Universidade Estadual de Londrina

MATEUS KOMARCHESQUI

DETECÇÃO DE ANOMALIAS EM REDES DE COMPUTADORES UTILIZANDO COMPUTAÇÃO QUÂNTICA

LONDRINA
2024

MATEUS KOMARCHESQUI

DETECÇÃO DE ANOMALIAS EM REDES DE COMPUTADORES UTILIZANDO COMPUTAÇÃO QUÂNTICA

Trabalho de Conclusão de Curso apresentado ao curso de Bacharelado em Ciência da Computação da Universidade Estadual de Londrina para obtenção do título de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Mario Lemes Proença Jr.

LONDRINA

2024

Ficha de identificação da obra elaborada pelo autor, através do Programa de Geração Automática do Sistema de Bibliotecas da UEL

Sobrenome, Nome.
Título do Trabalho : Subtitulo do Trabalho / Nome Sobrenome. - Londrina, 2017.
100 f. : il.

Orientador: Nome do Orientador Sobrenome do Orientador.
Coorientador: Nome Coorientador Sobrenome Coorientador.
Dissertação (Mestrado em Ciência da Computação) - Universidade Estadual de Londrina, Centro de Ciências Exatas, Programa de Pós-Graduação em Ciência da Computação, 2017.
Inclui bibliografia.

1. Assunto 1 - Tese. 2. Assunto 2 - Tese. 3. Assunto 3 - Tese. 4. Assunto 4 - Tese. I. Sobrenome do Orientador, Nome do Orientador. II. Sobrenome Coorientador, Nome Coorientador. III. Universidade Estadual de Londrina. Centro de Ciências Exatas. Programa de Pós-Graduação em Ciência da Computação. IV. Título.

MATEUS KOMARCHESQUI

DETECÇÃO DE ANOMALIAS EM REDES DE COMPUTADORES UTILIZANDO COMPUTAÇÃO QUÂNTICA

Trabalho de Conclusão de Curso apresentado ao curso de Bacharelado em Ciência da Computação da Universidade Estadual de Londrina para obtenção do título de Bacharel em Ciência da Computação.

BANCA EXAMINADORA

Orientador: Prof. Dr. Mario Lemes Proença Jr.
Universidade Estadual de Londrina

Prof. Dr. Elieser Botelho Manhas Jr.
Universidade Estadual de Londrina

Vitor Gabriel da Silva Ruffo
Universidade Estadual de Londrina

Londrina, 30 de abril de 2024.

Dedico este trabalho aos audazes curiosos que enxergaram potencial computacional no emaranhamento que é a mecânica quântica.

AGRADECIMENTOS

Aos meus pais, por todo o esforço e dedicação que me garantiram a educação necessária para escrever este trabalho. E aos demais familiares pelo apoio e amor, em especial Felipe e Maria Julia.

Ao professor Dr. Mario Lemes Proença Jr. e ao grupo de pesquisa ORION da Universidade Estadual de Londrina, que muito me engrandeceram profissional e academicamente e me inspiram a fazer ciência.

Aos professores do curso de Ciência da Computação, por me proporcionarem amadurecimento e direção no decorrer da minha graduação.

Ao Conselho Nacional de Desenvolvimento Científico e Tecnológico, por prover recursos que viabilizam o desenvolvimento de pesquisa científica.

Aos servidores da UEL, pelo seu trabalho essencial que mantém a universidade funcionando como um organismo.

"Se você acha que entendeu física quântica,
é porque você não entendeu.
(Richard Feynman)"

KOMARCHESQUI, M.. **Detecção de anomalias em redes de computadores utilizando Computação Quântica**. 2024. 90f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Universidade Estadual de Londrina, Londrina, 2024.

RESUMO

Dada a ampla aplicação das redes de computadores e o potencial de exposição de dispositivos e dados, o que pode servir como catalisador para ataques, o investimento na pesquisa e desenvolvimento de métodos para assegurar a segurança nesse meio faz-se primordial. Os Sistemas de Detecção de Intrusão, como proposto neste trabalho, monitoram o tráfego da rede e determinam em tempo satisfatório se dado fluxo de dados é considerado normal ou anômalo, detectando possíveis ameaças. Ao longo dos anos, diversas implementações de Sistemas de Detecção de Intrusão de Rede utilizaram Aprendizado de Máquina no âmbito da computação clássica. Mesmo com modelos complexos de Aprendizado Profundo, a discussão acerca de detecção de intrusão de rede continua em aberto. Almeja-se com este trabalho estudar e implementar um modelo de Aprendizado de Máquina Híbrido clássico-quântico, mostrando as particularidades desse novo paradigma e explorando sua viabilidade no contexto proposto. Três variações do sistema utilizando Aprendizado de Máquina Raso foram implementados: um modelo completamente clássico e dois modelos híbridos clássico-quânticos. Ambos os sistemas foram executados no servidor remoto do Grupo Orion, uma máquina clássica capaz de simular o comportamento quântico utilizando a biblioteca *PennyLane* para a linguagem *python*. Todos os modelos analisaram janelas de 60, 30 e 15 segundos de tráfego de rede e tiveram resultados excelentes, pontuando $1,000$ em todas as métricas, exceto a primeira variação híbrida, que apresentou um falso negativo em sua classificação. Essa imperfeição sugere que a natureza probabilística da Computação Quântica, além de maior custo computacional e tempo de execução, apresenta incertezas classificatórias. Conclui-se que ambos os modelos são excelentes em classificar tráfego de rede, no entanto o custo computacional e o determinismo do paradigma clássico para Sistemas de Detecção de Intrusão ainda é mais atrativo.

Palavras-chave: Segurança de Redes. Detecção de Anomalias. Detecção de Intrusão. Computação Quântica.

KOMARCHESQUI, M.. **Anomaly detection in computer networks using Quantum Computing**. 2024. 90p. Final Project (Bachelor of Science in Computer Science) – State University of Londrina, Londrina, 2024.

ABSTRACT

Given the widespread application of computer networks and the potential exposure of devices and data, which can serve as a catalyst for attacks, investment in research and development of methods to ensure security in this environment is paramount. Intrusion Detection Systems, as proposed in this work, monitor network traffic and determine in a satisfactory time whether a given data flow is considered normal or anomalous, detecting potential threats. Over the years, various implementations of Network Intrusion Detection Systems have used Machine Learning in the realm of classical computing. Even with complex Deep Learning models, the discussion regarding network intrusion detection remains open. The aim of this work is to study and implement a classical-quantum Hybrid Machine Learning model, demonstrating the peculiarities of this new paradigm and exploring its feasibility in the proposed context. Three variations of the system using Shallow Machine Learning were implemented: a completely classical model and two classical-quantum hybrid models. Both systems were executed on the remote server of the Orion Group, a classical machine capable of simulating quantum behavior using the PennyLane library for the Python language. All models analyzed 60, 30, and 15-second windows of network traffic and achieved excellent results, scoring 1.000 in all metrics, except for the first hybrid variation, which presented a false negative in its classification. This imperfection suggests that the probabilistic nature of Quantum Computing, in addition to greater computational cost and execution time, presents classification uncertainties. It is concluded that both models are excellent at classifying network traffic; however, the computational cost and determinism of the classical paradigm for Intrusion Detection Systems are still more attractive.

