Computação Quântica 2021-2022

Implementação de Algoritmos Quânticos numa CNN

Rafael Alves Vieira e José Pedro Araujo Azevedo

1. Introdução

Este relatório descreve os procedimentos, análises e conclusões obtidas na realização do projeto de computação quântica, que consiste na implementação de camadas quânticas em CNNs.

Foram elaboradas 3 versões do trabalho, sendo que para cada uma existe um notebook enviado em anexo, que pode ser corrido no google collab, e faz uso das GPUs da google para executar os treinos.

Como este trabalho cogitamos a melhora do exemplo dado no website do qiskit, no que toca a camada quantica da CNN. Para tal efeito, iniciou-se por entender o bom funcionamento da QNN e expandir o problema para as 10 classes do dataset, em vez das duas iniciais. De seguida, procorou-se expandir o circuito quântico usado na camada quântica para ter mais do que 1 qubit de forma a ter a hipótese de fazer algo de útil com as propriedades quânticas dos qubits. Outro tipo de circuito quantico foi testado (QAOA). Estes testes e passos feitos estão descritos nas secções seguintes.

Inicialmente pensávamos que apenas precisaríamos de fazer alterações no circuito quântico, e no máximo na hybrid layer da QNN. Mas o que acabámos por entender é que uma alteração no numero de qubits, ou no número de classes provoca alterações e adaptações que precisam de ser efetuadas em todo o código (forward pass, back propagation, em todos os aspetos e etapas da QNN). Por isso perdemos muito mais tempo do nosso trabalho a corrigir erros e bugs de pytorch/python do que a otimizar a parte quântica, mais direcionada ao objetivo do projeto e da cadeira.

Isto trouxe-nos muitos problemas e por causa disso não conseguimos implementar bem a parte de usar múltiplos qubits no circuito quântico e a parte de usar QAOA. Ambos os notebooks para estes dois casos correm sem erros mas obtemos accuracies muito baixas, e portanto pensamos que hajam erros de implementação.

2. Funcionamento da Quantum Neural Network

Partiu-se do exemplo disponibilizado no *Qiskit* (*Hybrid quantum-classical Neural Networks with PyTorch and Qiskit*), onde se tem uma *CNN* cuja última camada é híbrida (tem elementos de computação clássica e quântica) e que treina os parâmetros que calibram os qubits. Na camada híbrida existe um circuito quântico com 1 qubit (no caso de aplicação do Qiskit). Os parâmetros do circuito (rotação a ser usada nos gates) são os outputs da CNN que são treinados.

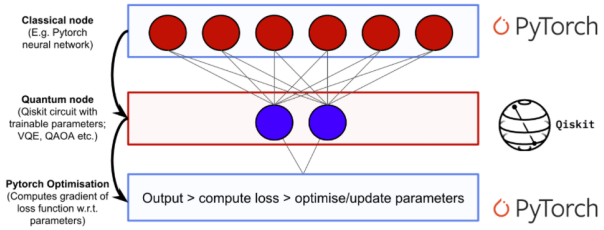


Figura 1: Ilustração da implementação de uma camada quântica na CNN.

Nas secções seguintes vai-se abordar o efeito que o número de *Qubits* e o número de classes possui neste tipo de problemas. Onde começamos por ter um circuito quantico com N *qubits*, nos quais fazemos "*load*"alguns dados sejam eles *classical* ou *quantum*, neste caso são dados *clássicos* visto pretendemos treinar *datasets*. Depois podemos aplicar uma série de *unitary gates*, estes podem ser *random rotations(y,x,z)* e uma série de *control gates* e por fim aplicamos um *measurement* num *qubit*, com isto calcula-se um gradiente para os nossos parâmetros. Eis um exemplo:

