Lucas Pontes de Albuquerque

Algoritmos Quânticos para Classificação de Imagens

Universidade Federal de Pernambuco

Recife
2021

Lucas Pontes de Albuquerque

Algoritmos Quânticos para Classificação de Imagens

Trabalho apresentado ao Programa de Bacharelado em Engenharia da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Bacharel em Engenharia da Computação.

Área de Concentração: *Aprendizagem de Máquina Quântica; Visão Computacional*
Orientador: *Tsang Ing Ren*
Co-Orientador: *Adenilton Jose da Silva*

Recife
2021

ABSTRACT

Machine Learning methods have been widely used in pattern recognition applications involving a wide variety of data. Convolutional Neural Networks (CNN) have been broadly explored in literature for unstructured data, reaching the state of art of the most used learning techniques. Although convolutional layers can learn features using few parameters, this amount of parameters can easily reach millions or billions due to the high depth of the architectures. Thus, a large amount of energy is required to use these techniques making them less replicable and scalable. One of the more prominent computational paradigms with more efficient results in memory and time is Quantum Computation. The classic example is Shor's algorithm, which finds the solution for prime numbers factoring and discrete logarithm in polynomial time, while the more efficient classic algorithm has exponential complexity. Because quantum computation has high parallelism capacity and the exponential increase of the vector space with the number of qubits, the development of machine learning methods using quantum mechanics has been explored. Since quantum computation can offer new learning approaches, hybrid methods are explored to build new architectures. Thus, the purpose of this work is to develop hybrid learning methods for image classification, comparing the perfomance of quantum circuito model with classical approaches.

Keywords: Quantum Machine Learning. Quantum Computation. Computer Vision. Deep Learning.

RESUMO

Os métodos de Aprendizagem de Máquina tem sido vastamente utilizados em aplicações de reconhecimento de padrões envolvendo diversos tipos de dados. Das técnicas de aprendizagem de máquina mais utilizadas, as Redes Neurais Convolucionais (CNN) tem sido amplamente exploradas na literatura para dados não estruturados, alcançando resultados que hoje são o estado da arte. Apesar das camadas convolucionais ter a capacidade de aprender características úteis utilizando poucos parâmetros, devido a alta profundidade das arquiteturas, essa quantidade de parâmetros pode chegar facilmente a milhões ou bilhões. Assim, uma grande quantidade de energia é necessária para utilizar essas técnicas as tornando menos replicáveis e escaláveis. Um dos métodos computacionais que tem ganhado destaque por obterem soluções para certos problemas em tempo e memória mais eficientes é a Computação Quântica. O exemplo clássico é da fatoração de números primos e logaritmo discreto no qual o algoritmo de Shor consegue achar a solução em tempo polinomial, enquanto o algoritmo clássico mais eficiente tem complexidade exponencial. Pelo fato da Computação Quântica ser capaz de uma alta capacidade de paralelismo e sua dimensionalidade vetorial aumentar exponencialmente pela quantidade de *qubits* (unidade de memoria utilizada) tem sido explorado o desenvolvimento de métodos de aprendizado de máquina usando mecânica quântica. Como a computação quântica pode oferecer novas abordagens de aprendizagem, métodos híbridos utilizando a computação clássica e quântica tem sido explorados na criação de novas arquiteturas. Assim, a proposta desse trabalho é desenvolver métodos de aprendizagem híbridos para classificação de imagens, comparando com abordagens clássicas.

Palavras-chave: Aprendizagem de Máquina Quântica. Computação Quântica. Visão Computacional. Aprendizagem Profunda.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ACRÔNIMOS

CIFAR10	*Canadian Institute For Advanced Research*
CNN	*Convolutional Neural Networks*
DAG	*Directed Acyclic Graph*
MLP	*Multilayer Perceptron*
PCA	*Principal Component Analysis*
QCNN	*Quantum Convolutional Neural Networks*
QGAN	*Quantum Generative Adversarial Networks*
QVA	*Quantum Variational Autoencoder*

LISTA DE SÍMBOLOS

$\mathbf{C}$	Conjunto dos números complexos
$\mathbf{N}$	Conjunto dos números naturais
$\mathbf{R}$	Conjunto dos números reais
$\otimes$	Produto de Kronecker
σ	Função de ativação *softmax*
e	Número/constante de Euler
i	Número imaginário $\sqrt{-1}$
$\|\cdot\rangle$	Estado quântico

LISTA DE ALGORITMOS

SUMÁRIO

1

INTRODUÇÃO

1.1 Motivação

O desenvolvimento de métodos de classificação de imagens têm sido vastamente explorado pela comunidade acadêmica e pela indústria em diversos tipos de aplicações. Apesar da existência de métodos sem aprendizagem para construir soluções para esses problemas, foi com o desenvolvimento das Redes Neurais Convolucionais (CNN) (LeCun *et al.*, 1999) que começaram a surgir soluções com maior acurácia e capacidade de generalização. Em (Krizhevsky *et al.*, 2012) foi apresentada uma arquitetura de CNN, treinada em GPUs (Graphical Processing Units), que supera o desempenho humano no banco de dados ImageNet (Deng *et al.*, 2009). Devido a isso, posteriormente foram construídas arquiteturas de CNN para as mais diversas aplicações envolvendo visão computacional, como segmentação (Ronneberger *et al.*, 2015), detecção de objetos (Redmon *et al.*, 2015), restauração de imagens (Ulyanov *et al.*, 2017) e transferência de estilo (Johnson *et al.*, 2016). Apesar de camadas convolucionais serem reconhecidas pelo baixo número de parâmetros e diversidade na extração de características dos dados, devido a alta profundidade das arquiteturas o número de parâmetros podem chegar facilmente a milhões ou bilhões. Isso pode ser observado em métodos como VGG (Simonyan & Zisserman, 2015)(135 milhões de parâmetros, ResNet (He *et al.*, 2015)(de 11 a 58 milhões de parâmetros, UNet (Ronneberger *et al.*, 2015)(17 milhões) e, mais recentemente, *Vision Transformers* (Dosovitskiy *et al.*, 2020)(82 milhões). Como essas arquiteturas são muito robustas e exigem muita memória e processamento, isso faz com que também seja necessária uma grande quantidade de energia além de tornar esses métodos menos replicáveis e escaláveis.

Algoritmos quânticos têm ganhado destaque por obterem soluções para certos problemas em tempo e memória mais eficientes que os algoritmos clássicos. O caso mais conhecido é o algoritmo quântico da fatoração de números primos e logaritmo discreto (Shor, 1997) no qual consegue achar a solução em tempo polinomial, enquanto o algoritmo clássico conhecido mais eficiente atualmente tem complexidade exponencial. Pelo fato da computação quântica ser capaz de um alto paralelismo e sua dimensionalidade vetorial aumentar exponencialmente pela quantidade de *qubits*(unidade de memória utilizada) uma das áreas com potencial utilização desse paradigma é a aprendizagem de máquina. Recentemente tem sido desenvolvido diversos

métodos de aprendizagem quânticos que buscam um análogo com os reais, como *Quantum Neural Networks* (Abbas *et al.*, 2020), *Quantum Convolutional Networks* (Cong *et al.*, 2019; Henderson *et al.*, 2019), *Quantum Variational Autoencoders* (Khoshaman *et al.*, 2018) e *Quantum Graph Neural Networks* (Verdon *et al.*, 2019). Como a computação quântica pode oferecer novas abordagens de aprendizagem, métodos híbridos utilizando a computação clássica e quântica têm sido explorados na criação de novas arquiteturas. Em (Pérez-Salinas *et al.*, 2020) é apresentado um método análogo a *Multileyer Perceptrons* (MLPs), onde é demonstrada sua capacidade como classificador universal, e em (Mari *et al.*, 2020) é apresentada uma arquitetura utilizando *Transfer Learning* entre modelos clássicos e quânticos para um subconjunto dos dados da ImageNet (Deng *et al.*, 2009).

1.2 Objetivos

O objetivo **geral** deste trabalho é explorar o desempenho de diferentes configurações de arquiteturas de aprendizagem híbridas para classificação de imagens, utilizando extração de características através de arquitetura de CNN pré-treinadas e métodos quânticos para realizar a classificação. O foco dos experimentos realizados é demonstrar a capacidade dos métodos híbridos de resolver o problema de classificação de imagem em comparação com os outros métodos clássicos.

São objetivos **específicos** deste projeto:

- Extrair de arquiteturas CNN pré-treinadas características das imagens de entrada;
- Projetar circuitos quânticos variacionais para serem otimizados utilizando *Parameter Shift-Rule* (Crooks, 2019);
- Treinar os circuitos variacionais em diferentes bancos de dados;
- Projetar arquitetura híbrida utilizando circuitos variacionais e as arquiteturas CNN pré-treinadas escolhidas;
- Treinar diversas configurações das arquiteturas híbridas propostas em diferentes conjuntos de dados e fazer uma analise comparativa dos resultados;

2

CONCEITOS BÁSICOS

2.1 Computação Quântica

Para desenvolver a compreensão de como construir modelos híbridos, primeiramente devemos introduzir os conceitos básicos sobre computação quântica e seus métodos de aprendizagem. Na primeira seção vemos como é definida a unidade de memória quântica, o *qubit*, que servirá como base para compreender como são carregados os dados de entrada em estados quânticos. Nas seções seguintes vemos como o espaço vetorial é descrito no computador quântico, os operadores utilizados para realizar a construção de algoritmos quânticos, como a informação clássica é recuperada através da medição e, por fim, os conceitos básicos sobre aprendizagem de máquina quântica.

2.1.1 Estados quânticos e o *qubit*

Na computação clássica, realizada nos *chips* de processamento mais comuns no mercado, são definidas unidades de memória mínimas nas quais são realizadas operações nessas unidades. Nesse caso os computadores atuais utilizam o *bit* como unidade de memória para codificar estruturas de dados e realizar operações matemáticas para computar os seus devidos algoritmos. Para formular os conceitos da computação quântica, fazendo definições análogas, definimos a unidade de memória quântica como *qubit*. Essa unidade pode ser representada como um vetor $|\psi\rangle \in \mathbf{C}^2$, sendo $\mathbf{C}$ o conjunto dos números complexos e $|\cdot\rangle$ a notação que usamos para definir um vetor complexo de dimensão finita. Por se tratar de um espaço vetorial, definindo suas bases podemos representar toda a informação presente no *qubit* através de uma combinação linear dessas bases. Como exemplo, podemos definir o conjunto $\{|0\rangle, |1\rangle\}$ como base vetorial, sendo os vetores

$$|0\rangle \equiv \begin{bmatrix} 1 \\ 0 \end{bmatrix}, |1\rangle \equiv \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \tag{2.1}$$

e assim, definir o estado quântico $|\psi\rangle$ como uma soma dessas bases ponderadas por $a, b \in \mathbf{C}$,

$$|\psi\rangle = a|0\rangle + b|1\rangle = \begin{bmatrix} a \\ b \end{bmatrix}, \tag{2.2}$$

onde $|a|^2 + |b|^2 = 1$. Essa última condição faz com que possamos representar o estado quântico codificado em um *qubit* de forma polar, ou seja, como um ponto de uma esfera unitária com os ângulos (θ, φ) como é mostrado na Figura 1. Essa esfera é chamada de *esfera de Bloch* e o estado $|\psi\rangle$ é representado por ela através dos ângulos θ e ϕ, como mostrado na equação

$$|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\varphi}\text{sen}\frac{\theta}{2}|1\rangle, \tag{2.3}$$

sendo $i = \sqrt{-1}$ e e a constante de Euler[1].