Keywords: Network security. Anomaly Detection. Intrusion detection. Quantum Computing.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

2D	Duas Dimensões
3D	Três Dimensões
AI	Artificial Inteligence
APT	Advanced Persistent Threat
AWS	Amazon Web Services
CCNOT	Controlled-Controlled-NOT
CMOS	Complementary Metal-Oxide Semiconductor
CNN	Convolutional Neural Network
CNOT	Controlled-NOT
CSWAP	Controlled-SWAP
DDoS	Distributed Denial of Service
DL	Deep Learning
DNN	Deep Neural Network
DoS	Denial of Service
FN	Falso Negativo
FP	Falso Positivo
GAN	Generative Adversarial Network
GB	Gigabyte
HTTP	Hypertext Transfer Protocol
IBM	International Business Machines Corporation
ICMP	Internet Control Message Protocol
IGTD	Gerador de Imagens para Dados Tabulares
IoT	Internet of Things
IP	Internet Protocol

LSTM	Long Short-Term Memory
LTS	Long-Term Support
MCC	Matthew's Correlation Coefficient
ML	Machine Learning
NIDS	Network Intrusion Detection System
NISQ	Noisy Intermediate-Scale Quantum
NISQ Era	Noisy Intermediate-Scale Quantum Era
NN	Neural Networks
NP	Non-Deterministic Polynomial time
QC	Quantum Computing
QCNN	Quantum Convolutional Neural Network
QNOT	Quantum NOT
QUBIT	Quantum Bit
RAM	Random Access Memory
RSA	Rivest-Shamir-Adleman
SDN	Software Defined Network
SVC	Support Vector Classifier
SVM	Support Vector Machine
SYN	Synchronize
Tbps	Terabits por segundo
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UEL	Universidade Estadual de Londrina
VIME	Valor de Imputação e Estimação de Máscara
VN	Verdadeiro Negativo
VP	Verdadeiro Positivo

SUMÁRIO

1 INTRODUÇÃO

Com a democratização do acesso às redes e aos computadores pessoais, sejam *Desktops*, telefones celulares, *gadgets* inteligentes, entre outros, praticamente todos os serviços do cotidiano traçaram uma relação inextricável com a rede. Seu uso é evidenciado desde a navegação na Internet, envio de mensagens, acesso a entretenimento até ensino a distância, reuniões virtuais e *e-commerce* [2], [3].

Para garantir o funcionamento estável da rede, uma infraestrutura complexa formada por diversos ativos deve ser gerenciada de maneira rápida e confiável [4], [5]. Normalmente estes equipamentos possuem interfaces e fabricantes distintos, o que dificulta sua integração e gerenciamento. Para contornar essa dificuldade, utiliza-se a Rede Definida por Software, do inglês *Software Defined Network* (SDN), uma arquitetura de rede onde definem-se os planos de dados e de controle separadamente. Centraliza-se todo o controle da rede num único dispositivo chamado controlador, programado em alto nível visando definir as políticas e lógica de controle da rede [5]. Os dispositivos físicos do plano de dados (roteadores e *switches*) encarregam-se apenas de encaminhar os pacotes segundo o protocolo *Openflow* e a lógica do controlador [5], [6].

Por conta do grande número de usuários e dispositivos conectados à rede, ataques motivados por diferentes razões podem ocorrer, causando ou tirando proveito de falhas. As vulnerabilidades exploradas comumente têm origem em fraquezas físicas, lacunas de segurança em *software*, ou intervenção direta por agentes humanos [7]. As atividades maliciosas comprometem a integridade, disponibilidade e confidencialidade dos dados em trânsito ou armazenados em rede, potencialmente resultando em perdas substanciais e irreparáveis [8]. Normalmente essas ações deixam evidências discerníveis no fluxo de pacotes de dados [9].

A abordagem que mais tem sido explorada no meio científico para detecção desses rastros anômalos em redes SDN é o Sistema de Detecção de Intrusão de Rede, do inglês *Network Intrusion Detection System* (NIDS) [10], [9], [11]. Um NIDS pode ser implementado de maneira central e observa o tráfego da rede a procura desses indícios, tendo em vista o comportamento normal determinado com base no tráfego histórico. Uma vez encontrados, o sistema alerta os administradores da rede, os quais tomam as medidas cabíveis para mitigar o problema. Esses sistemas são implementados utilizando modelos de Aprendizado de Máquina, podendo variar desde Aprendizado Profundo [12], [13], até algoritmos mais simples denominados de Aprendizado Raso [14], [15].

Mesmo com o uso de abordagens clássicas mais complexas de Aprendizado Profundo como *Generative Adversarial Network* (GAN) [16], *Long Short-Term Memory* (LSTM)

[17] e *Convolutional Neural Network* (CNN) [18], o problema de detecção de intrusão continua uma discussão em aberto considerando o exponencial aumento no volume de tráfego e na velocidade das redes atualmente [9]. Soluções inovadoras como a computação quântica tornam-se alvo de experimentação.

O desejo por uma máquina não-determinística vem desde o século passado. Ainda em 1994, Peter Shor propôs um algoritmo quântico capaz de fatorar números grandes de maneira exponencialmente mais veloz que os computadores tradicionais [19]. Apesar da proposta ser interessante e inovadora, dela falou-se pouco, uma vez que não havia sequer projeto de um computador quântico, muito menos a possibilidade de implementar esse algoritmo [20]. Nos últimos anos, no entanto, a cena da computação quântica mudou. Empresas como *Google*, *IBM* e *Intel* têm investido em larga escala no desenvolvimento desses dispositivos [21], [22], [23], [24].

A promessa da maior capacidade computacional e potencial supremacia quântica sobre o paradigma tradicional tem despertado um crescente interesse na comunidade científica [25], [26]. O tópico foi inundado de questionamentos, desde a sobrevivência dos computadores clássicos como conhecemos até a segurança da criptografia contemporânea. Visando tirar proveito dos benefícios dessa nova abordagem, modelos de Aprendizado de Máquina com aperfeiçoamento quântico, o *Quantum Enhanced Machine Learning*, têm sido empregados para diversas tarefas [27], [28], [29], inclusive a detecção de intrusão [30], [31], [32].

Este trabalho propõe-se a estudar o ambiente da computação quântica, revisando os diferentes modelos de máquinas, o estado da arte dessa nova arquitetura e a base matemática que justifica seu funcionamento. Esse conhecimento é aplicado na implementação de um NIDS quântico, posteriormente comparado com um modelo desenvolvido na computação clássica.

2 FUNDAMENTAÇÃO TEÓRICA-METODOLÓGICA

2.1 Redes Definidas por Software

A rede é a base para a comunicação entre dispositivos. Essa só faz-sa possível com a presença de uma infraestrutura capaz de direcionar os dados de um nó X para um nó Y ou um conjunto de nós Z. Essa infraestrutura é genericamente organizada em três planos de funcionalidade, verticalmente integrados [33].

