturas de QNNs baseadas em um único neurônio. Utilizamos a tarefa de classificação de portas lógicas como um problema de benchmark para avaliar e comparar diferentes estratégias de codificação de dados e algoritmos de otimização.

Métodos e Procedimentos

A metodologia abrange a construção de um pipeline experimental, desde a definição programática do circuito quântico até a análise de desempenho.

Arquitetura do Neurônio Quântico. O modelo de neurônio quântico implementado utiliza um circuito parametrizado que atua sobre um ou mais qubits de entrada e um qubit de saída para a classificação. A computação é realizada através da aplicação de portas de rotação, cujos ângulos são os parâmetros treináveis do modelo, e portas de controle múltiplo para gerar a saída. O framework foi construído sobre a biblioteca Qiskit [2], permitindo simulações de alta performance e a potencial execução em hardware quântico.

Estratégias de Codificação de Dados. Duas abordagens para codificar os dados de entrada no estado quântico foram implementadas e analisadas através de módulos de software dedicados:

- **Codificação na Fase:** Os valores de entrada binários $\{0, 1\}$ são mapeados para $\{+1, -1\}$ e usados para modular os ângulos de rotação de portas quânticas (e.g., $R_Z(\theta_1 \cdot x_1)$, $R_X(\theta_2 \cdot x_2)$). A informação é, portanto, armazenada na fase relativa do estado quântico.
- **Codificação na Amplitude:** Nesta abordagem, o estado inicial dos qubits de entrada é preparado para corresponder ao vetor de entrada (ex: $|01\rangle$ para a entrada $(0, 1)$). Portas de rotação com ângulos treináveis (θ_i) são então aplicadas a cada qubit. A classificação

é realizada por uma porta de controle múltiplo (MCX) que atua sobre um qubit ancilla. Cada vetor de entrada é avaliado numa execução de circuito separada.

Framework de Treinamento e Otimização. O processo de treinamento, gerido pela classe principal do framework, visa otimizar os parâmetros θ_i do circuito para minimizar uma função de custo. A função de custo é definida como o erro entre a probabilidade de medição do estado de saída desejado e o valor esperado pela porta lógica, promediado sobre todas as entradas possíveis. Um ficheiro de configuração define os hiperparâmetros de cada experimento, incluindo as portas lógicas (XOR, AND, etc.), as codificações e os métodos de treino. Para os resultados aqui apresentados, foi utilizado o método de busca exaustiva em grade (`cg-exhaustive_search`) para garantir uma exploração completa do espaço de parâmetros.

O critério de convergência é definido de forma rigorosa: para garantir que o erro para cada combinação de entrada individual seja inferior a 0.5 (assegurando uma classificação correta na maioria das vezes), a tolerância para o erro médio total (E_{total}) deve ser estritamente menor que $0.5/2^n$, onde n é o número de qubits de entrada. Para os experimentos com $n = 2$, esta tolerância foi, portanto, definida como 0.125 ou 12.5%.

Resultados

Os experimentos foram executados através de um notebook computacional que gerencia a execução paralela das simulações para cada combinação de parâmetros definida. Os dados de convergência foram salvos e posteriormente processados por um script de visualização para gerar as análises gráficas.

A Figura 1 apresenta um heatmap que resume o desempenho comparativo das duas codificações, exibindo a média de iterações necessárias para atingir a convergência (erro abaixo da tolerância de 12.5%). A análise quantitativa revela que a codificação de fase foi, em geral, mais eficiente para problemas não lineares como XOR e XNOR. Em contraste, a codificação de amplitude que não apresentou convergência para nenhuma da portas lógicas.

As curvas de convergência, que detalham a evolução do erro ao longo das iterações, corroboram estes achados. Observou-se que, para neurônios simples apenas a codificação na fase foi capaz de aprender alguma porta lógica e apenas as lineramente não separaveis, em contraposição ao resultado de redes neurais clássicas.

Conclusões

Este trabalho demonstrou o desenvolvimento de um framework computacional versátil para a prototipagem e avaliação de Redes Neurais Quânticas. A plataforma permitiu uma comparação sistemática entre diferentes arquiteturas de codificação de dados, confirmando a capacidade de um único neurônio quântico de resolver problemas não linearmente separáveis, em conformidade com achados prévios na literatura [1].