Com a finalidade de aumentar a quantidade de informação codificada no computador quântico, precisamos definir como são representados os estados quando existem múltiplos *qubits* disponíveis. Para isso, utilizamos o *produto de Kronecker*, representado por $\otimes$, entre os estados presentes em cada *qubit*. Por exemplo, se temos um sistema com dois *qubits* com estados $|\psi_0\rangle = a_0|0\rangle + b_0|1\rangle$ e $|\psi_1\rangle = a_1|0\rangle + b_1|1\rangle$, o estado final é dado por

$$|\Psi\rangle = |\psi_0\rangle \otimes |\psi_1\rangle = \begin{bmatrix} a_0 a_1 \\ a_0 b_1 \\ b_0 a_1 \\ b_0 b_1 \end{bmatrix}. \tag{2.4}$$

Analisando a dimensionalidade do estado quântico de acordo com o número de *qubits* percebemos que dado um computador quântico com *n qubits*, temos um estado $|\Psi\rangle \in \mathbf{C}^{2^n}$, ou seja, o número de dimensões do espaço vetorial aumenta exponencialmente com a quantidade de *qubits*.

Para simplificarmos a notação para computação com múltiplos *qubits*, dizemos que o estado resultante do produto de Kronecker entre os $|\psi\rangle$ de cada *qubit* pode ser descrito por

$$|\Psi\rangle \equiv |\psi_0\rangle \otimes |\psi_1\rangle \otimes \cdots \otimes |\psi_{n-1}\rangle \equiv |\psi_0\rangle|\psi_1\rangle \cdots |\psi_{n-1}\rangle \equiv |\psi_0 \psi_1 \cdots \psi_{n-1}\rangle, \tag{2.5}$$

sendo n o número de *qubits*. Assim, podemos definir estados com *n qubits* utilizando um somatório

$$|\Psi\rangle = \sum_{j_0=0}^{1} \sum_{j_1=0}^{1} \cdots \sum_{j_{n-1}=0}^{1} a_{j_0 j_1 \cdots j_{n-1}} |j_0 j_1 \cdots j_{n-1}\rangle = \sum_{i=0}^{N-1} a_i |i\rangle, \tag{2.6}$$

onde $\sum_{i=0}^{N-1} |a_i|^2 = 1$, $N = 2^n$ e $i \in \mathbf{N}$ a representação decimal da cadeia binária $j_0 j_1 \cdots j_n \in \{0,1\}^n$.

[1] Essa equação é obtida através da representação exponencial de $a, b \in \mathbf{C}$, logo também existe uma fase global $e^{i\lambda}$ que é omitida por não possuir efeitos observáveis na medição.

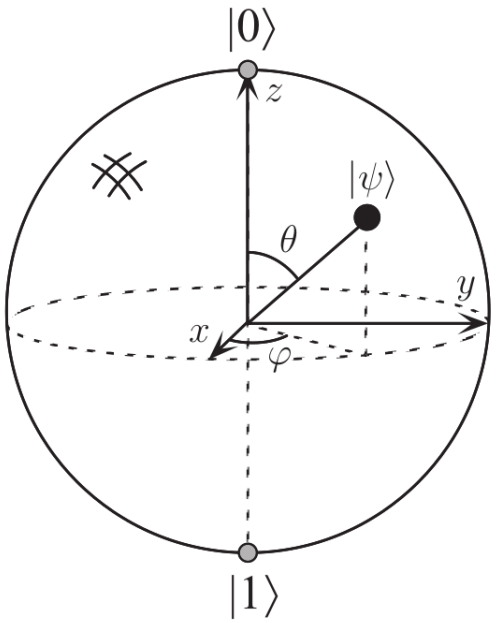

Figura 1: Representação gráfica da esfera de Bloch

2.1.2 Operadores

Sabendo dessas definições temos o conhecimento de como a informação pode ser armazenada no computador quântico. Assim como nos computadores convencionais, precisamos definir as operações realizadas sobre os *qubits* para desenvolver algoritmos. Se num computador clássico definimos operações *booleanas* sobre os *bits* (e.g. operação unária de negação $\mathrm{NOT}(1) = 0$ e operações binárias como $\mathrm{AND}(1,0) = 0$), em computadores quânticos construímos um cenário análogo. Como o estado de um *qubit* é representado por um vetor $|\psi\rangle = a|0\rangle + b|1\rangle$ onde $a, b \in \mathbf{C}$ satisfazem $|a|^2 + |b|^2 = 1$, as operações devem preservar essa norma dos estados e, por tanto, devem ser matrizes unitárias 2×2. As operações mais básicas são as matrizes de Pauli X, Y e Z, a porta Hadamard (denotada pela matriz H) e a matriz identidade I, sendo elas

$$\mathrm{X} \equiv \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}; \mathrm{Y} \equiv \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}; \mathrm{Z} \equiv \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}; \mathrm{H} \equiv \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}; I \equiv \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}. \tag{2.7}$$

As matrizes de Pauli quando usadas como potência de uma exponencial complexa dão origem a outros operadores, conhecidos como *operadores de rotação* dos eixos $\hat{x}$, $\hat{y}$ e $\hat{z}$ da esfera de Bloch e são definidos pelas equações:

$$R_x(\theta) \equiv e^{-i\theta \mathrm{X}/2} = \cos\frac{\theta}{2} I - \mathrm{sen}\frac{\theta}{2} \mathrm{X} = \begin{bmatrix} \cos\frac{\theta}{2} & -i\mathrm{sen}\frac{\theta}{2} \\ -i\mathrm{sen}\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix} \tag{2.8}$$

$$R_y(\theta) \equiv e^{-i\theta \mathrm{Y}/2} = \cos\frac{\theta}{2} I - \mathrm{sen}\frac{\theta}{2}\mathrm{Y} = \begin{bmatrix} \cos\frac{\theta}{2} & -\mathrm{sen}\frac{\theta}{2} \\ \mathrm{sen}\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix} \tag{2.9}$$

$$R_z(\theta) \equiv e^{-i\theta \mathrm{Z}/2} = \cos\frac{\theta}{2} I - \mathrm{sen}\frac{\theta}{2}\mathrm{Z} = \begin{bmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{bmatrix}. \tag{2.10}$$

2.1.3 Operadores Controlados e de Múltiplos *qubits*

No caso de operadores que atuam em dois *qubits*, definimos o operador de negação controlado CNOT, no qual o estado do *qubit* de controle inverte o *bit* da base canônica (i.e. $\{|0\rangle, |1\rangle\}$) do *qubit* alvo quando o valor do estado de controle é $|1\rangle$. Por exemplo, digamos que o estado de um sistema quântico com dois *qubits* é $|\Psi\rangle = \sum_{i=0}^{3} a_i|i\rangle$, onde $a_0 = a_2 = 1/\sqrt{2}$ e o restante iguais a zero, aplicando o operador CNOT temos:

$$\mathrm{CNOT}|\Psi\rangle = \mathrm{CNOT}(a_0|00\rangle + a_2|10\rangle) = a_0|00\rangle + a_2|11\rangle, \tag{2.11}$$

nesse caso o primeiro *qubit* é o de controle. Assim, definimos CNOT como a seguinte matriz:

$$\mathrm{CNOT} \equiv \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{2.12}$$

Podemos também definir operadores em múltiplos *qubits* fazendo o produto de Kronecker entre os operadores que estão sendo aplicados em cada *qubit*. Assim, dizemos que um operador unitário $U = \bigotimes_{i=0}^{n-1} U_i$ aplicado a um estado $|\Psi\rangle = \bigotimes_{i=0}^{n-1} |\psi_i\rangle$ com n *qubits* pode ser definido como:

$$U|\Psi\rangle = U_0 \otimes U_1 \otimes \cdots \otimes U_{n-1}|\Psi\rangle = \bigotimes_{i=0}^{n-1} U_i|\psi_i\rangle. \tag{2.13}$$

A dimensionalidade de U resultante do produto de Kronecker entre operadores U_i de um *qubit* será $2^n \times 2^n$.

2.1.4 Construção de um circuito quântico

Precisamos agora entender como são desenvolvidos algoritmos quânticos através de uma visualização gráfica de como ficam dispostos os operadores aplicados nos *qubits* a cada passo de computação. Isso pode ser feito descrevendo um circuito quântico como um grafo acíclico direcionado (DAG). Assim, para representar um operador aplicado a 1 *qubit* usamos a notação de caixas(operadores) em cima de linhas(*qubit*) como mostrado na Figura 2. No caso da aplicação de vários operadores aplicados em 1 *qubit* colocamos as respectivas caixas dos operadores em

série (Figura 3).

Figura 2: Representação gráfica do circuito equivalente a equação $|\psi^{out}\rangle = \mathrm{U}|\psi^{in}\rangle$, sendo U um operador unitário, $|\psi^{in}\rangle$ o estado de entrada e $|\psi^{out}\rangle$ o estado de saída.

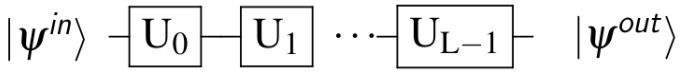

Figura 3: Representação gráfica do circuito equivalente a equação $|\psi^{out}\rangle = \mathrm{U}_{\mathrm{L}-1}\cdots\mathrm{U}_1\mathrm{U}_2|\psi^{in}\rangle$, sendo U_l operadores unitários, $|\psi^{in}\rangle$ o estado de entrada e $|\psi^{out}\rangle$ o estado de saída.

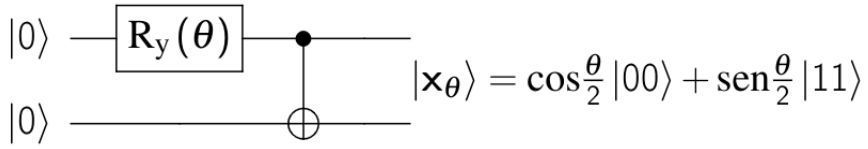

Figura 4: Representação gráfica do circuito equivalente a equação $|\Psi\rangle = \mathrm{CNOT}(\mathrm{R_y}(\theta)\otimes I)|00\rangle$, sendo $\mathrm{R_y}(\theta)$ operador de rotação em θ radianos no eixo $\hat{y}$ da esfera de Bloch, CNOT operador de negação controlado e I a identidade, $|00\rangle$ o estado de entrada e $|\Psi\rangle$ o estado de saída.