Os planos de funcionalidade são [33], [34], [35]:

- Plano de Encaminhamento: também chamado de plano de dados, nesse plano ficam localizados os dispositivos de rede responsáveis por encaminhar o tráfego de rede, como roteadores e *switches*. Esses equipamentos seguem regras definidas pelo plano de controle.
- Plano de Controle: define como o tráfego de rede deve ser manipulado. As regras de encaminhamento e lógicas de controle são definidas por esse plano.
- Plano de Gerenciamento: também denominado de plano de aplicação, é o plano onde os gerentes de rede definem as regras que serão empregadas no plano de controle.

A arquitetura convencional das redes de computadores implementa os planos de funcionalidade de maneira colapsada, ou seja, o plano de encaminhamento e de controle ficam dispostos em uma única interface em cada equipamento de maneira individual. Cada dispositivo possui sua interface de programação a qual o gerente de rede irá configurar as lógicas de encaminhamento e políticas [36]. Fabricantes diferentes possuem interfaces de programação distintas, o que aumenta significativamente o custo de administração da rede e inviabiliza um gerenciamento centralizado [35].

Uma *Software Defined Network* (SDN) centraliza o controle da rede em um único ponto, o controlador, separando-o do plano de encaminhamento, delimitado pelos dispositivos que realizam a tarefa de direcionar os pacotes de dados (roteadores e *switches*) [34]. O controlador é responsável por implantar as políticas da rede por meio do protocolo aberto *OpenFlow* pela interface *southbound* [35].

O controlador é delimitado por duas interfaces de programação: *northbound* e *southbound* [34], [35]. A *southbound* é a responsável por providenciar a interface de comunicação entre o plano de controle e o plano de encaminhamento, configurando os dispositivos deste plano. Essa é chamada de cérebro da rede ou sistema operacional da rede. A *northbound*, por sua vez, oferece a interface de programação para o desenvolvimento

de aplicações como monitoramento da rede, gerenciamento de políticas, engenharia de tráfego, aplicação de detecção e mitigação de intrusão, entre outros [34].

O controlador pode assumir duas variações: centralizado e distribuído [37]. No modelo centralizado, todo o controle da rede é concentrado em um único controlador SDN. Isso significa que todas as decisões de roteamento e encaminhamento são tomadas por este único controlador. No modelo distribuído, o controle da rede é distribuído entre vários controladores SDN. Cada controlador é responsável por uma parte da rede, e eles trabalham em conjunto para tomar decisões de roteamento e encaminhamento. Isso significa que não há um único ponto de falha, pois a falha de um controlador não afeta toda a rede. Controladores distribuídos geralmente são mais escaláveis, pois podem lidar com um grande número de dispositivos de rede.

Ambos os modelos têm vantagens e desvantagens. O modelo centralizado oferece uma arquitetura simplificada e eficiente no tratamento de mensagens de requisição, mas pode enfrentar problemas de escalabilidade, especialmente em redes grandes. Os controladores distribuídos lidam melhor com a escalabilidade e podem fornecer maior *throughput* e disponibilidade, mas exigem procedimentos de troca de mensagens adequados entre os controladores no *cluster*.

A Rede Definida por Software é de crucial importância para a gerência de rede uma vez que permite centralizar a lógica de encaminhamento, oferecendo uma abstração da rede para os aplicativos que nela serão implantados, promovendo maior flexibilidade e adaptabilidade da rede às possíveis demandas [34], [36]. Os administradores de rede podem definir políticas de serviço que direcionam o tráfego para funções específicas, como análise de tráfego, otimização de largura de banda e prevenção de ameaças, tudo isso de forma coordenada e automatizada [33], [35]. Para este trabalho, o uso de SDN permite o desenvolvimento de um Sistema de Detecção de Intrusão centralizado que analisa o tráfego da rede e gera alertas aos administradores sobre possíveis ameaças.

2.2 Anomalias de Rede

O fluxo de dados de uma rede é reflexo da interação dos dispositivos sob seu domínio, portanto os pacotes que nela trafegam tendem a seguir um padrão [38], chamado de *baseline* [39]. Essa norma pode ser observada ao coletar dados da rede durante um período de tempo [38], [39], [40]. Aqueles pacotes de dados que destoam do padrão são considerados anômalos [38], [40], [41].

Existem dois grupos abrangentes os quais as anomalias de rede podem ser categorizadas [38]:

- Problemas de Falhas e Performance da rede: falhas de servidores de arquivos, pagi-

nação pela rede, tempestades de *broadcast*, "nós tagarelas", congestão transitória e erros em protocolos.

- Problemas de Segurança: ataques como *Denial of Service* ou intrusões de rede, onde partes cruciais da rede podem acabar desativadas ou usuários legítimos deixados em *starvation*. Um enorme volume de tráfego ocorre nessa categoria de anomalias.

Os ataques de Negação de Serviço Distribuído (DDoS) são uma das principais ameaças enfrentadas pelas infraestruturas de rede em todo o mundo na atualidade [42]. Estudos realizados pela *Arbor Networks* revelam que mais de 2000 ataques DDoS são registrados diariamente em escala global. Esses ataques têm mostrado um aumento significativo, como evidenciado por um aumento de 31% no primeiro trimestre de 2021 em comparação com o mesmo período em 2020, totalizando impressionantes 2.9 milhões de ataques. Exemplos emblemáticos incluem o ataque DDoS massivo direcionado à AWS em 2020, que alcançou uma taxa de tráfego entrante de 2.3 terabits por segundo (Tbps), mantendo as equipes de resposta da empresa ocupadas por cerca de três dias. Em 2021, a exchange de criptomoedas EXMO sofreu um ataque DDoS significativo, resultando na indisponibilidade de seus serviços por quase cinco horas. Outro caso notório foi o ataque DDoS direcionado ao GitHub, uma plataforma de gerenciamento de código online, que atingiu uma taxa de transferência de 1.3 Tbps e 126.9 milhões de pacotes por segundo.

Entre os tipos mais comuns de ataques DDoS utilizados para perturbar os serviços de rede, destacam-se os ataques de inundação de protocolo UDP, SYN, HTTP e ICMP [42]. Os ataques de inundação UDP ocorrem quando os invasores exploram o protocolo de compartilhamento de dados rápido, o Protocolo de Datagrama de Usuário, para enviar pacotes enormes sem a autorização do destinatário. O ataque de inundação SYN é uma forma de ataque baseada na ocupação de servidores, enviando pacotes com endereços IP falsificados para os servidores da vítima, explorando uma falha no protocolo TCP de *three-way handshake*. Já o ataque volumétrico de inundação HTTP visa sobrecarregar uma rede com solicitações HTTP, resultando na negação de serviço a arquivos e solicitações de aplicativos de usuários legítimos assim que a vítima é inundada com solicitações e não consegue responder ao tráfego regular. O ataque de inundação ICMP aproveita o protocolo de mensagens para regular o tráfego de rede, enviando um número excessivo de *pings* para o computador da vítima sem obter resposta, congestionando assim sua capacidade de resposta. Esses tipos de ataques DDoS representam uma ameaça contínua e significativa para a infraestrutura de rede.