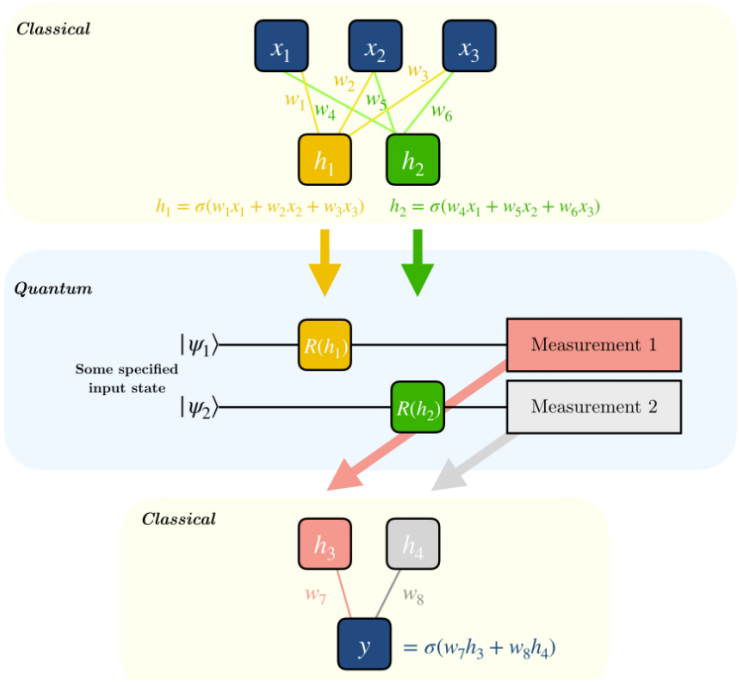


Figura 2: Ilustração de como fazemos uso da camada quântica.

Para calcular os gradientes nos circuitos quânticos usou-se a seguinte forma:

$$\nabla_\theta \boxed{\text{Quantum Circuit}(\theta)} = \boxed{\text{Quantum Circuit}(\theta+s)} - \boxed{\text{Quantum Circuit}(\theta-s)}$$

Figura 3: Ilustração do cálculos do circuito quântico.

onde, θ representa os parâmetros do circuito quântico e s um *shift* macroscópico. Então o gradiente é a diferença entre o circuito quântico avalido em $\theta + s$ e $\theta - s$. Desta forma pode-se diferenciar o circuito quântico como parte de uma rotina de retropropagação maior, vulgarmente esta regra é conhecida como *parameter shift rule*.

3. MULTI-CLASS

O primeiro passo que demos para melhorar a QNN inicial foi testá-la para múltiplas classes e mudar um pouco os hiperparâmetros bem como a arquitetura da rede que está antes da camada quantica.

Como queremos mais do que 2 classes começámos por alterar o dataset. Depois de ter o dataset com todas as classes, mudámos a arquitetura da rede. Tinhamos um kernel size de 5*5, mas como as nossas imagens são tão pequenas, achámos melhor um kernel size mais pequeno. Como supostamente o output da rede é um valor que é usado como parâmetro de entrada na hybrid layer, queremos ao invez de 1 valor, 10 valores (um para cada classe) e replicámos a hybrid layer em 10 layers com um ciclo for, (uma para cada output da rede convolucional).

Ou seja, não aumentamos o numero de qubits no circuito, apenas fizemos cópias deste circuito para as diferentes classes, e há parametros diferentes a serem treinados para cada classe, rotaçoes aplicadas no qubit diferentes para cada situação também.

No entanto, ao invez de termos uma accuracy de 100% como tinhamos no caso de duas classes apenas conseguimos 85%.