É mostrado na Figura 4 um exemplo de combinação de operadores em mais de 1 *qubit* em um circuito quântico. Podemos considerar esse circuito como sendo os passos para carregar em um estado quântico $|\mathbf{x}_\theta\rangle$ o vetor $\mathbf{x}_\theta = (\cos\frac{\theta}{2}, 0, 0, \mathrm{sen}\frac{\theta}{2})$, armazenado num computador clássico. Considerando as bases canônicas em circuito de 2 *qubits* (i.e. $|00\rangle \equiv |0\rangle$, $|01\rangle \equiv |1\rangle$, $|10\rangle \equiv |2\rangle$ e $|11\rangle \equiv |3\rangle$) como os índices de cada dimensão do vetor e o conteúdo associado ao índice a constante que multiplica o estado da base canônica. Chamamos esse carregamento de *codificação em amplitude*, no qual dado um vetor clássico $\mathbf{x} = (x_0, x_1, \ldots, x_{2^n-1})$, sendo $x_i \in \mathbf{C}$ e $i = 0, 1, \ldots, 2^n - 1$, onde $\sum_{i=0}^{2^n-1} |x_i|^2 = 1$, codificamos em um estado quântico $|\mathbf{x}\rangle = \sum_{i=0}^{2^n-1} x_i |i\rangle$. Assim, podemos codificar $\mathbf{x}_\theta$ em um estado $|\mathbf{x}_\theta\rangle$, considerando $|00\rangle$ o estado inicial(ou de entrada), primeiramente aplicando um operador $\mathrm{R}_y(\theta)$ no primeiro *qubit*, como mostrado na equação

$$|\mathbf{x}_\theta\rangle = \mathrm{CNOT}(\mathrm{R}_y(\theta)\otimes I)|00\rangle = \mathrm{CNOT}(\cos\frac{\theta}{2}|00\rangle + \mathrm{sen}\frac{\theta}{2}|10\rangle),$$

e em seguida um CNOT com o primeiro *qubit* controlando o segundo ficando com o estado de saída do circuito

$$|\mathbf{x}_\theta\rangle = \cos\frac{\theta}{2}|00\rangle + \mathrm{sen}\frac{\theta}{2}|11\rangle.$$

Sabendo como funciona o operador CNOT podemos definir operadores de rotação controlados. Na Figura 5 é mostrado como podem ser construídos os operadores R_y de forma controlada. Para controlarmos o operador utilizando o estado $|0\rangle$ do *qubit* de controle, utilizamos o circuito mostrado na Figura 6. Para construir um exemplo de como fica o estado do circuito

após a aplicação do operador R_y controlado são mostrados dois circuitos na Figura 7, onde no circuito superior R_y é controlado por um estado $|1\rangle$ e no inferior controlado por $|0\rangle$, sendo $|\psi\rangle$ o estado de saída do circuito.

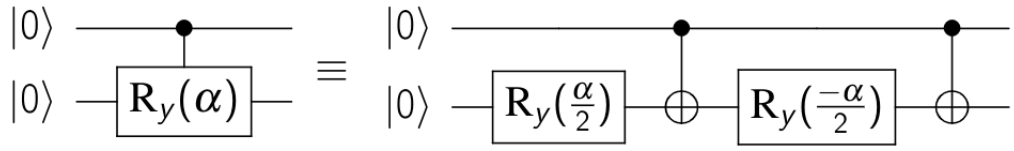

Figura 5: Representação gráfica dos circuitos equivalentes a operadores R_y com ângulo α controlados pelo estado $|1\rangle$.

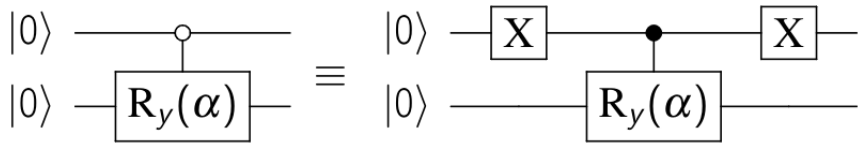

Figura 6: Representação gráfica dos circuitos equivalentes a operadores R_y com ângulo α controlados pelo estado $|0\rangle$.

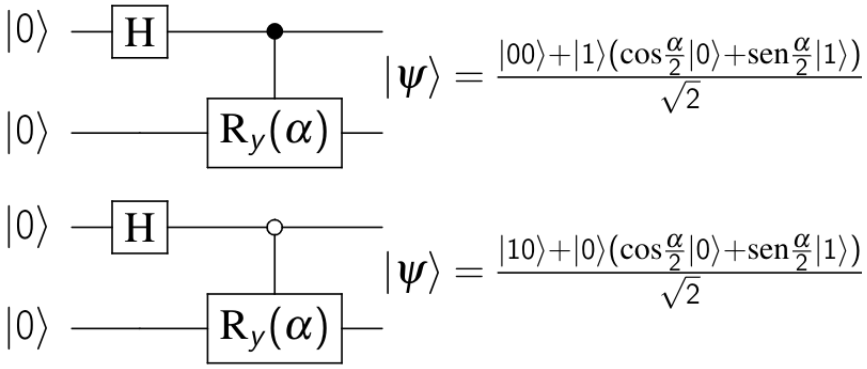

Figura 7: Representação gráfica de dois exemplos de circuitos utilizando operadores R_y com ângulo α controlados. O circuito superior tem R_y controlado pelo estado $|1\rangle$ e o inferior pelo $|0\rangle$, sendo $|\psi\rangle$ o estado de saída do circuito.

2.1.5 Medição

Depois de construir o circuito para determinado algoritmo precisamos extrair a informação clássica existente no estado final do circuito. Para isso, temos que realizar a operação de medição nos *qubits*, onde a informação clássica obtida é relacionada a escolha de bases vetoriais. Para exemplificar, dado um estado $|\psi\rangle = a|0\rangle + b|1\rangle$, dizemos que é um estado em sobreposição em relação a base $\{|0\rangle, |1\rangle\}$. Associando os *bits* clássicos aos estados da base temos 0 está associado a $|0\rangle$ e 1 a $|1\rangle$. Assim, fazendo uma medição sobre $|\psi\rangle$ nas bases $|0\rangle$ e $|1\rangle$ temos que $p(0) = |a|^2$ e que $p(1) = 1 - p(0) = |b|^2$, sendo $p : \{0,1\} \to \mathbf{R}$ a função de probabilidade. Ou seja, a norma quadrática dos complexos que multiplicam as bases que formam $|\psi\rangle$ são as amplitudes de probabilidade para obter cada uma das bases após a medição.

As medições podem ser definidas como um operador de medição $M \in \mathbf{C}^{2^n \times 2^n}$ chamado de observador, sendo M igual ao seu conjugado transposto $M^\dagger$. A informação $\langle M \rangle \in \mathbf{R}$ obtida da medição em um estado $|\psi\rangle$ é definida pela equação

$$\langle M \rangle = \langle \psi | M | \psi \rangle , \tag{2.14}$$

sendo $\langle \psi |$ o transposto conjugado de $|\psi\rangle$. Definindo operador de Pauli Z como observável, considerando $|\psi\rangle$ o estado definido no inicio desta seção, temos que o resultado obtido após a medição é dado por

$$\langle Z \rangle = \langle \psi | Z | \psi \rangle = p(0) - p(1) = |a|^2 - |b|^2,$$

com isso temos que $\langle Z \rangle \in [-1, 1]$. A função p é aproximada através da média dos resultados obtidos num número finito de medições, no caso do operador Z como observador temos que o resultado de uma medição é 1 se a base observada for $|0\rangle$ e -1 se $|1\rangle$.

Se temos $|\psi\rangle = \sum_{i=0}^{2^n-1} a_i |i\rangle$ e aplicarmos uma medição com observador M_j no *qubit* $j = 0, 1, \ldots, n-1$, temos que o resultado medição desse *qubit* é dado pela expressão

$$\langle M_j \rangle = \langle \psi | I^{\otimes j} \otimes M_j \otimes I^{\otimes n-j-1} | \psi \rangle \tag{2.15}$$

onde $I^{\otimes n} \equiv \bigotimes_0^{n-1} I$. Na Figura 8 é apresentada a representação simbólica do simbolo da medição em um circuito após a aplicação de um operador unitário $\mathrm{U} \in \mathbf{C}^{8\times 8}$ no estado inicial $|000\rangle$.

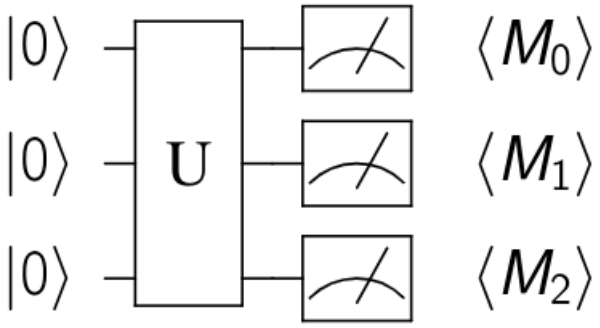

Figura 8: Exemplo da utilização do símbolo de medição num circuito de 3 *qubits* após a aplicação de um operador $\mathrm{U} \in \mathbf{C}^{8\times 8}$ unitário no estado inicial $|000\rangle$.

2.2 Aprendizagem de Máquina Quântica

Como observado anteriormente, os algoritmos quânticos são definidos através da construção de circuitos e operações matriciais. Apesar de suas restrições, como as matrizes dos operadores serem unitárias, é possível formular métodos de aprendizagem utilizando os conceitos da computação quântica. Isso pode ser feito através da definição de operadores em três etapas: codificação dos dados clássicos em estados quânticos, construção de operadores parametrizados e medição. Na codificação, temos um operador $U(\mathbf{x})\,|0\rangle^{\otimes n} = |\mathbf{x}\rangle$, sendo $\mathbf{x} \in \mathbf{R}^N$ uma amostra

do banco de dados e N a sua quantidade de características. Após a codificação, aplica-se um operador parametrizado $U(\Theta)|\mathbf{x}\rangle = |\psi_\Theta\rangle$, onde Θ é o conjunto de parâmetros a serem aprendidos ou otimizados. E assim, é feita a medição $\langle M_j \rangle$ em cada *qubit* $j = 0, 1, \ldots, n-1$, utilizando um observador M_j. Todo processo das três etapas está representado graficamente na Figura 9.

2.2.1 Codificação dos Dados em Estados Quânticos

Dos métodos mais conhecidos de codificação podemos citar a codificação por amplitude (Möttönen *et al.*, 2005), por base canônica (Farhi & Neven, 2018) e por produto tensorial (Guerreschi & Smelyanskiy, 2017). No caso da codificação por base canônica, o operador pode ser descrito por $U(\mathbf{x})|0\rangle^{\otimes n} = |i\rangle$, sendo $\mathbf{x} \in \{0,1\}^n$, $i = 0, 1, \ldots, 2^n - 1$ e n o número de *qubits*. Para a codificação por produto tensorial temos o operador $U(\mathbf{x})|0\rangle^{\otimes n} = \bigotimes_{i=0}^{n-1} \mathrm{U}(x_i)|0\rangle$, sendo $\mathbf{x} = (x_0, x_1, \ldots, x_{n-1})$ e $\mathrm{U}(x_i)$ um operador de rotação qualquer com ângulo x_i. A codificação por amplitude foi exemplificada na Seção 2.1.4, e pode ser representada pelo operador $U(\mathbf{x})|0\rangle^{\otimes n} = \sum_{i=0}^{2^n-1} x_i |0\rangle$.