É de suma importância que o gerente de rede possa identificar e mitigar problemas de segurança a fim de garantir a integridade, disponibilidade e confidencialidade do que trafega ou é armazenado em rede. Sistemas de Detecção de Intrusão são amplamente implementados em Redes Definidas por *Software*.

2.3 Detecção de Anomalias de Rede

Os Sistemas de Detecção de Intrusão de Rede, *Network Intrusion Detection System* (NIDS), são uma das soluções mais aceitas para a detecção de anomalias de rede no meio científico. Sua função é reportar qualquer tráfego com traços anormais ao administrador de rede [43], funcionando como um complemento para o *firewall*.

Os ataques estão em constante evolução e concepção [44], dificultando sua detecção. Além da agilidade na detecção de uma anomalia [4], o NIDS deve ser capaz de identificar anomalias nunca antes vistas. Esse sistema pode ser implementado de acordo com duas principais estratégias para a detecção [43], [45], [46]. Essas são:

- Baseado em Assinatura: Armazena em um banco de dados os padrões de anomalias já conhecidos e tenta sempre identificar algum desses padrões no tráfego de rede [44]. Devido ao fato de gerar alertas apenas quando uma anomalia for detectada, esse método gera menos falsos positivos [9] e é mais eficiente em identificar anomalias conhecidas. Anomalias não conhecidas não são detectadas [46].

- Baseado em Anomalia: Constrói um modelo que representa o tráfego normal da rede utilizando dados históricos [9]. O NIDS compara de forma constante o comportamento atual do tráfego da rede com o comportamento considerado normal o qual foi treinado. Isso leva à detecção de anomalias sempre que o comportamento atual difere do esperado como padrão. Essa abordagem possibilita a identificação de anomalias que ainda não foram observadas [9], [46].

A abordagem mais utilizada na literatura é a Baseada em Anomalia e é a que será implementada neste trabalho, visto que é mais flexível na detecção de novas ameaças.

Desde 2004, o grupo de pesquisa ORION da Universidade Estadual de Londrina tem contribuído para o estudo e implementação de Sistemas de Detecção de Intrusão eficientes e flexíveis. Diversos trabalhos foram publicados pelo grupo ao longo dos anos e citados por outros autores acerca da segurança em redes, desde uma revisão detalhada da literatura e da fundamentação teórica [9] até variadas abordagens para detecção de anomalias na abordagem computacional clássica [47], [48], [49], [50], [18], [17], [16], [51], [39], [52]. Trabalhos como [53] e [54] contribuem de maneira direta para o gerenciamento da rede, sendo cruciais para a área do conhecimento. A comunidade científica nos últimos anos tem amplamente utilizado modelos de aprendizado de máquina para implementar esses sistemas.

2.4 Aprendizado de Máquina

O Aprendizado de Máquina, *Machine Learning* (ML), é uma subárea da Inteligência Artificial, *Artificial Inteligence* (AI). Essa solução computacional utiliza um conjunto de dados base para treinar um modelo que tem por objetivo ser capaz de fazer previsões sobre os dados que o alimentam [55], distintos dos utilizados no treinamento. O *Machine Learning* vem para suprir a necessidade computacional de automatizar a análise e interpretação de grandes volumes de dados complexos, tornando possível identificar padrões e relações que seriam difíceis ou impossíveis de serem percebidos por meio de métodos tradicionais [56].

Pode-se exemplificar a superioridade do ML por meio da aplicação apresentada no artigo [57]. Nele, é exposto o problema: identificar e classificar vida animal capturada por armadilhas fotográficas. Utilizando um modelo treinado com imagens de vida animal, foi possível automatizar o processo de identificação e classificação de novas fotografias com uma precisão notável, tarefa que seria impossível de ser realizada com algoritmos tradicionais devido à complexidade e variabilidade das imagens.

O aprendizado pode ser classificado em três categorias principais, cujas aplicações estão diretamente relacionadas ao conjunto de dados de treinamento [58]: supervisionado 2.4.1, semi-supervisionado 2.4.2 e não-supervisionado 2.4.3 [56], [59], [60].

2.4.1 Aprendizado Supervisionado

O aprendizado supervisionado utiliza dados onde cada observação X possui um rótulo Y relacionado capaz de descrevê-lo [61]. O objetivo do modelo que utiliza o aprendizado supervisionado é mapear uma observação de entrada X para um Y' satisfatoriamente próximo de Y [56].

Essa abordagem pode ser implementada utilizando diversos algoritmo tanto de classificação quanto de regressão [62]. Neste trabalho um modelo híbrido clássico-quântico supervisionado será estudado e implementado para classificar o tráfego de uma rede de computadores como normal ou anômalo.

2.4.2 Aprendizado Semi-supervisionado

Visto que a etapa de rotulação dos dados é custosa e grande parte das observações não carregam consigo rótulos [56], a aplicação do aprendizado semi-supervisionado pode ser interessante, uma vez que usa dados que contenham rótulos juntamente aos dados que não os contém no processo de treinamento [61].

Uma analogia para a melhor compreensão dessa abordagem é utilizada no livro [56], comparando o aprendizado do ser humano ao aprendizado semi-supervisionado:

> The way we learn is similar to the process of semi-supervised learning. A child is supplied with
>
> 1. Unlabeled data provided by the environment. The surroundings of a child are full of unlabeled data in the beginning.
> 2. Labeled data from the supervisor. For example, a father teaches his children about the names (labels) of objects by pointing toward them and uttering their names.

2.4.3 Aprendizado Não-supervisionado

O aprendizado não-supervisionado faz uso de conjuntos de dados sem rotulação. O objetivo dessa abordagem é encontrar estruturas escondidas nos dados. Uma variedade de motivos podem levar os dados a não possuírem rotulação, desde o alto custo e trabalho agregado à rotulação manual até a natureza intrínseca não-rotulada dos dados [56].

Essa abordagem é amplamente utilizada em sistemas de recomendação [63], que permeiam a internet seja em propagandas baseadas em interesses pessoais, sugestões de produtos em plataformas de comércio eletrônico, ou recomendações de conteúdo em serviços de *streaming*.

2.4.4 Aprendizado Raso

Um sistema de aprendizado é um programa de computador capaz de tomar decisões com base no conhecimento que tem acumulado de casos resolvidos no passado. O Aprendizado Raso é todo sistema que toma decisões com base no conhecimento acumulado de maneira que não busca imitar o raciocínio humano [64]. Na sua forma primordial, o aprendizado de máquina é raso [65]. Diversas técnicas podem ser empregadas para fazer com que uma máquina aprenda, desde métodos matemáticos complexos até busca sistemática de um grande número de possibilidades [64].