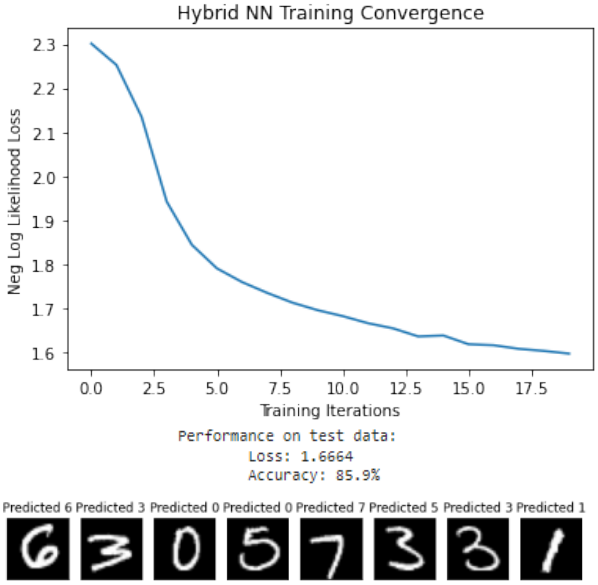


Figura 4: Resultados obtidos

4. MULTI-QUBIT

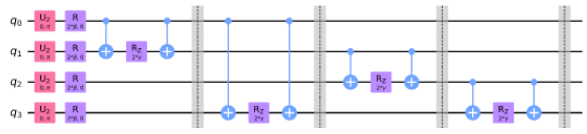
Para implementar mais do que 1 qubit nos circuitos quânticos, na nossa classe do circuito quantico temos o calculo da expetativa do valor de rotaçao para um certo parametro de entrada, vindo da CNN. Este calculo precisa de ser alterado para devolver em vez de 1 valor, N valores onde N é o numero de qubits que queremos usar. No nosso caso exprimentamos com 4 qubits. Como estamos a alterar o output do circuito quantico precisamos de adaptar o codigo onde fazemos uso deste output:

- **Backpropagation:** O calculo dos gradientes agora precisa de ser transformado num algoritmo escalável para o número de qubits que estamos a utilizar.
- **Arquitetura da rede:** Se na etapa anterior (multi-class) precisamos de expandir o output da CNN para o numero de classes, agora precisamos de expandir para numero de classes*numero de qubits visto que para cada classe vamos querer ter um circuito quantico com N qubits (o objetivo disto seria depois explorar possiveis ações com estados conjuntos e *entanglement*). Houve outra hipótse, que foi o que inicialmente tentámos fazer: ter um circuito quantico com N qubits e cada um destes ter parâmetros diferentes de rotação a serem treinados pelo output da rede (ou seja ter um numero de qubits igual ao numero de classes) tornou-se dificil de implementar e optámos por esta maneira de pensar mais linear, que evoluía de uma adaptação da etapa anterior (multi-class), onde testaríamos a vantagem de ter mais qubits no circuito quantico.
- **Forward pass:** Se a arquitetura da rede e a maneira como lidamos com os seus outputs muda, entao temos de alterar a forward pass de acordo com as alteraçoes feitas. Ou seja temos de chamar a hybrid layer N vezes onde N é numero de classes*numero de qubits. Cada classe tem 4 (sendo 4 o numero de qubits) parametros a serem treinados (os 4 angulos que queremos dar como parametro de entrada às rotaçoes dos 4 qubits) e temos N classes.

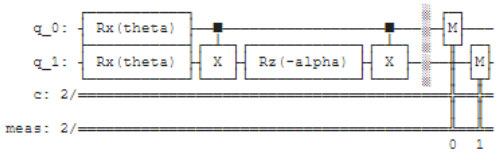
A implementação de múltiplos qubits foi uma falha visto que não se sabe qual o problema de implementação mas as accuracies nunca passaram dos 10% (o que é como estar a escolher aleatoriamente a classe de uma imagem, há 10% de chance de acertar). Como a implementação multi-qubit não é funcional, a secção seguinte que faz uso dos mesmos princípios aqui descritos também tem resultados pouco satisfatórios.