Como pode ser observado, na codificação por amplitude existe uma compressão exponencial, já que considerando um vetor de entrada $\mathbf{x} \in \mathbf{R}^{2^n}$ podemos usar um circuito com n *qubits* para codificá-lo, diminuindo o custo em memória do computador quântico. Nas outras codificações é observado que o espaço vetorial do circuito dado um vetor de entrada $\mathbf{x} \in \mathbf{R}^N$ será de tamanho 2^N, já que serão necessários N *qubits* para codificação.

2.2.2 Circuito Quântico Parametrizado

Um circuito quântico parametrizado, também chamado de circuito quântico variacional, pode ser descrito como um conjunto de operadores de rotação com parâmetros θ aplicados em cada *qubit* junto com operadores CNOT. Diversas configurações de circuitos variacionais são possíveis e determinar que tipo de configuração é a mais apropriada para modelar uma função objetivo ainda é um problema em aberto. Em (Sim *et al.*, 2019) é calculada uma métrica de expressividade em diferentes configurações de circuitos parametrizados e assim pode servir como base para definir o operador $U(\Theta)$.

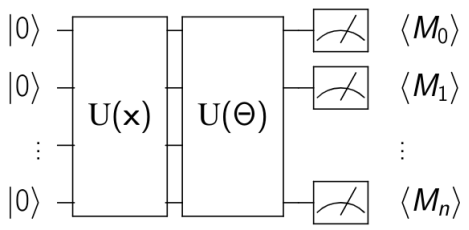

Figura 9: Representação gráfica das etapas para a construção de um circuito parametrizado. Na codificação, temos um operador $U(\mathbf{x})|0\rangle^{\otimes n} = |\mathbf{x}\rangle$, sendo $\mathbf{x} \in \mathbf{R}^{2^n}$ uma amostra do banco de dados. No modelo quântico aplica-se um operador parametrizado $U(\Theta)|\mathbf{x}\rangle = |\psi_\Theta\rangle$, onde Θ é o conjunto de parâmetros a serem aprendidos. Na medição é aplicado um observador M_j em cada *qubit* $j = 0, 1, \ldots, n-1$ obtendo $\langle M_j \rangle$.

2.2.3 Otimização dos Parâmetros

Para o circuito quântico variacional modelar a função objetivo, precisamos de um de um método para extrair os gradientes da função de erro em relação a determinado parâmetro. Podemos utilizar a demonstração apresentada em (Crooks, 2019) para fazer um cálculo da derivada parcial do parâmetro dada uma função objetivo f. O cálculo dessa derivada pode ser apresentada de forma simplificada através da equação

$$\frac{\partial f}{\partial \theta} = f(\theta + s) - f(\theta - s) \tag{2.16}$$

sendo θ o parâmetro de rotação de um operador aplicado em um *qubit* e $s \in \mathbf{R}$ uma constante. Essa operação também pode ser representada de acordo com a Figura 10, onde o gradiente é calculo de acordo com a medição no circuito com o parâmetro θ deslocado para "esquerda"(i.e $\theta + s$) e depois medido com θ deslocado para "direita"(i.e. $\theta - s$). Pelo fato do gradiente ser calculado através desses deslocamentos, esse calculo é chamado de *Parameter Shift Rule*.

Após o calculo do gradiente os parâmetros podem ser atualizados através de qualquer otimizador clássico. Se por exemplo for utilizado SGD (*Stochastic Gradient Descendent*) para atualizar o parâmetro θ_t no passo de iteração t temos a equação

$$\theta_t = \theta_{t-1} - \alpha \frac{\partial f}{\partial \theta} \tag{2.17}$$

onde α é a taxa de aprendizagem(ou *learning rate*).

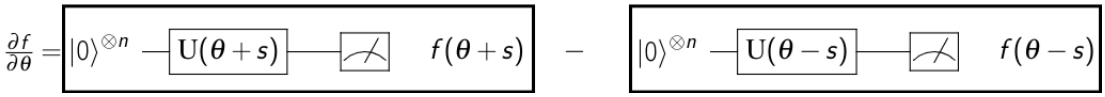

Figura 10: Representação da operação de *Parameter Shift Rule* para o calulo de gradientes dos parâmetros de um circuito variacional

3

TRABALHOS RELACIONADOS

3.1 Extração de Características

Certos tipos de dados possuem distribuições estatísticas nas quais tornam difícil a modelagem de uma função utilizando métodos de aprendizagem, sendo necessário o uso de redução de dimensionalidade das amostras e a extração de características. Uma das abordagens mais antigas para abordar esse problema é a PCA (*Principal Component Analysis*) (Jolliffe, 1986). Mais recentemente, arquiteturas de redes neurais vêm sendo utilizadas para essa tarefa, como os métodos de *Autoencoders* (Kingma & Welling, 2014). A utilização da saída de camadas intermediárias de uma arquitetura de CNN pré-treinada também tem sido explorado como método extração de características, realizando o treinamento através de aprendizagem por transferência (ou *Transfer Learning*). Um exemplo de arquiteturas utilizadas para esse caso é a VGG (Simonyan & Zisserman, 2015) como pode ser visto em (Texler *et al.*, 2020), para transferência de estilo. Outros exemplos e problemas onde são abordados *Transfer Learning* podem ser encontrados em (Pan & Yang, 2010) e em (Raina *et al.*, 2007).

3.2 Arquiteturas Híbridas e Modelos Quânticos

Com o desenvolvimento de algoritmos utilizando os fundamentos da mecânica quântica, foi observado que esse tipo de paradigma computacional poderia ser utilizado para desenvolver algoritmos de aprendizagem. Um exemplo básico desses algoritmos é o classificador quântico baseado nas distâncias entre as amostras (ou classificador Hadamard) (Schuld *et al.*, 2017), onde o modelo que representa a classe é definido através de uma amostra de referência do banco de dados. Após o desenvolvimento de métodos para cálculo de gradientes em operadores quânticos, como o *Parameter Shift Rule* (Crooks, 2019) mencionado no Capítulo 2, tornou possível a construção dos circuitos variacionais. Além disso, a computação quântica se mostrou capaz de construir métodos análogos a arquiteturas de *Deep Learning* como *Quantum Convolutional Networks* (QCNN) (Cong *et al.*, 2019), *Quantum Variational Autoencoders* (QVA) (Khoshaman *et al.*, 2018) e *Quantum Generative Adversarial Networks* (QGAN) (Dallaire-Demers & Killoran, 2018).

Nos trabalhos (Schuld, 2021) e (Schuld & Killoran, 2019) é demonstrado que a codificação dos dados no circuito quântico e uma posterior medição podem ser utilizados como função de *kernel*. Recentemente foi apresentado em (Liu *et al.*, 2021) um algoritmo híbrido de *Suport Vector Machine* utilizando como função de *kernel* um computador quântico tolerante a falhas e mostrou que seu desempenho para uma certa distribuição de dados tem vantagem em tempo sobre os métodos clássicos. Em (Pérez-Salinas *et al.*, 2020) foi proposta uma configuração de circuitos variacionais para uma arquitetura híbrida na qual se constrói um análogo às redes MLP, porém, com uma diminuição na quantidade de parâmetros para certos tipo de distribuição de dados. Em (Abbas *et al.*, 2020) é desenvolvida uma métrica para medir a expressividade de modelos e através dela é verificado que redes neurais quânticas alcançam uma dimensão eficaz melhor que as clássicas. Com essa métrica em (Sim *et al.*, 2019) foram medidas a expressividade por quantidade de camada de certas configurações de circuitos variacionais.

Além das técnicas de processamento de imagens em circuitos quânticos, compilados em (Cai *et al.*, 2018), têm sido desenvolvidos algoritmos quânticos para o problema da classificação de imagens. Em (Tacchino *et al.*, 2019) é mostrado um método de classificação de imagens com dimensões 2×2 em escala de cinza capaz de performar em computadores quânticos reais. Também foram desenvolvidos algoritmos híbridos para geração de imagens utilizando QGANs (Huang *et al.*, 2020) e classificação de imagens combinando camadas convolucionais clássicas e quânticas (Henderson *et al.*, 2019). No caso da classificação de imagens utilizando *Transfer Learning* de modelos clássicos para quânticos, em (Mari *et al.*, 2020) é utilizada uma ResNet (He *et al.*, 2015) pré-treinada no conjunto de imagens ImageNet (Deng *et al.*, 2009) para extrair características dadas como entrada de um modelo híbrido, onde é feita a inferência.

4

MÉTODO PROPOSTO

4.1 Visão Geral

Sabendo como são definidos os circuitos quânticos variacionais, podemos utilizá-los em conjunto com métodos de aprendizagem clássicos para modelar um problema de classificação de imagens. De maneira semelhante ao trabalho (Mari *et al.*, 2020), propomos um método de aprendizagem para classificação de imagens utilizando *Transfer Learning* em circuitos parametrizados. Dado um conjunto de dados com imagens e seus respectivos rótulos, utilizamos uma arquitetura de CNN (*Convolutional Neural Networks*) pré-treinada para extrair características que serão codificadas em amplitude no circuito quântico variacional. Após a aplicação das camadas com operadores parametrizados, é aplicada a medição que será a entrada de uma camada linear com uma função de ativação, onde é feita a inferência. Essas etapas podem ser melhor visualizadas na Figura 11.

4.2 Extração de Caraterísticas

Os modelos quânticos se tornam rapidamente custosos para simular em computadores clássicos pelo fato de sua dimensão vetorial crescer exponencialmente com a quantidade de *qubits*. Além disso, os computadores quânticos disponíveis para experimentação ainda são bastante ruidosos e possuem um número baixo de *qubits* (Preskill, 2018). Para tornar a implementação viável de acordo com os recursos disponíveis é preciso reduzir a dimensionalidade de representação das imagens. Logo, a extração de características através de uma arquitetura clássica pré-treinada no conjunto de dados escolhido para realizar a classificação pode ser útil para essa redução, além de diminuir a complexidade da distribuição de cada classe.

4.2.1 *Transfer Learning* utilizando *Mobilenetv2*

Após ser definido o banco de dados com as imagens a serem classificadas, o método escolhido para a extração das características foi o *Mobilenetv2* (Sandler *et al.*, 2019) utilizando

parâmetros pré-treinados[1]. Retirando a última camada linear, onde é feita a inferência, dada uma imagem $\mathbf{x}_m \in \mathbf{R}^{H\times W\times C}$ como entrada, a saída da arquitetura será o vetor de características $\mathbf{x}'_m \in \mathbf{R}^{1280}$. Como precisamos de um vetor com dimensionalidade em potência de 2 e devido a limitações computacionais para simular o circuito quântico para treinamento, é aplicado um método de redução de dimensionalidade para obter o vetor $\mathbf{x}''_m \in \mathbf{R}^N$ de entrada a ser codificado, onde $N = 2^n$ e n o número de *qubits*. Dessas técnicas de redução, foram escolhidas para os experimentos o método PCA (Jolliffe, 1986) e a seleção de características.