Mesmo nos dias atuais, com a existência de modelos de aprendizado de máquina que buscam a cognição humana por meio de neurônios artificiais, o uso de modelos de aprendizado raso se faz presente [65]. Modelos como *Support Vector Machines*, *AdaBoost*, *Random Forest*, entre diversos outros são amplamente utilizados em sistemas comerciais [65] e comparados com modelos *mainstream* complexos e recentes [66], [67].

Este trabalho tem o intuito de estudar e implementar um modelo de *Support Vector Machine* (SVM) híbrido de computação clássica e quântica. Assim como em [67], estudo recente que otimiza um modelo de SVM para a implementação de Sistema de Detecção de Intrusão, este trabalho busca aproveitar o potencial do aprendizado raso aliado à computação quântica para implementar um NIDS.

2.4.4.1 Support Vector Machine

O modelo *Support Vector Machine* (SVM) é um método de aprendizado de máquina raso baseado na teoria de aprendizado estatístico proposto por Vapnik *et al.* na década de 1990 [68], [69], [70]. Esse possui uma base teórica sólida respaldada em princípios matemáticos bem estabelecidos, dessa maneira seu comportamento é compreensível e de certa maneira previsível. O SVM ferece intrinsecamente uma solução esparsa, onde pontos mais importantes são elencados para definir os limites de decisão do modelo, e otimização global, buscando uma solução globalmente ótima [68], [69].

A forma mais básica do SVM é aplicável apenas ao aprendizado supervisionado, no entanto, existem extensões do modelo para trabalhar com os outros tipos de aprendizado [68]. Apesar de ser uma abordagem antiga e mais simples, esse modelo é amplamente utilizado em áreas como categorização de texto, reconhecimento de escrita manual, detecção de faces, análise de dados de expressão genética e várias outras [70], inclusive detecção de anomalias e NIDS [66], [67].

O princípio básico do SVM é encontrar um hiperplano H capaz de separar espacialmente as observações de dados segundo suas classes [68]. Outros dois hiperplanos H_1 e H_2, paralelos a H, conterão os pontos de amostra, pontos esses localizados à "beira"do intervalo de cada uma das classes. Esses pontos são chamados de *support vectors*. A distância entre os hiperplanos H e H_1 ou H e H_2 é chamada de margem. O objetivo do modelo quanto à margem pode ser diferente conforme a separabilidade dos dados e/ou a escolha da função de manipulação da dimensionalidade dos dados [68], [70], [71].

Os dados de entrada do modelo podem ser lineares ou não-lineares. No caso de dados lineares, a dimensionalidade das observações é mantida, uma vez que um hiperplano linear é capaz de realizar a separação dos mesmos [71]. A figura 1 exemplifica o caso de dados lineares e separáveis em um plano bi-dimensional. As classes são representadas pelas cores distintas e os pontos contidos nas retas H_1 e H_2 são os *support vectors* de suas respectivas classes [70], [71].

É possível notar que os pontos azuis e vermelhos estão agrupados de maneira consistente, sem os chamados *outliers* (dados significantemente diferentes dos outros dados da mesma classe). O plano de separação H é capaz de dividir os dados de forma satisfatória e os dados de teste serão, de maneira geral, bem classificados. Nesse caso é feito o uso da margem rígida, *Hard Margin*, onde os vetores de suporte são os pontos absolutamente mais à margem. Essa abordagem é chamada também de *maximal margin*, uma vez que a distância entre H e H_1/H_2 é a máxima possível [70].

Casos como da figura 2, onde existem *outliers*, tornam o uso da *Hard Margin* inviável, uma vez que esses dados muito fora do padrão influenciam no posicionamento do plano de separação H de maneira que novos dados são classificados de forma inconsistente.

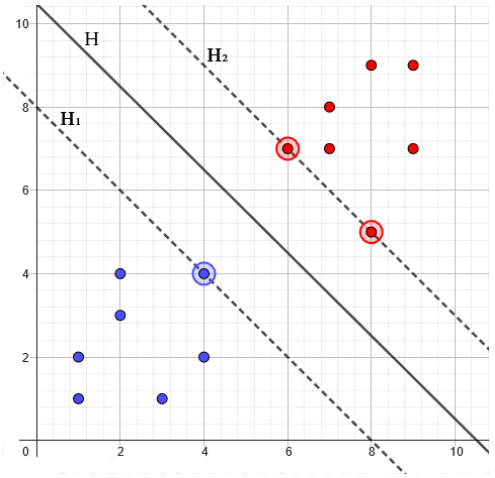

Figura 1 – Dados Lineares e Separáveis usando Hard Margin

Esse caso é denominado de linear e não-separável [70]. Isso pode ser observado no ponto rosa, que é similar aos pontos vermelhos, mas acaba ficando ao lado dos pontos azuis pela separação.

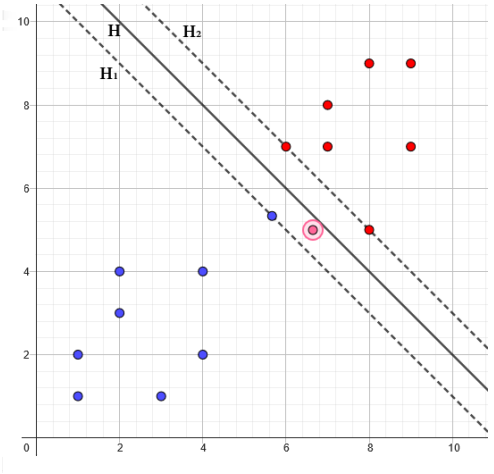

Figura 2 – Dados Lineares e Não-Separáveis usando Hard Margin

Por conta da presença de *outliers*, conceito de *Soft-Margin* faz-se necessário [70]. Essa versão do algoritmo é mais robusta uma vez que tolera algumas classificações incorretas, geradas por ruídos, visando uma melhora global dos resultados do modelo. Ela funciona relaxando a restrição de margem quando necessário [69], [70]. É possível observar uma aplicação de *soft-margin* na figura 3. Essa figura segue a mesma distribuição de dados da figura 2, onde o ponto rosa deve ser classificado juntamente aos pontos vermelhos e um *outlier* azul, destacado na figura, deve ser classificado com os demais pontos azuis. Uma vez que espacialmente esse ponto azul aproxima-se dos pontos vermelhos, considerá-lo rigidamente como *support vector* acarretaria numa piora global dos resultados de classificação, como destacado anteriormente. Tomá-lo como ruído e tolerar a sua classificação incorreta permite a classificação do ponto rosa e demais futuros pontos juntamente aos

vermelhos, como deve ser.