Os resultados indicam que a escolha da estratégia de codificação é um fator determinante na eficiência do treinamento, com diferentes codificações sendo mais adequadas para classes distintas de problemas. O framework estabelecido serve como uma base sólida para trabalhos futuros, que podem incluir a expansão para arquiteturas com múltiplos neurônios, a exploração de codificações mais complexas e a avaliação do impacto do ruído em processadores quânticos reais.

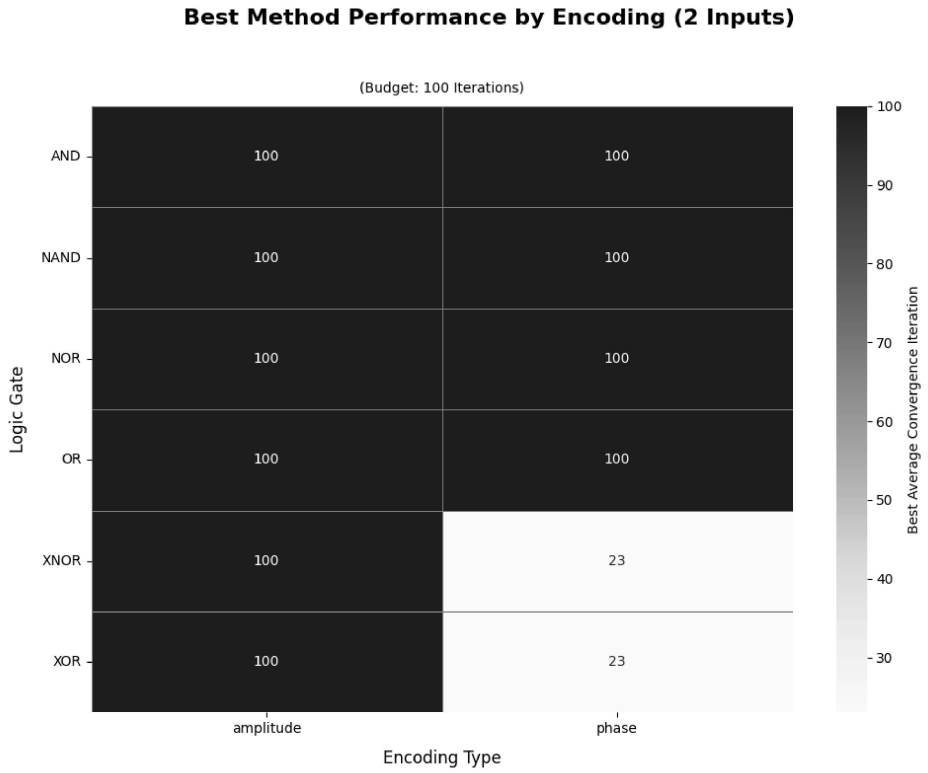

Figure 1: Heatmap comparando o número médio de iterações para convergência entre as codificações de Fase e Amplitude para diferentes portas lógicas. Valores menores (tons mais escuros) indicam melhor desempenho. A imagem é uma representação ilustrativa dos resultados.

Declaração de Conflito de Interesses: Os autores declaram não haver conflito de interesses.

Agradecimentos: Os autores agradecem ao Centro de Inteligência Artificial (C4AI-USP), pelo apoio da Fundação de Amparo à Pesquisa do Estado de São Paulo (FUSP, processo nº 2019/07665-4), da IBM Corporation, do Centro de Ciências Matemáticas Aplicadas à Indústria (CeMEAI, FAPESP, processo nº 2013/07375-0) e do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq).

References

[1] I.V. Grossu, "Single qubit neural quantum circuit for solving Exclusive-OR", *MethodsX*, vol. 8, p. 101573, 2021.

[2] A. Asfaw, L. Bello, Y. Ben-Haim et al., "Learn quantum computation using Qiskit", 2020. [Online]. Disponível em: `http://community.qiskit.org/textbook`.

[3] M. Minsky and S. Papert, "Perceptrons: An Introduction to Computational Geometry", MIT Press, Cambridge, 1969.