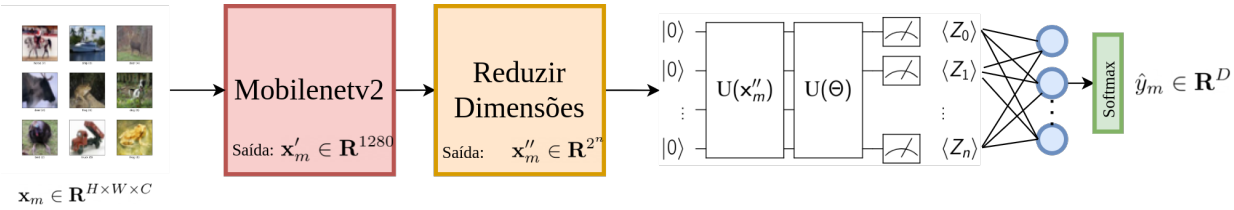

Figura 11: Representação da arquitetura híbrida desenvolvida no nosso método. A imagem $\mathbf{x}_m \in \mathbf{R}^{H\times W\times C}$ com índice m do banco de dados é dada como entrada da arquitetura *Mobilenetv2*. Nas características de saída $\mathbf{x}'_m \in \mathbf{R}^{1280}$ da *Mobinetv2* é aplicado um método para diminuir a dimensionalidade e assim o vetor $\mathbf{x}''_m \in \mathbf{R}^{2^n}$ ser dado como entrada, sendo n o número de *qubits* do circuito variacional. Após ser aplicada a medição com observador Z_i, os resultados $\langle Z_i\rangle$ de cada *qubits* $i = 0, 1, \ldots, n-1$ são dados como entrada de uma camada linear com uma função de ativação *softmax*, resultando no vetor de inferência $\hat{y}_m \in \mathbf{R}^D$, sendo $D \in \mathbf{N}$ o número de classes.

4.2.2 Descrição da Arquitetura *Mobilenetv2*

Na Figura 12 é mostrado o bloco básico utilizado na construção da *Mobilenetv2*, que chamaremos de *Bottleneck residual block*. O primeiro componente desse bloco é formado por uma convolução com *kernel* de dimensões 1×1. Em seguida, é aplicada uma camada de convolução separável por profundidade (ou *depthwise separable convolution*), apresentada em (Chollet, 2017), com *kernel* 3×3. Essa camada de convolução separável pode ser descrita como uma versão fatorizada da convolução completa, reduzindo custo computacional. Todas as camadas, com exceção da última, utilizam a função de ativação ReLU6, que pode ser descrita pela expressão $\text{ReLU}(x) = \min(\max(0, x), 6)$, sendo $x \in \mathbf{R}$, aumentando a robustez para computação de baixa precisão. Na Tabela 1 é mostrada as dimensionalidades de entrada e saída do *Bottleneck* das devidas operações e pode ser verificado que a última camada não possui função de ativação.

Na Tabela 2 é mostrada a configuração completa da arquitetura utilizada para a extração de características, sendo n o número de repetições de cada operação, s o número de *strides*, t a constante de expansão e c o número de canais de saída. No caso onde $n > 1$ e $s > 1$, o *stride* é aplicado apenas uma vez, nas outras repetições $s = 1$. Como pode ser observado, após a aplicação da operação de AvgPool, temos o vetor de características $\mathbf{x}'_m \in \mathbf{R}^{1280}$.

[1]Os parâmetros e implementação da arquitetura utilizados estão disponíveis em `https://github.com/huyvnphan/PyTorch_CIFAR10/blob/master/cifar10_models/mobilenetv2.py`

Entrada	Operação	Saída
$h \times w \times c$	1×1 conv2d, ReLU6	$h \times w \times tc$
$h \times w \times tc$	3×3 dwiseconv2d $s = s$, ReLU6	$\frac{h}{s} \times \frac{h}{s} \times tc$
$\frac{h}{s} \times \frac{h}{s} \times tc$	1×1 conv2d, Linear	$\frac{h}{s} \times \frac{h}{s} \times k'$

Tabela 1: Dimensionalidades de entrada e saída do *Bottleneck* das operações em cada camada, onde c é o número de canais de entrada, t é a constante de expansão e s é o passo da convolução (ou *stride*).

Entrada	Operador	t	c	n	s
$H \times W \times C$	3×3 conv2d	-	32	1	1
$H \times W \times 32$	*Bottleneck*	1	16	1	1
$H \times W \times 16$	*Bottleneck*	6	24	2	1
$H \times W \times 24$	*Bottleneck*	6	32	3	2
$H/2 \times W/2 \times 32$	*Bottleneck*	6	64	4	2
$H/4 \times W/4 \times 64$	*Bottleneck*	6	96	3	1
$H/4 \times W/4 \times 96$	*Bottleneck*	6	160	3	2
$H/8 \times W/8 \times 160$	*Bottleneck*	6	320	1	1
$H/8 \times W/8 \times 320$	1×1 conv2d	-	1280	1	1
$H/8 \times W/8 \times 1280$	$H/8 \times W/8$ AvgPool	-	-	1	-

Tabela 2: Descrição das camadas de operações da *Mobilenetv2* utilizada no nosso método, sendo n o número de repetições de cada operação, s o número de *strides*, t a constante de expansão e c o número de canais de saída. No caso onde $n > 1$ e $s > 1$, o *stride* é aplicado apenas uma vez, nas outras repetições $s = 1$.

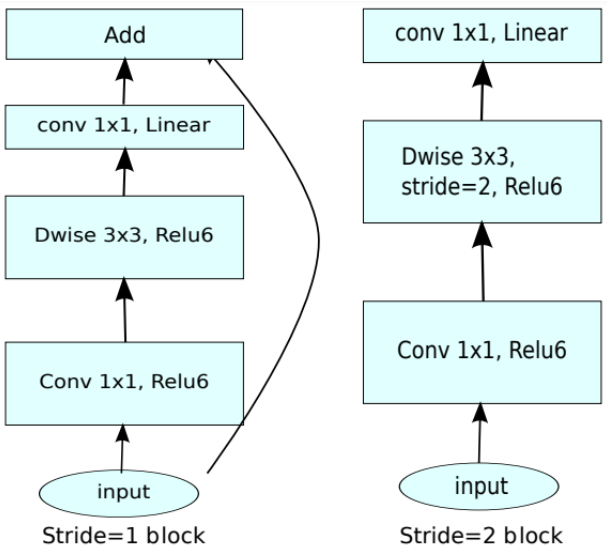

Figura 12: Representação do *Bottleneck residual block* apresentada em (Sandler *et al.*, 2019). Na imagem a esquerda está a representação do bloco com aplicação residual. Na imagem a direito o bloco tem da convolução separavel tem $stride = 2$, logo não pode ser feita uma camada residual pela diferença de dimensionalidade entre a entrada e a saída.

4.3 Codificação em Estados Quânticos

O primeiro passo para construir um circuito variacional é a codificação das características extraídas $\mathbf{x}''_m \in \mathbf{R}^N$ em estados quânticos. Como o treinamento será realizado através de simulação em computadores clássicos, deve ser escolhida uma forma de codificação que reduza o uso de *qubits*. Assim, foi escolhida a codificação por amplitude, pois o circuito necessário para codificar $\mathbf{x}''_m$ precisará de $n = \log_2 N$ *qubits*. O algoritmo escolhido, apresentado em (Möttönen *et al.*, 2005), após normalizar os vetores $\mathbf{x}''_m = (x_0, x_1, \ldots, x_{127})$, faz a operação $|0\rangle \rightarrow |\psi_m\rangle$ no qual $|\psi_m\rangle = \sum_{i=0}^{N} x_i |i\rangle$ é o estado de saída, sendo $|0\rangle$ o estado inicial.

Na Figura 13 é mostrado um exemplo do circuito resultante da codificação de um vetor $\mathbf{x} \in \mathbf{R}^{2^n}$ com $n = 3$ *qubits* aplicando portas de rotação $\mathrm{R}_y(\alpha_j^i)$ com ângulos α_j^i onde $i = 0, 1, \ldots, n-1$ é o índice dos *qubits* e $j = 0, 1, \ldots, 2^i - 1$ o índice da porta de rotação controlada no *qubit* i. O algoritmo para extrair os ângulos α_j^i é descrito em Algoritmo 1 e para construir o circuito são aplicados uma sequência de 2^i operadores de rotação $\mathrm{R}_y(\alpha_j^i)$ em cada *qubit* i controlado pelo estado $\left| j_0, j_1, \ldots, j_{log_2(j)} \right\rangle$ onde $j_0, j_1, \ldots, j_{log_2(j)}$ é a representação binária do índice j.

Algoritmo 1: Algoritmo `get_alphas`

Entrada: vetor de amostra do banco de dados $\mathbf{x} \in \mathbf{R}^N$;
Saída: vetor $A \in \mathbf{R}^{N-1}$ com os ângulos α;
if $dims(\mathbf{x}) > 1$ **then**
    Criar vetor auxiliar $\mathbf{x}_{new} \in \mathbf{R}^{N/2}$;
    for $k \leftarrow 0$ ***to*** $dims(\mathbf{x}_{new})$ **do**
        $\mathbf{x}_{new}[k] = \sqrt{|\mathbf{x}[2k]|^2 + |\mathbf{x}[2k+1]|^2}$
    A_{inner} =`get_alphas` ($\mathbf{x}_{new}$)
    Criar vetor auxiliar $A \in \mathbf{R}^{N/2}$;
    for $k \leftarrow 0$ ***to*** $dims(\mathbf{x}_{new})$ **do**
        if $\mathbf{x}_{new} \neq 0$ **then**
            if $\mathbf{x}[2k] > 0$ **then**
                $A[k] = 2\mathrm{arcsen}\left(\frac{\mathbf{x}[2k+1]}{\mathbf{x}_{new}[k]}\right)$
            else
                $A[k] = 2\pi - 2\mathrm{arcsen}\left(\frac{\mathbf{x}[2k+1]}{\mathbf{x}_{new}[k]}\right)$
        else
            $A[k] = 0$
    $A = concatenate(A_{inner}, A)$
    return A

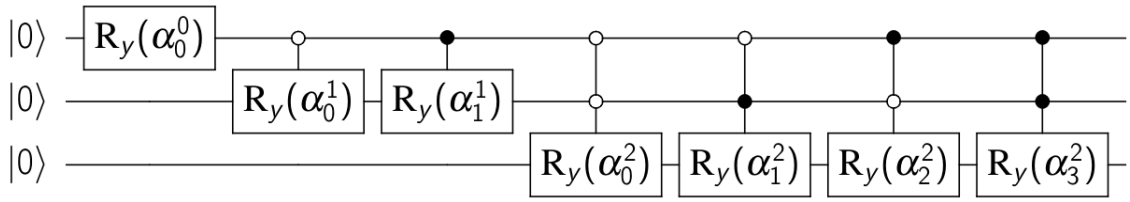

Figura 13: Circuito para a codificação de um vetor $\mathbf{x} \in \mathbf{R}^8$ no estado $|\mathbf{x}_m\rangle$ com 3 *qubits* aplicando portas de rotação controladas pelo conjunto de ângulos α_i para $i = 0, 1, ..., 6$.