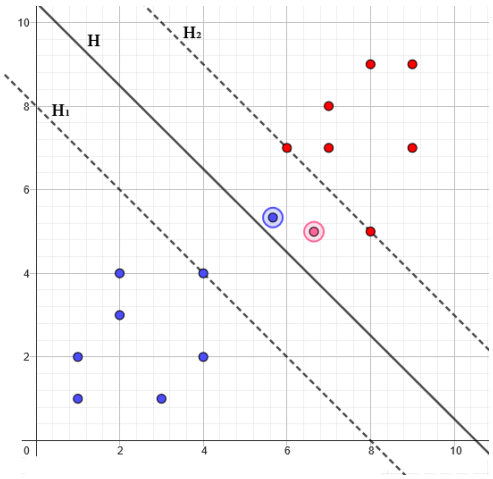

Figura 3 – Dados Lineares e Não-Separáveis usando Soft-Margin

Os dados podem ainda ser não-lineares, como mostrado na figura 4, não sendo possível traçar uma reta que separe os pontos azuis dos vermelhos no plano bi-dimensional nativo dos dados. O modelo de SVM não-linear é responsável em transformar os dados de entrada em dados de maior dimensão num espaço chamado *feature space*, onde é capaz de encontrar uma separação linear dos dados. Esse processo é chamado de *kernel trick* [72], o qual é aplicada uma função de transformação em todos os pontos dos dados, mapeando-os como desejado [70], [71], [73]. No exemplo da figura 5, os dados nativamente apresentados no plano 2D são mapeados segundo uma função ϕ e ficam dispostos no espaço 3D, sendo separáveis linearmente nesse novo espaço.

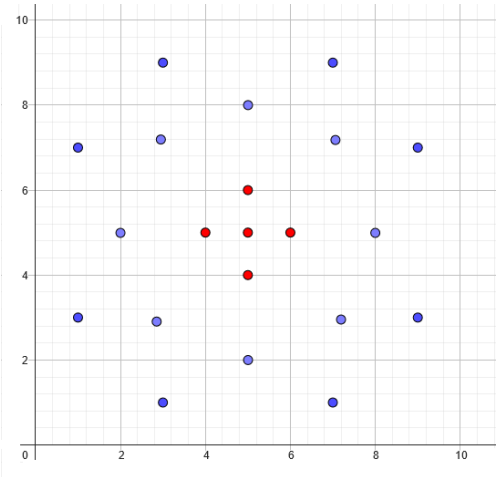

Figura 4 – Dados Não-lineares 2D

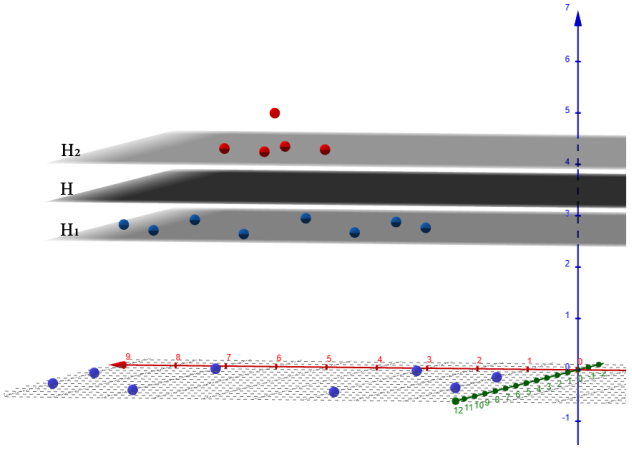

Figura 5 – Dados Mapeados em 3D

A função ϕ, responsável por mapear os dados do espaço original para o *feature space*, é chamada de *kernel function* e é definida de acordo com o problema a ser resolvido [70]. Uma função de *kernel* pode ser representada como uma matriz quadrada $K \in \mathbb{R}^{l \times l}$

tal que $K_{ij} = k(X_i, X_j)$ para um conjunto de vetores $\{X_0, \ldots, X_l\} \subseteq X$ e uma dada função *kernel* k [69], [70], [73].

2.4.4.2 Treinamento e Hiperparâmetros

Como destacado em 2.4.4, um modelo de aprendizado de máquina é capaz de tomar decisões baseando-se em conhecimento acumulado. Com o intuito de proporcionar esse conhecimento à máquina, o modelo deve passar pela fase de treinamento. O treinamento envolve oferecer uma parcela do conjunto de dados ao modelo de maneira que ele possa aprender e, posteriormente, possa ser testado com a parcela restante dos dados. É importante separar corretamente os dados em conjunto de treino e teste, uma vez que testar o modelo com dados que ele já conhece acarreta em respostas "viciadas"e não reflete corretamente o desempenho global do mesmo [71], [73].

Mesmo com a separação correta dos dados, um treino que gere resultados não satisfatório pode ocorrer. Nesse caso, o ajuste dos hiperparâmetros pode ser o detalhe necessário para obter um modelo melhor. Os hiperparâmetros são todos aqueles parâmetros que não são aprendidos pelo modelo durante seu treinamento. Esses são definidos previamente a execução do programa e podem ser ajustados pelo programador por meio de diversas técnicas, como o *Grid Search*, que é uma busca exaustiva pela melhor combinação de hiperparâmetros [72].

O modelo de SVM possui diversos hiperparâmetros, no entanto, grande parte deles é referente à funções de *kernel* clássicas, previamente implementadas, e quanto à margem do hiperplano [72]. Os parametros importantes para este trabalho são:

- Parâmetro de Regularização: Conhecido como parâmetro C, é responsável por controlar o *trade-off* entre maximizar a margem do hiperplano de classificação e minimizar a classificação incorreta. Em linhas gerais, controla o quanto se quer evitar de classificação incorreta.

- Kernel: Define o *kernel* a ser usando pelo modelo. No caso deste trabalho, um *kernel* computado pelo módulo quântico da implementação.

A maneira mais simples e direta de treinar um modelo SVM é com o uso de uma matriz quadrada que relaciona os pontos de dados par a par segundo uma função ϕ [70], [71]. Apesar dessa maneira de computar a função de *kernel* demandar alto uso de memória, ela resulta numa implementação mais explicita do processo matemático e com menos abertura para erros de aproximação ou decomposição [71], [73]. Uma vez que este trabalho mira em implementar um modelo híbrido clássico-quântico, onde o processo de cálculo da função de *kernel* é designada à parcela quântica do modelo, a computação total da matriz de *kernel* é mais adequada e assim será feita.

2.4.5 Aprendizado Profundo

Uma especialização do Aprendizado de Máquina amplamente utilizada é a chamada Rede Neural, *Neural Network* (NN), inspirada no cérebro, utilizando elementos chamados de neurônios e suas conexões para efetuar cálculos complexos [56]. Os neurônios são organizados em camadas sequenciais e adjacentes, o que permite a conexão somente entre neurônios de camadas adjacentes. A primeira e última camada são denominadas respectivamente de camada de entrada e de saída. A primeira é responsável por receber uma entrada X proveniente do conjunto de dados. A última representa a saída Y calculada para a entrada X, seja classificação ou regressão. Entre essas camadas existe pelo menos uma camada oculta. A combinação de número de neurônios por camada e número de camadas ocultas constitui a arquitetura da NN, permitindo inúmeras variações de implementações [62].

O Aprendizado Profundo, *Deep Learning* (DL), é uma especialização mais potente da Rede Neural, sendo denominada Rede Neural Profunda, *Deep Neural Network* (DNN), pois implementa múltiplas camadas ocultas entre a camada de entrada e a de saída [43], [58], [44]. A DNN é capaz de performar modelagens mais complexas que a NN [43], [74], solucionando problemas mais elaborados em detrimento de maior uso do poder computacional. O Aprendizado Profundo requer um número maior de observações para ser treinado em comparação ao Aprendizado de Máquina, utilizando as características dos dados para se auto-ajustar [44].