A Computational Framework for the Implementation and Analysis of Quantum Neural Networks Applied to Logic Gate Classification

Pedro C. Delbem, Lucca Rodrigues Cunha,
Bernardo Maia Coelho, Rodrigo Silva de Almeida

pedrodelbem@usp.br

Supervisor: Prof. Dr. Alexandre Delbem and Prof. Dr. Filippo Ghiglieno

University of São Paulo (USP)

August 25, 2025

Objectives

The field of machine learning, and particularly artificial neural networks, has revolutionized numerous fields of science and technology. The fundamental unit of these networks is the artificial neuron, or perceptron, a mathematical model designed to mimic its biological analogue. However, as early as 1969, Minsky and Papert demonstrated that a single perceptron is inherently limited to solving linearly separable problems, being incapable of learning basic functions such as the "exclusive OR" (XOR) [3]. This limitation drove the development of multi-layered networks (deep learning), which, although powerful, entail a high computational cost.

Quantum computing presents a fundamentally new paradigm for information processing. Its basic unit, the qubit, differs crucially from a classical bit: while a bit assumes discrete states of 0 or 1, a qubit can exist in a linear superposition of both states simultaneously. This capability allows quantum algorithms to explore a fundamentally different computational space.

In this context, Quantum Neural Networks (QNNs) emerge as a promising field that seeks to merge the power of quantum computing with the architecture of neural networks. A QNN encodes information into the parameters of quantum circuits—typically, the rotation angles of quantum gates. One of the most notable theoretical advantages is that a single quantum neuron can, in principle, solve non-linearly separable problems like XOR, a task that requires multiple layers in the classical domain [1].

This work presents a computational framework developed in Python for the implementation, training, and systematic analysis of QNN architectures based on a single neuron. We use the task of classifying logic gates as a benchmark problem to evaluate and compare different data encoding strategies and optimization algorithms.

Materials and Methods

The methodology covers the construction of an experimental pipeline, from the programmatic definition of the quantum circuit to the performance analysis.

Quantum Neuron Architecture. The implemented quantum neuron model uses a parameterized circuit that acts on one or more input qubits and one output qubit for classification. The computation is performed by applying rotation gates, whose angles are the trainable parameters of the model, and multi-control gates to generate the output. The framework was built on the Qiskit library [2], enabling high-performance simulations and the potential for execution on quantum hardware.

Data Encoding Strategies. Two approaches to encode input data into the quantum state were implemented and analyzed through dedicated software modules:

- **Phase Encoding:** The binary input values $\{0, 1\}$ are mapped to $\{+1, -1\}$ and used to modulate the rotation angles of quantum gates (e.g., $R_Z(\theta_1 \cdot x_1)$, $R_X(\theta_2 \cdot x_2)$). The information is therefore stored in the relative phase of the quantum state.
- **Amplitude Encoding:** In this approach, the initial state of the input qubits is prepared to correspond to the input vector (e.g., $|01\rangle$ for the input $(0, 1)$). Rotation gates with trainable angles (θ_i) are then applied to each qubit. The classification is performed by a multi-controlled X (MCX) gate acting on an ancillary qubit. Each input vector is evaluated in a separate circuit execution.

Training and Optimization Framework. The training process, managed by the framework's main class, aims to optimize the circuit parameters θ_i to minimize a cost function. The cost function is defined as the error between the measurement probability of the desired output state and the value expected by the logic gate, averaged over all possible inputs. A configuration file defines the hyperparameters for each experiment, including the logic gates (XOR, AND, etc.), encodings, and training methods. For the results presented here, the exhaustive grid search method (`cg-exhaustive_search`) was used to ensure a complete exploration of the parameter space. The convergence criterion is strictly defined: to ensure that the error for each individual input combination is less than 0.5 (ensuring correct classification most of the time), the tolerance for the total average error (E_{total}) must be strictly less than $0.5/2^n$, where n is the number of input qubits. For the experiments with $n = 2$, this tolerance was therefore set to 0.125 or 12.5%.

Results

The experiments were executed via a computational notebook that managed the parallel execution of simulations for each defined parameter combination. The convergence data was saved and subsequently processed by a visualization script to generate the graphical analyses.

Figure 1 presents a heatmap summarizing the comparative performance of the two encodings, displaying the average number of iterations required to reach convergence (error below the 12.5% tolerance). The quantitative analysis reveals that phase encoding was generally more efficient for non-linear problems like XOR and XNOR. In contrast, amplitude encoding did not converge for any of the logic gates. The convergence curves, which detail the error evolution over iterations, corroborate these findings. It was observed that for simple neurons, only phase encoding was able to learn any logic gate, and only the non-linearly separable ones, in contrast to the results from classical neural networks.