4.4 Camadas do Circuito Quântico Parametrizado

Para construir o circuito com portas parametrizadas aplicamos em cada *qubit* um porta de rotação $R_y(\theta_l^i)$ sendo θ o parâmetro a ser aprendido do *qubit* $i = 0, 1, \ldots, n-1$ da camada $l = 0, 1, \ldots, L-1$ onde n é o número de *qubits* e L o número de camadas. Logo após a porta de rotação, é aplicada a porta CNOT onde o *qubit* i controla o $i+1$ em cada camada l. Para fazer a medição é aplicado como observador o operador Z_i com seu valor esperado $\langle Z_i \rangle$ em cada *qubit* i. Esse tipo de circuito pode ser visto em (Mari *et al.*, 2020), onde a codificação das características é feita por produto tensorial, e em (Sim *et al.*, 2019). Assim, podemos concluir que o circuito parametrizado junto com a codificação em amplitude modela uma função

$$f(\mathbf{x}, \Theta) = \begin{bmatrix} \langle \psi_\Theta | Z \otimes I \otimes \cdots \otimes I | \psi_\Theta \rangle \\ \langle \psi_\Theta | I \otimes Z \otimes \cdots \otimes I | \psi_\Theta \rangle \\ \vdots \\ \langle \psi_\Theta | I \otimes I \otimes \cdots \otimes Z | \psi_\Theta \rangle \end{bmatrix} = \begin{bmatrix} \langle Z_0 \rangle \\ \langle Z_1 \rangle \\ \vdots \\ \langle Z_n \rangle \end{bmatrix} \tag{4.1}$$

onde $\mathbf{x}$ é o vetor de entrada do circuito e $|\psi_\Theta\rangle$ é o estado do circuito após a aplicação dos operadores com os parâmetros $\theta_l^i \in \Theta$ no estado $|\mathbf{x}\rangle$, onde $\Theta \in \mathbf{R}^{nL}$. Na Figura 14 é mostrado o exemplo do circuito variacional escolhido para um modelo quântico com $n = 3$, $L = 2$ sendo $\mathbf{x} \in \mathbf{R}^8$ o vetor codificado em amplitude no estado $|\mathbf{x}\rangle$.

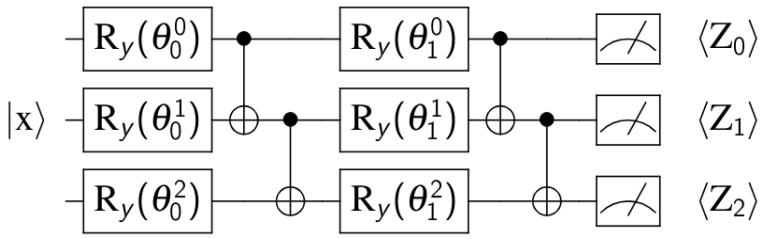

Figura 14: circuito para um modelo quântico com $n = 3$, $L = 2$ e com observador Z_i em cada *qubit* i, sendo $\langle Z_i \rangle$ o valor esperado da medição e $|\mathbf{x}\rangle$ o estado de entrada.

4.5 Camada de Inferência e Treinamento

Após ser feita as medições em cada *qubit*, é aplicada uma camada linear com pesos $W \in \mathbf{R}^{D\times n}$ e uma função *softmax* representada por σ. Assim, podemos dizer que o vetor $\hat{y} = (\hat{y}_0, \hat{y}_1, \ldots, \hat{y}_{D-1})$, sendo D o número de classes, representando a inferência dada pela arquitetura híbrida pode ser descrito pela equação

$$\hat{y}_j = \sigma(W \cdot f(\mathbf{x}, \Theta) + b), \tag{4.2}$$

sendo $b \in \mathbf{R}^D$ e a operação $(\cdot)$ o produto matricial. Para realizar o treinamento, foi escolhida a função de perda *Categorical Cross-Entropy* e para atualizar os pesos foi utilizado o otimizador Adam (Kingma & Ba, 2017).

5

EXPERIMENTOS

5.1 Metodologia

Para verificar o desempenho da arquitetura híbrida proposta, foram escolhidos os conjuntos dados de imagens com dígitos (chamaremos de DIGITS), disponível na biblioteca da linguagem Python Scikit[1], e o CIFAR10 (Krizhevsky, 2009). Foram realizados experimentos utilizando diferentes configurações de arquitetura de acordo com a quantidade de classes e comparamos o desempenho com arquiteturas de redes neurais clássicas. A extração de características da *Mobilenetv2* pré-treinada no conjunto CIFAR10 foi retirada da implementação citada no Capítulo 4. Para a implementação e treinamento do modelo quântico foi utilizada a ferramenta Tensorflow Quantum[2] da linguagem Python.

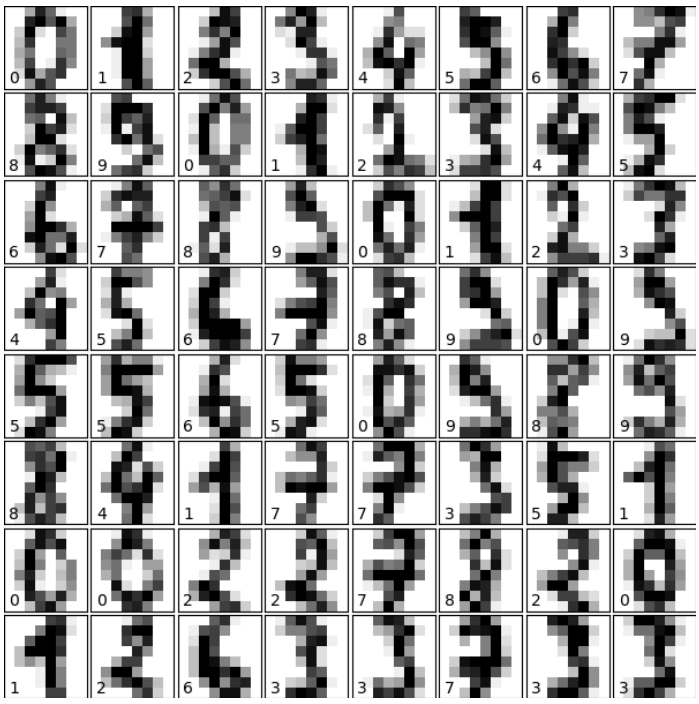

Figura 15: Amostras aleatórias do conjunto de dados DIGITS de cada uma das 10 classes.

[1]Descrição de como o conjunto de dados é carregado em `https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_digits.html`

[2]`https://www.tensorflow.org/quantum?hl=pt-br`

5.2 Conjuntos de dados utilizados

O conjunto de dados DIGITS é composto por imagens com dimensões 8×8 em escala de cinza, assim podemos considerar suas amostras como um vetor $\mathbf{x}_m \in \mathbf{R}^{64}$. Com isso, suas imagens foram utilizadas diretamente como entrada do circuito quântico proposto utilizando apenas 6 *qubits*. Esse conjunto foi escolhido para serem realizados testes sobre o desempenho do modelo quântico sem a extração de características. Na Figura 15 são mostradas algumas amostras desse conjunto de dados.

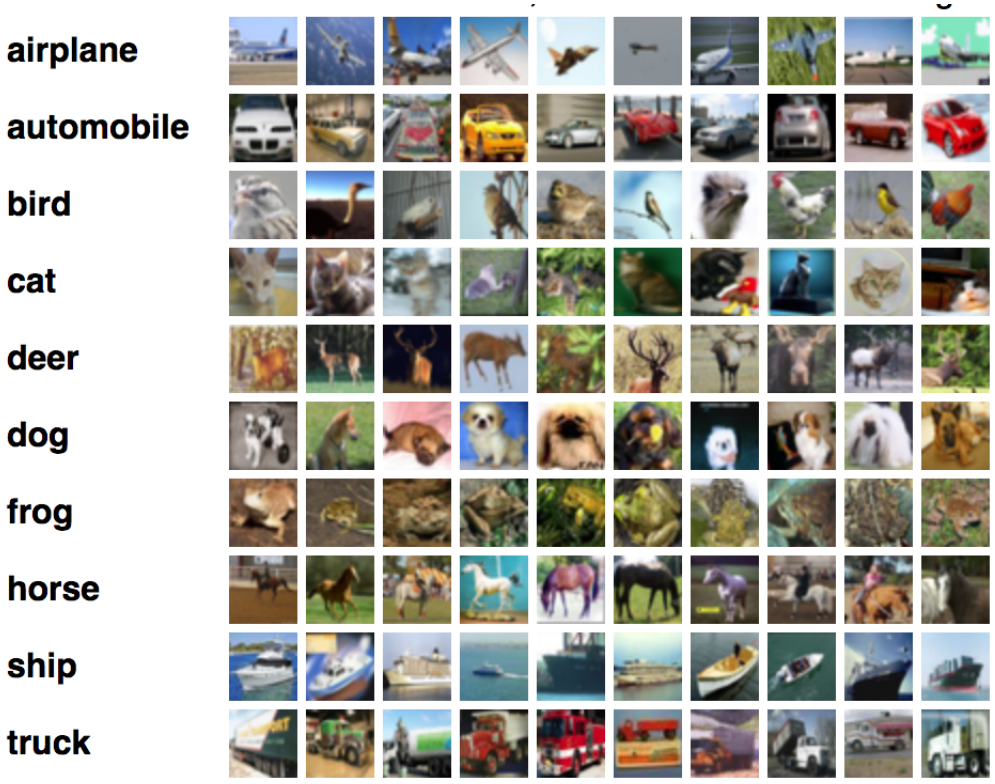

Figura 16: Amostras aleatórias do conjunto de dados CIFAR10 de cada uma das 10 classes.

No caso do CIFAR10, as imagens têm dimensões $32 \times 32 \times 3$, o que o torna um conjunto com bastante complexidade na distribuição das classes além de possuir uma dimensionalidade que torna difícil a simulação para o treinamento do circuito variacional com os recursos computacionais disponíveis. Esse problema é contornado utilizando características extraídas numa arquitetura *Mobilenetv2* pré-treinada nesse conjunto e depois aplicando métodos para reduzir a dimensionalidade, como descrito no Capítulo 4. Um dos métodos escolhidos foi o PCA que converte o vetor de características $\mathbf{x}'_m \in \mathbf{R}^{1280}$ para o vetor de entrada $\mathbf{x}''_m \in \mathbf{R}^{128}$. O segundo método escolhido foi a seleção de características, no qual as 512 primeiras características das 1280 da *Mobilenetv2* são escolhidas. Assim, temos o vetor $\mathbf{x}''_m \in \mathbf{R}^{512}$ como entrada do circuito.

Com esses dois métodos temos duas configurações de circuitos variacionais, uma com $n = 7$, utilizando redução através do PCA, e outra com $n = 9$, com redução através da seleção de

características. Na Figura 16 são mostradas algumas imagens das classes presentes no conjunto de dados. Foram escolhidas para os experimentos as duas e as quatro primeiras classes, 5000 amostras de treino e 1500 amostras de teste.

5.3 Arquiteturas Clássicas para Comparação

Como queremos mostrar que a arquitetura híbrida pode ter desempenho próximo a de métodos clássicos utilizando uma quantidade inferior de parâmetros, desenvolvemos 4 arquiteturas para comparar os resultados do treinamento. A primeira arquitetura (i.e. Arquitetura 1), mostrada na Figura 17, consiste de uma camada linear com matriz de pesos $W \in \mathbf{R}^{D\times 2^n}$, representada pela equação

$$\hat{y}_m = \sigma(W \cdot \mathbf{x}''_m + b), \tag{5.1}$$

sendo σ a função *softmax* e $b \in \mathbf{R}^D$, onde D é a quantidade de classes.