Uma Rede Neural Profunda pode ser classificada segundo sua forma de aprendizado, sendo [43]:

Discriminativa	Generativa	Híbrida
Aprendizado Supervisionado	Aprendizado Não-supervisionado	Combinação de Ambos

Tabela 1 – Tipos de DNN.

As abordagens de Redes Neurais Profundas para dados tabulares, como os utilizados neste trabalho, podem ser divididas taxonomicamente em três grandes categorias como proposto em [75]: métodos de Transformação de Dados, Arquiteturas Especializadas e modelos de Regularização.

Métodos de Transformação de Dados são essenciais para preparar dados tabulares para modelos de redes neurais profundas, abrangendo tanto codificações unidimensionais quanto multidimensionais. As codificações unidimensionais envolvem técnicas como codificação ordinal ou de rótulo, codificação *one-hot*, codificação *leave-one-out* e codificação baseada em *hash* para transformar variáveis categóricas em representações numéricas adequadas para modelos de aprendizado profundo. As codificações multidimensionais empre-

gam métodos como Valor de Imputação e Estimação de Máscara (VIME), SuperTML e o Gerador de Imagens para Dados Tabulares (IGTD) para criar representações homogêneas dos dados.

Existem dois tipos principais de Arquiteturas Especializadas para redes neurais profundas em dados tabulares: os modelos híbridos, que combinam abordagens clássicas de aprendizado de máquina com redes neurais, e os modelos baseados em transformadores, que utilizam mecanismos de atenção hierárquica.

Os Modelos de Regularização, enfrentam o desafio da não linearidade extrema e complexidade do modelo em dados tabulares. Eles propõem técnicas de regularização forte para destacar a importância das *features* dos dados e melhorar o desempenho. Essas técnicas mostram potencial em superar arquiteturas baseadas em árvores, embora exijam otimização intensiva de hiperparâmetros e tempo computacional substancial.

Um modelo de aprendizado profundo de Arquitetura Especializada chamado de Rede Adversária Generativa, *Generative Advesarial Network* (GAN), proposto em 2014 por Goodfellow *et al.* [76] tem sido amplamente empregado em diversas áreas, sendo considerado o estado da arte do aprendizado de máquina no paradigma da computação clássica. Suas aplicações vão desde a visão computacional, seja convertendo fotografias tiradas durante a noite para simular dia [77], seja gerando imagens realistas de alta qualidade a partir de descrição textual [78], até a detecção de anomalias [79], [80], e a implementação se Sistemas de Detecção de Intrusão.

A Rede Adversária Generativa é uma abordagem de Aprendizado Profundo híbrido, como exposto na tabela 1, baseada na Teoria dos Jogos. Essa é composta por duas Redes Neurais internas que competem entre si, onde uma NN gera dados sintéticos e a outra classifica os seus dados de entrada como sintéticos (gerados) ou orgânicos (advindos do conjunto de dados) [81], [82], [83].

A Rede Neural responsável por sintetizar dados é chamada de gerador e a responsável por distinguir dados reais de dados gerados é chamada de Discriminador. O objetivo do gerador é enganar o Discriminador e do Discriminador discernir o que lhe é alimentado.

Visto que a GAN é uma abordagem generativa, uma de suas principais aplicações é a geração de dados. O treinamento desse modelo tem como objetivo criar um gerador capaz de enganar completamente o discriminador, gerando dados muito similares aos reais [82].

Interagindo diretamente com a interface *northbound* do controlador, o NIDS monitora a rede em busca de vestígios anômalos, alertando os administradores, que tomam as medidas cabíveis para mitigar o problema. A tarefa de perceber tráfego incomum é complexa e melhor resolvida com abordagens de aprendizado computacional. Implementações variam de modelos simples de Aprendizado Raso, como o SVM, a modelos mais sofis-

ticados de Aprendizado Profundo, como GAN. Cada algoritmo tem suas peculiaridades como conjunto de hiperparâmetros, custo computacional, forma de detectar anomalias e estratégia de treinamento. Abordagens de Aprendizado Profundo podem, ainda, ser classificadas em três grandes escopos: Transformação de Dados, Arquitetura Especializada e Regularização. Este trabalho, por sua vez, implementa três variações do modelo SVM, sendo dois aperfeiçoados pela computação quântica. Ambas as implementações têm detecção baseada em anomalias e treinamento supervisionado. Os mesmos hiperparâmetros são utilizados em todas as variações, bem como o conjunto de dados, permitindo uma comparação justa do estado da arte.

3 TRABALHOS RELACIONADOS

A detecção de anomalias usando modelos que tiram proveito da computação quântica é uma abordagem recente e ainda não muito explorada, diferentemente do mundo clássico. Detecção de anomalias de rede, por sua vez, foram ainda menos exploradas usando um modelo híbrido de *Support Vector Machine* e *kernel* gerado por computação quântica.

Rebentrost *et al.* [84], em 2014, demonstra que um *Support Vector Machine* quântico pode ser implementado com um tempo de execução de $O(logNM)$ tanto nas etapas de treinamento quanto de classificação. A eficiência em N é alcançada devido a uma rápida avaliação quântica de produtos internos. Quanto à eficiência em M, o modelo de SVM é reinterpretado como um problema aproximado de mínimos quadrados que permite uma solução quântica com o algoritmo de inversão de matriz. Os autores utilizam uma técnica para a exponenciação de matrizes não esparsas, o que os permite revelar de forma eficiente em forma quântica os maiores autovalores e seus autovetores correspondentes das matrizes de *kernel* e covariância dos dados de treinamento.

Em 2018, Liu *et al.* [85] apresentou em seu trabalho a implementação de um *kernel Principal Component Analysis* quântico para um modelo de SVM a fim de detectar anomalias. Os autores comparam a implementação clássica com sua equivalente quântica e mostram que a versão quântica é capaz de ser executada em tempo logarítmico. Para conjuntos de treino de M estados quânticos, cada um sendo d dimensional, a execução é logarítmica em M e d para estados puros e apenas em d para estados quânticos mistos.

Em 2020, Gouveia *et al.* [86] propôs o uso de um modelo de SVM com *kernel* computado por computador quântico para implementar um Sistema de Detecção de Intrusão. Por meio de *feature map* os autores convertem os dados clássicos para representações dos dados no computador quântico, chamados no artigo de "dados quânticos". Os dados são pré-processados por um algoritmo de *Principal Component Analysis* e têm sua dimensão reduzida usando *Autoencoders*. Esses dados são codificados para os *qubits*, e calcula-se, na parcela quântica do modelo híbrido, a matriz que relaciona todos os dados de treinamento par a par. Os autores concluem que a performance obtida mostra que o modelo híbrido pode ser viável num futuro para a implementação de NIDS, comparado ao SVM clássico.