5. QAOA

Testou-se ainda a implementação de um QAOA (*Quantum Approximate Optimization Algorithm*), que é um tipo de algoritmo usado para resolver problemas de otimização (em matemática, problemas de otimização são aqueles onde se tem de encontrar a melhor solução para um problema, dado um conjunto de possíveis soluções). Este algoritmo aplica-se no problema das QNNs visto que o que se passa na última camada da rede (a que tem o circuito quântico) é um problema de otimização. Ou seja, na QNN o objetivo é encontrar qual o melhor conjunto de parâmetros a serem aplicados no circuito quântico, para obter uma classificação mais precisa. No exemplo do Qiskit para QAOA usa-se o seguinte circuito quântico:



Na implementação para a QNN apenas foram usados dois qubits, ou seja, o circuito fica com este aspeto:



Como são usados dois qubits e 10 classes, esta é como uma junção das últimas duas versões do projeto (multi-qubit e multi-class) com um cirquito quântico diferente. Portanto as alterações que se fez ao código nas últimas duas versões foram adaptadas para esta versão, mas com este circuito quântico. O programa corre sem erros, mas obtém-se uma accuracy muito baixa, portanto apesar do sentido que faria a aplicação de um QAOA nas QNNs, a complexidade da implementação aumenta e podem haver detalhes que tenham escapado.

6. Conclusões

Ao longo deste projeto, concluiu-se que escolher o número apropriado de *qubits* para o problema , é uma das partes fundamentais.
Retornar $2^n qubits$ da camada quântica não funciona conforme esperado, isto ocorre porque o circuito ”pensa”que deve haver uma correlação com medidas que diferem apenas um pouco. Concluiu-se que as *Hybrid NNs* podem ter aplicações potenciais para dispositivos NISQ(Noisy Intermediate-Scale Quantum Computing).

7. Visão geral mais recente do *State of the art*

Existem várias áreas onde este projeto pode ser usado além da classificação. As *Hybrid Quantum Neural-Networks* já estão a ser implementadas em diversas aplicações com uso intensivo de dados e também para acelarar a computação em *fully distributed platforms*, como *Blockchain*, áreas da saúde (predições de cancro), lazer(jogos 3D, realiade virtual), aplicações industriais(3D modeling/reconstruction, veículos autonomos) entre outras.[1]

QNN já tem aplicações em diversas áreas do mundo real. Como é descrito em [1] Kouda et al, estudou a performance de uma QNN para compressão de imagens de grande tamanho, por poucas palavras , este engenheiro provou que as QNN tem alta capacidade de processamento na compressão de imagens, comparando com o processamento clássico.

8. Perspetivas futuras

Até ao momento, ideias quânticas foram propostas para a realização efetiva de computação clássica - em vez de *neural* computation. O conceito de computação quântica pode ser rastreada até ao trabalho pioneiro de Richard Feynman, que examinou o papel que os efeitos quânticos desempenhariam no desenvolvimento de hardware futuro.Á medida que as velocidades de hardware continuam a aumentar, o hardware escala corrrespondentemente com a diminuição e em algum ponto num futuro próximo. Como é descrito em [2], Feynman percebeu que *gates* e *wires* podem consistir em apenas alguns átomos, e os efeitos quânticos desempenharão um papel importante no hardware, Feynman ainda percebeu que esses dispositivos quânticos podem ter vantagens significativas sobre os computadores clássicos.

Num futuro próximo, teremos *gate model quantum computers* com um número suficiente de *qubits* e *high gate fidelity* suficiente para executar circuitos com profundidade suficiente para executar tarefas que não podem ser simuladas em computadores clássicos.[3]

A potência da computação quântica em redes neuronais certamente levará a inteligência artificial(AI) a atingir um novo patamar.[1]

Referências

[1] S. K. Jeswal e S. Chakraverty. “Recent Developments and Applications in Quantum Neural Network: A Review”. Em: (2018). DOI: https://www.researchgate.net/publication/324936022_Recent_Developments_and_Applications_in_Quantum_Neural_Network_A_Review.

[2] Alexandr A. Ezhov e Dan Ventura. “Future Directions for Intelligent Systems and Information Science”. Em: (2000). DOI: https://axon.cs.byu.edu/Dan/papers/ezhov2000fdisis.pdf.

[3] Edward Farhi e Hartmut Neven. “Classification with Quantum Neural Networks on Near Term Processors”. Em: (2018). DOI: https://arxiv.org/pdf/1802.06002.pdf.