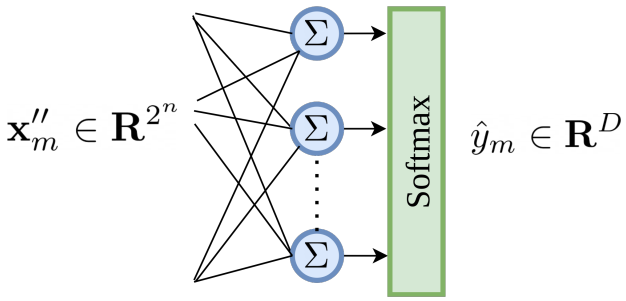

Figura 17: Representação da Arquitetura 0, utilizada como base para comparar os resultados.

Para reduzir o crescimento do número de parâmetros com o aumento das características e de classes, são propostas as Arquiteturas 1 e 2, mostradas nas Figuras 18 e 19 respectivamente. Essas arquiteturas podem ser descritas de maneira geral como um primeira camada linear com função de ativação ReLU e em sequência outra camada linear acompanhada da função *softmax*, podendo ser representadas pela expressão

$$\hat{y}_m = \sigma(W_1 \cdot ReLU(W_0 \cdot \mathbf{x}''_m + b_0) + b_1), \tag{5.2}$$

onde $W_0 \in \mathbf{R}^{1\times 2^n}$, $W_1 \in \mathbf{R}^{D\times 1}$ e $b_0 \in \mathbf{R}$ no caso da Arquitetura 1 e $W_0 \in \mathbf{R}^{2\times 2^n}$, $W_1 \in \mathbf{R}^{D\times 1}$ e $b_0 \in \mathbf{R}^2$ para a Arquitetura 2, sendo $b_1 \in \mathbf{R}^D$ para ambos os casos.

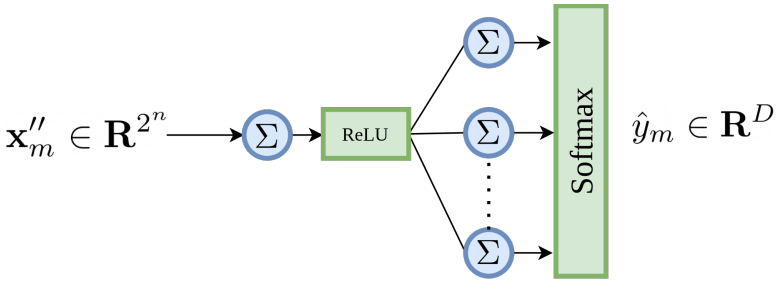

Figura 18: Representação da Arquitetura 1, utilizada como base para comparar os resultados.

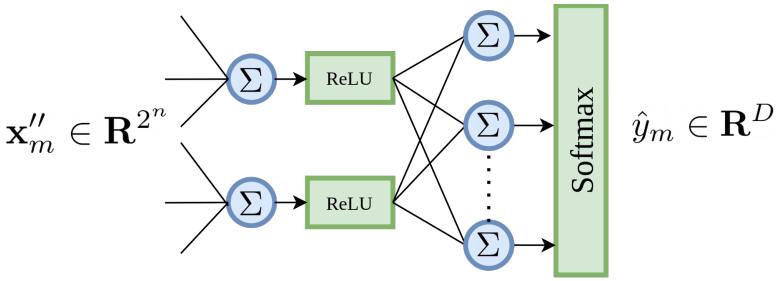

Figura 19: Representação da Arquitetura 2, utilizada como base para comparar os resultados.

Mesmo havendo uma redução no crescimento da quantidade de parâmetros de acordo com o tamanho da entrada e a quantidade de classes, as configurações escolhidas na Arquitetura 1 e 2 podem ter o seu desempenho bastante reduzido dependendo do conjunto de dados, tornando a comparação injusta. Para comparar com camadas mais robustas de redes neurais que podem reduzir a quantidade de parâmetros, foi desenvolvida a Arquitetura 3, mostrada na Figura 20. Essa arquitetura consiste de uma primeira camada convolucional com *kernel* de dimensão $k = 8$, *stride* de tamanho $s = 6$ e $c = 2$ filtros (ou camadas de saída). A última camada dessa arquitetura, utilizada para inferência, é equivalente à última camada linear com *softmax* presente nas arquiteturas 1 e 2.

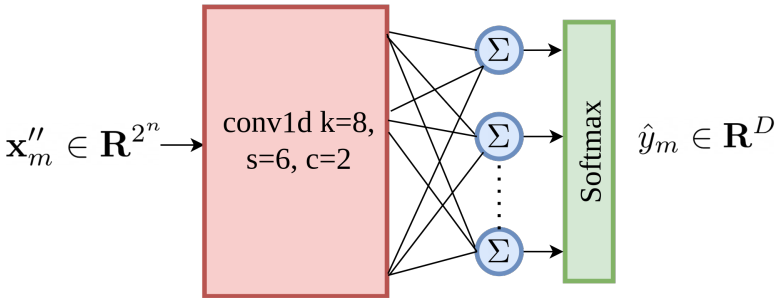

Figura 20: Representação da Arquitetura 3, utilizada como base para comparar os resultados.

6

RESULTADOS

6.1 Resultados do treinamento no DIGITS

Na Tabela 3 vemos a acurácia e os hiperparâmetros dos treinamentos do modelo quântico utilizando o conjunto de dados DIGITS. Podemos observar que o circuito variacional foi capaz de convergir para 100% de acurácia no conjunto de validação utilizando o conjunto de dados com duas classes (i.e. imagens apenas com os dígitos 0 e 1), alcançando a arquitetura 0 que possui uma quantidade maior de parâmetros, mostrado na Tabela 4. Ainda é valido notar que a Arquitetura 3 também obteve o mesmo resultado com um número menor de parâmetros. O modelo quântico também atingiu um nível de acurácia satisfatório quando o treinamento é feito com 4 classes (i.e. imagens com os dígitos 0,1,2 e 3), obtendo um resultado próximo a Arquitetura 0, porém inferior a Arquitetura 3, como pode ser visto na Tabela 5. No caso do conjunto inteiro, utilizando as 10 classes, na Tabela 6 é mostrado que o nível de acurácia obtido foi próximo ao da Arquitetura 3 com apenas 22 parâmetros a mais. Esses resultados demonstram a capacidade do método quântico de generalizar as classes do conjunto de dados DIGITS de uma maneira próxima a métodos clássicos mais robustos ou com número próximo de parâmetros.

Além de ter desempenho próximo às arquiteturas 0 e 3, o modelo quântico obteve uma acurácia superior a dos modelos das arquiteturas 1 e 2. No caso do treinamento do conjunto utilizando 4 classes, podemos observar que a arquitetura 2 obteve uma acurácia bem abaixo da arquitetura 3 e o modelo híbrido mesmo contendo um número superior de parâmetros. Com isso,

Número de Classes	2	4	10
Número de Épocas	30	30	50
Número de Camadas	12	15	30
Tamanho do *Batch*	32	32	20
Taxa de Aprendizado	0.1	0.1	0.1
Número de Parâmetros	86	118	250
Acurácia	100%	96,29%	87.59%

Tabela 3: Tabela dos resultados do treinamento do modelo quântico utilizando o conjunto de dados DIGITS.

Arquitetura	0	1	2	3
Número de Parâmetros	130	69	85	60
Acurácia	100%	46.30%	10%	100%

Tabela 4: Tabela dos resultados do treinamento nas arquiteturas clássicas utilizando o conjunto de dados DIGITS com apenas as 2 primeiras classes, dígitos 0 e 1. A configuração de hiper-parâmetros utilizada foi um tamanho de *batch* igual a 16, 50 épocas, taxa de aprendizado igual 0.01 com o otimizador Adam.

Arquitetura	0	1	2	3
Número de Parâmetros	260	73	142	102
Acurácia	100%	19.91%	21.76%	99.07%

Tabela 5: Tabela dos resultados do treinamento nas arquiteturas clássicas utilizando o conjunto de dados DIGITS com as 4 primeiras classes, dígitos 0,1,2 e 3. A configuração de hiper-parâmetros utilizada foi um tamanho de *batch* igual a 16, 50 épocas, taxa de aprendizado igual 0.01 com o otimizador Adam.

no conjunto DIGITS pode ser concluído que o desempenho do método quântico foi próximo da arquitetura 3, obtendo acurácias próximas com diferença de 13 a 22 parâmetros.

6.2 Resultados do treinamento no CIFAR10

Nas Tabelas 7 e 10 são observados os resultados em 2 cenários. O primeiro, é feita diminuição de dimensionalidade das características extraídas da *Mobilenetv2*, como descrito no Capítulo 4, utilizando PCA e assim o vetor resultante é dado como entrada do circuito quântico. O segundo cenário para gerar o vetor de entrada é feita uma seleção de 512 das 1280 características da *Mobilenetv2*.

6.2.1 PCA

Na Tabela 7 vemos o resultado do modelo quântico para o conjunto no cenário utilizando o PCA e seus respectivos hiper-parâmetros do treinamento. Na Tabela 8 temos os resultados do treinamento nas arquiteturas clássicas para o conjunto com as classes "airplane"e "automobile". Notamos que o modelo híbrido obteve uma acurácia inferior em relação aos clássicos, com uma diferença de 4.33% para a arquitetura clássica com menor número de parâmetros. Os resultados

Arquitetura	0	1	2	3
Número de Parâmetros	650	85	160	228
Acurácia	99.54%	9.63%	8.7%	88.70%

Tabela 6: Tabela dos resultados do treinamento nas arquiteturas clássicas utilizando o conjunto de dados DIGITS com todas as 10 classes. A configuração de hiper-parâmetros utilizada foi um tamanho de *batch* igual a 16, 50 épocas, taxa de aprendizado igual 0.01 com o otimizador Adam.

Número de Classes	2	4
Número de Épocas	10	20
Número de Camadas	25	30
Tamanho do *Batch*	16	8
Taxa de Aprendizado	0.1	0.1
Número de Parâmetros	191	208
Acurácia	95.47%	87.7%

Tabela 7: Tabela dos resultados do treinamento do modelo quântico com o conjunto de dados CIFAR10 utilizando PCA.

Arquitetura	0	1	2	3
Número de Parâmetros	258	133	264	104
Acurácia	99.80%	98.80%	98.80%	98.80%

Tabela 8: Tabela dos resultados do treinamento nas arquiteturas clássicas com o conjunto de dados CIFAR10 utilizando PCA e as classes "airplane"e "automobile". Sendo a configuração de hiper-parâmetros: tamanho de *batch* igual a 16, 50 épocas, taxa de aprendizado igual 0.01 com o otimizador Adam.

do treinamento utilizando as classes "airplane", "automobile, "bird"e "cat são mostrados na Tabela 9. Nesse caso os modelos híbridos saíram pior que os clássicos com uma diferença 4.7% para a arquitetura com menor número de parâmetros.

Com o PCA como método para reduzir a dimensionalidade observamos que o modelo quântico não foi capaz de generalizar os conjuntos de dados com o mesmo desempenho que os modelos clássicos. Isso é concluído pois, mesmo contendo um número superior de parâmetros à arquitetura 1, sua acurácia é inferior nos dois cenários apresentados.