Conclusions

This work demonstrated the development of a versatile computational framework for the prototyping and evaluation of Quantum Neural Networks. The platform allowed for a systematic comparison between different data encoding architectures, confirming the ability of a single quantum neuron to solve non-linearly separable problems, in accordance with previous findings in the literature [1].

The results indicate that the choice of encoding strategy is a determining factor in training efficiency, with different encodings being more suitable for distinct classes of problems. The established framework serves as a solid foundation for future work, which may include expanding to multi-neuron architectures, exploring more complex encodings, and evaluating the impact of noise on real quantum processors.

Conflict of Interest Statement: The authors declare no conflict of interest.

Acknowledgements: The authors of this work would like to thank the Center for Artificial Intelligence (C4AI-USP), the support from the São Paulo Research Foundation (FAPESP grant nº 2019/07665-4), from the IBM Corporation, the Center for Mathematical Sciences Applied do Industry (CeMEAI, FAPESP grant nº 2013/07375-0), and the National Council for Scientific and Technological Development (CNPq).

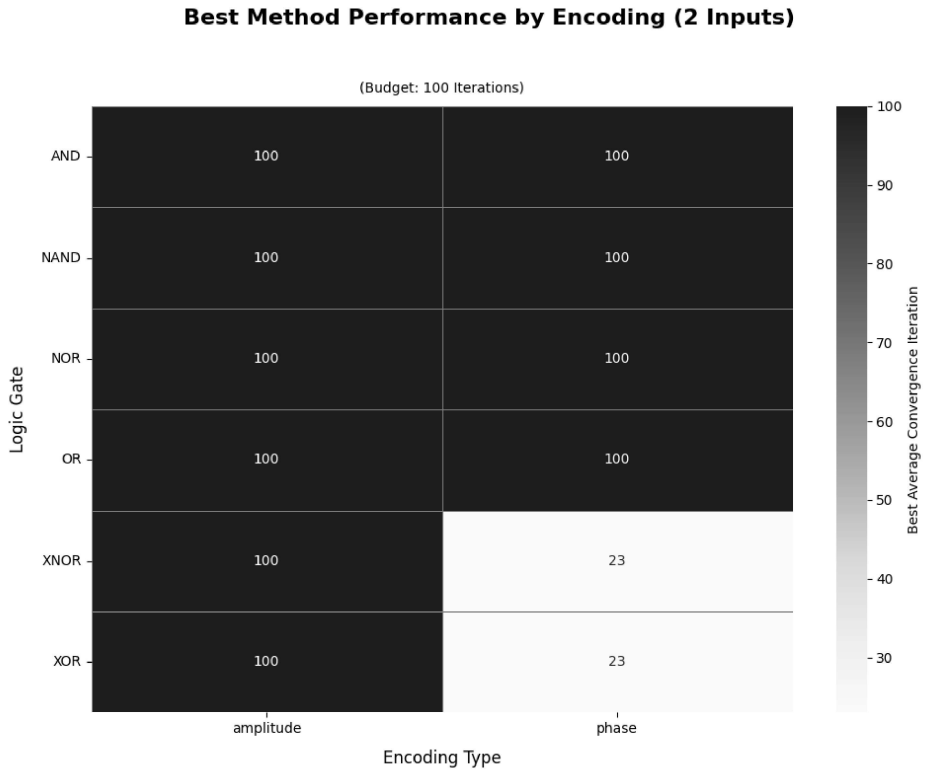

Figure 1: Heatmap comparing the average number of iterations to convergence between the Phase and Amplitude encodings for different logic gates. Lower values (darker tones) indicate better performance. The image is an illustrative representation of the results.

References

[1] I.V. Grossu, "Single qubit neural quantum circuit for solving Exclusive-OR", *MethodsX*, vol. 8, p. 101573, 2021.

[2] A. Asfaw, L. Bello, Y. Ben-Haim et al., "Learn quantum computation using Qiskit", 2020. [Online]. Available: `http://community.qiskit.org/textbook`.

[3] M. Minsky and S. Papert, "Perceptrons: An Introduction to Computational Geometry", MIT Press, Cambridge, 1969.