6.2.2 Seleção de Características

Na Tabela 10 vemos o resultado para o conjunto no cenário utilizando as 512 primeiras características da ultima camada do *Mobilenetv2* e os respectivos hiper-parâmetros. Na Tabela 11 temos os resultados do treinamento nas arquiteturas clássicas para o conjunto com as classes "airplane"e "automobile". Pode ser observado que apesar do modelo quântico obter uma acurácia inferior aos clássicos, tem uma diferença de $\approx 5.80\%$ para a arquitetura com menor número de parâmetros, mesmo esta possuindo ≈ 4 vezes mais parâmetros que o método híbrido. Os resulta-

Arquitetura	0	1	2	3
Número de Parâmetros	516	137	270	190
Acurácia	94.80%	92.40%	94.47%	94.93%

Tabela 9: Tabela dos resultados do treinamento nas arquiteturas clássicas utilizando o conjunto de dados CIFAR10 utilizando PCA e as classes "airplane", "automobile, "bird"e "cat. Sendo a configuração de hiper-parâmetros: tamanho de *batch* igual a 16, 50 épocas, taxa de aprendizado igual 0.01 com o otimizador Adam.

Número de Classes	2	4
Número de Épocas	5	20
Número de Camadas	8	10
Tamanho do *Batch*	8	8
Taxa de Aprendizado	0.1	0.1
Número de Parâmetros	92	128
Acurácia	93.13%	93.47%

Tabela 10: Tabela dos resultados do treinamento do modelo quântico com o conjunto de dados CIFAR10 escolhendo 512 características da *Mobilenetv2* pré-treinada.

Arquitetura	0	1	2	3
Número de Parâmetros	1026	517	1032	360
Acurácia	98.80%	98.93%	98.80%	98.93%

Tabela 11: Tabela dos resultados do treinamento nas arquiteturas clássicas com o conjunto de dados CIFAR10 utilizando 512 características e as classes "airplane"e "automobile". A configuração de hiper-parâmetros utilizada foi um tamanho de *batch* igual a 16, 50 épocas, taxa de aprendizado igual 0.01 com o otimizador Adam.

dos do treinamento com as classes "airplane", "automobile, "bird"e "cat"pode ser observado na Tabela 12. Esse foi o cenário no qual o modelo híbrido obteve melhor desempenho, pois obteve uma acurácia próxima a todos métodos clássicos, com maior diferença sendo de 1.8% pontos, mesmo contendo uma quantidade de parâmetros ≈ 4 vezes inferior a menor arquitetura.

De maneira geral, com a seleção de características é observado que os modelos híbridos tiveram um desempenho próximo aos clássicos mesmo possuindo uma quantidade inferior de parâmetros. No caso do conjunto de dados utilizando 4 classes, é observado que o desempenho do modelo híbrido supera o da arquitetura 1. Essa proximidade dos resultados no cenário com 4 classes também é observada no caso da arquitetura 2 que, mesmo este possuindo ≈ 8 vezes o número de parâmetros que o modelo híbrido, obteve acurácia $\approx 0.33\%$ maior.

Arquitetura	0	1	2	3
Número de Parâmetros	2052	521	1038	702
Acurácia	95.27%	93.13%	93.80%	93.80%

Tabela 12: Tabela dos resultados do treinamento nas arquiteturas clássicas utilizando o conjunto de dados CIFAR10 utilizando 512 características e as classes "airplane", "automobile, "bird"e "cat. A configuração de hiper-parâmetros utilizada foi um tamanho de *batch* igual a 16, 50 épocas, taxa de aprendizado igual 0.01 com o otimizador Adam.

7

CONCLUSÃO

Neste trabalho podemos verificar que os modelos quânticos podem ser utilizados em conjunto com métodos clássicos para problemas de classificação de imagens. Isso foi demonstrado através da construção de uma arquitetura híbrida utilizando características de uma *Mobilenetv2* pré-treinada, redução de dimensionalidade e circuitos variacionais com codificação por amplitude.

Com a codificação em amplitude, a dimensionalidade após a medição é reduzida em fator exponencial em relação ao vetor clássico codificado como entrada. Logo, dada uma amostra de entrada de tamanho N, a quantidade de parâmetros da camada de inferência será multiplicada por um fator $\log_2(N)$, caso a medição seja feita em todos os *qubits*.

Para comparar a capacidade de generalização de acordo com o conjunto de dados, foram projetadas 4 arquiteturas clássicas. Nos os experimentos realizados, os métodos híbridos obtiveram resultados próximos aos clássicos, com cenários onde o modelo híbrido obteve uma acurácia superior mesmo com um número inferior de parâmetros.

7.1 Trabalhos Futuros

Os métodos quânticos ainda possuem um certo grau de liberdade quando se trata na definição dos circuitos parametrizados. Levando a explorar no futuro outras abordagens, como funções de perda mais robustas e outros tipos de configurações do circuito (e.g recarregamento do vetor de entrada no circuito entre camadas parametrizadas e portas de rotação controladas por redes neurais). Porém, os métodos de aprendizagem baseados em circuitos quânticos variacionais são bastante recentes e ainda possuem problemas em aberto quando relacionado a vantagens sobre os métodos clássicos.

Pelo fato do alto custo de simulação dos algoritmos quânticos, a eficiência desses modelos só pode ser verificada através de computadores quânticos reais. Os computadores quânticos existentes ainda são bastante ruidosos e possuem pequeno número de *qubits* fazendo com que tenha utilidade bastante limitada. Esse ruído faz com que os circuitos variacionais existentes na literatura utilizem bancos de dados com amostras com pouca dimensionalidade vetorial. Também faz necessário o uso de codificação em estados quânticos diferentes dá por amplitude, que tem

uma complexidade assintótica linear em relação a dimensão do vetor de entrada (i.e. exponencial em relação a quantidade de *qubits*) aumentando a quantidade de ruído adicionado pelo circuito. Com isso, ainda existe muito a ser desenvolvido pela comunidade acadêmica para tornar viável a utilização desses modelos e explorar de forma ampla suas vantagens.

REFERÊNCIAS

Abbas, A., Sutter, D., Zoufal, C., Lucchi, A., Figalli, A., & Woerner, S. (2020). The power of quantum neural networks.

Cai, Y., Lu, X., & Jiang, N. (2018). A survey on quantum image processing. *Chinese Journal of Electronics*, 27:718–727.

Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions.

Cong, I., Choi, S., & Lukin, M. D. (2019). Quantum convolutional neural networks. *Nature Physics*, 15(12):1273–1278.

Crooks, G. E. (2019). Gradients of parameterized quantum gates using the parameter-shift rule and gate decomposition.

Dallaire-Demers, P.-L. & Killoran, N. (2018). Quantum generative adversarial networks. *Phys. Rev. A*, 98:012324.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 248–255.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., & Houlsby, N. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *CoRR*, abs/2010.11929.

Farhi, E. & Neven, H. (2018). Classification with quantum neural networks on near term processors.

Guerreschi, G. G. & Smelyanskiy, M. (2017). Practical optimization for hybrid quantum-classical algorithms.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep residual learning for image recognition. *CoRR*, abs/1512.03385.

Henderson, M., Shakya, S., Pradhan, S., & Cook, T. (2019). Quanvolutional neural networks: Powering image recognition with quantum circuits.

Huang, H.-L., Du, Y., Gong, M., Zhao, Y., Wu, Y., Wang, C., Li, S., Liang, F., Lin, J., Xu, Y., Yang, R., Liu, T., Hsieh, M.-H., Deng, H., Rong, H., Peng, C.-Z., Lu, C.-Y., Chen, Y.-A., Tao, D., Zhu, X., & Pan, J.-W. (2020). Experimental quantum generative adversarial networks for image generation.

Johnson, J., Alahi, A., & Li, F. (2016). Perceptual losses for real-time style transfer and super-resolution. *CoRR*, abs/1603.08155.

Jolliffe, I. T. (1986). *Principal Component Analysis*. Springer-Verlag, Berlin; New York.

Khoshaman, A., Vinci, W., Denis, B., Andriyash, E., Sadeghi, H., & Amin, M. H. (2018). Quantum variational autoencoder. *Quantum Science and Technology*, 4(1):014001.

Kingma, D. P. & Ba, J. (2017). Adam: A method for stochastic optimization.

Kingma, D. P. & Welling, M. (2014). Auto-encoding variational bayes.

Krizhevsky, A. (2009). Learning multiple layers of features from tiny images. Technical report.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C. J. C., Bottou, L., & Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems*, 25.

LeCun, Y., Haffner, P., Bottou, L., & Bengio, Y. (1999). Object recognition with gradient-based learning. In *Shape, Contour and Grouping in Computer Vision*, 319.

Liu, Y., Arunachalam, S., & Temme, K. (2021). A rigorous and robust quantum speed-up in supervised machine learning. *Nature Physics*.

Mari, A., Bromley, T. R., Izaac, J., Schuld, M., & Killoran, N. (2020). Transfer learning in hybrid classical-quantum neural networks. *Quantum*, 4:340.

Möttönen, M., Vartiainen, J., Bergholm, V., & Salomaa, M. (2005). Transformation of quantum states using uniformly controlled rotations. *Quantum Inf. Comput.*, 5:467–473.

Pan, S. J. & Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359.

Preskill, J. (2018). Quantum computing in the nisq era and beyond. *Quantum*, 2:79.

Pérez-Salinas, A., Cervera-Lierta, A., Gil-Fuster, E., & Latorre, J. I. (2020). Data re-uploading for a universal quantum classifier. *Quantum*, 4:226.

Raina, R., Battle, A., Lee, H., Packer, B., & Ng, A. Y. (2007). Self-taught learning: Transfer learning from unlabeled data. In *Proceedings of the 24th International Conference on Machine Learning*, 759–766.

Redmon, J., Divvala, S. K., Girshick, R. B., & Farhadi, A. (2015). You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640.

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (2019). Mobilenetv2: Inverted residuals and linear bottlenecks.

Schuld, M. (2021). Supervised quantum machine learning models are kernel methods.

Schuld, M., Fingerhuth, M., & Petruccione, F. (2017). Implementing a distance-based classifier with a quantum interference circuit. *EPL (Europhysics Letters)*, 119(6):60002.

Schuld, M. & Killoran, N. (2019). Quantum machine learning in feature hilbert spaces. *Phys. Rev. Lett.*, 122:040504.

Shor, P. W. (1997). Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509.

Sim, S., Johnson, P. D., & Aspuru-Guzik, A. (2019). Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms. *Advanced Quantum Technologies*, 2(12):1900070.

Simonyan, K. & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition.

Tacchino, F., Macchiavello, C., Gerace, D., & Bajoni, D. (2019). An artificial neuron implemented on an actual quantum processor. *npj Quantum Information*, 5(1).

Texler, O., Futschik, D., Fišer, J., Lukáč, M., Lu, J., Shechtman, E., & Sýkora, D. (2020). Arbitrary style transfer using neurally-guided patch-based synthesis. *Computers & Graphics*, 87:62–71.

Ulyanov, D., Vedaldi, A., & Lempitsky, V. S. (2017). Deep image prior. *CoRR*, abs/1711.10925.

Verdon, G., McCourt, T., Luzhnica, E., Singh, V., Leichenauer, S., & Hidary, J. (2019). Quantum graph neural networks.