Em 2022, Hudregtsen *et al.* [87] apresenta em seu artigo o uso de *Trainable Quantum Embedding Kernels* para construir uma matriz de *kernel*. O método proposto utiliza circuitos quânticos parametrizados para criar um mapa de características, *feature map*, quânticas, os quais são otimizados iterativamente para alcançar uma alinhamento desejado com os alvos. Um *Support Vector Machine* é treinado usando a matriz de *kernel*

resultante para classificar novos dados. Para lidar com o ruído inerente aos dispositivos quânticos, são propostas técnicas de mitigação específicas e métodos de regularização, que são avaliados em diversas combinações. No entanto, há desafios remanescentes, como a necessidade de acessar a matriz livre de ruído para avaliar os métodos de mitigação de ruídos, bem como investigar o impacto desses métodos na precisão da classificação.

Compreendendo o paradigma clássico da computação e aplicação que extrapola o ambiente SDN, Gopinath M. e Sethuraman [88] de 2023 aborda de forma ampla e inovadora diferentes abordagens de detecção de *malware* baseadas em *Machine Learning* e *Deep Learning*, especialmente aplicadas em *Sandboxing*, Android, iOS, Windows, IoT, APT e Ransomware. Os pesquisadores destacam que a maioria das ameaças de *software* malicioso visa o sistema operacional Windows, podendo ser *malware* generalizado de automação em massa ou especializado sofisticado. O estudo ressalta que essas ameaças podem ser identificadas de maneira estática, examinando os arquivos executáveis do software suspeito, ou de maneira dinâmica, observando os comportamentos durante a execução do programa malicioso em um ambiente de *sandbox* virtual. Enquanto técnicas de análise estática podem ter dificuldades em lidar com *malware* que utiliza técnicas de ofuscação, elas oferecem uma detecção mais eficaz para famílias de *malware* conhecidas. Assim, uma abordagem dinâmica ou uma combinação híbrida de ambas é preferível para obter resultados mais eficientes na detecção de ameaças.

Apostando em um modelo convolucional, Changqing Gong *et al.* [89] apresenta em seu trabalho de 2024 um Sistema de Detecção de Intrusão com *accuracy* de 94.51%. Sua implementação de QCNN, baseada em *variational quantum neural networks*, é comparada com modelos tradicionais de Aprendizado de Máquina como *Artificial Neural Network*, *Logistic Regression*, *K-nearest neighbor*, SVM e *Decision Tree*, apresentando desempenho superior. Ademais, a implementaão quântica é capaz de efetivamente reduzir a *feature loss* dos dados e melhorar a precisão de deteccão, mostrando-se promissora na na classificação de tráfego.

Alon Kukliansky *et al.* [90] investiga o uso de *Quantum Neural Networks* para detectar intrusões em redes em 2024. Apesar dos desafios apresentados pelas atuais máquinas quânticas de escala intermediária, o desempenho das redes neurais quânticas foi otimizado dentro das limitações existentes. Uma implementação preliminar dessa abordagem alcançou uma pontuação F1 de 0.86, utilizando uma máquina quântica real. Além disso, os autores introduziram uma nova métrica, o fator de certeza, que pode ser empregado para prever a suscetibilidade ao ruído em sistemas de classificação quântica binária. Esse trabalho evidencia o potencial das redes neurais quânticas para a detecção de intrusões, e oferece *insights* na detecção de intrusão quântica.

Dos poucos trabalhos que utilizam o mesmo modelo implementado neste trabalho de conclusão de curso, todos seguem a mesma premissa: extrair o melhor de ambos

os mundos clássico e quântico. O modelo de aprendizado de máquina é um modelo de SVM clássico e roda num computador clássico comum. A parcela que utiliza computação quântica é responsável por calcular o *kernel* que relaciona os dados do conjunto de dados por similaridade. Esse *kernel* é usado pelo SVM para classificar novos dados que venham a ser alimentados ao modelo. De maneira geral, a performance desses modelos híbridos tende a ser mais lenta que dos seus equivalentes puramente clássicos, especialmente modelos híbridos executados com simulação. A promessa é que com o avanço da computação quântica esse desempenho seja acelerado, tornando viável e interessante a implementação dos modelos híbridos.

4 COMPUTAÇÃO QUÂNTICA

Muita incerteza tem permeado o tema "Computação Quântica". Alguns dizem ser o fim da computação como conhecemos, o fim da criptografia, mas ainda estamos muito distantes de substituir os computadores clássicos [25], [20].

Desde 1965, os computadores tiveram sua capacidade de processamento descrita segundo a Lei de Moore [91], a qual afirma que a cada dois anos o poder de processamento de um dispositivo dobra [92]. Essa preditiva que se mostrou válida desde *chips* integrados com 100 transistores até os dias atuais, onde bilhões de transistores compõem um único *chip* [20]. Autores como [93] afirmaram com plena convicção que a evolução tecnológica segue um caminho razoavelmente previsível. Apesar das previsões do eventual fim da Lei de Moore por alguma barreira tecnológica ou científica, os engenheiros e cientistas têm encontrado formas de contornar essas barreiras, prolongando indeterminadamente a vigência dessa Lei.

Essas afirmações contradizem a física moderna, que garante um limite absoluto que a ciência e engenharia consegue atingir. O princípio da Incerteza de Heisenberg postula que há uma limitação da precisão com que podemos medir certas propriedades de partículas subatômicas [94]. O físico Stephen Hawking em 2005, durante sua visita à Intel, afirmou que a velocidade da luz e a natureza atômica da matéria seriam os limites fundamentais para a microeletrônica.

O fim ou não da Lei de Moore é uma discussão em aberto, mas é inegável que a incerteza da sua continuidade foi crucial para direcionar os holofotes em direção à computação quântica. Ainda em 1993, Berstein *et al.* mostrou que os computadores quânticos poderiam violar a Tese Estendida de Church-Turing [95], que afirma que qualquer "modelo razoável"de computação pode ser eficientemente simulado em um modelo padrão, como uma Máquina de Turing, uma Máquina de Acesso Aleatório ou um autômato celular.

Apenas um ano depois, em 1994, Peter Shor mostrou um algoritmo capaz de resolver um problema exponencialmente mais rápido que um computador clássico. O chamado Algoritmo de Shor é um exemplo prático de fatoração de números grandes onde é confirmada a tese apresentada em 1993 por Berstein *et al.* [19].

Por mais que esse algoritmo fosse interessante para o período, nenhum cientista tinha sequer ideia de como construir um computador quântico. Nos dias de hoje não apenas é possível implementar o Algoritmo de Shor[0] , mas também executá-lo em simulações ou computadores quânticos reais. É importante levar em conta que a implementação genérica

[0] O Algoritmo de Shor é um algoritmo de tempo polinomial que usa, como mostrado em [96], $2n + 3$ *qubits* para fatorar um número inteiro de n *bits*.

desse está limitada pela disponibilidade de *qubits*.

4.1 O Paradigma Quântico

Os computadores clássicos possuem circuitos integrados contendo bilhões de transistores que operam sobre os dígitos binários (*bits*), que por sua vez podem assumir valores determinísticos 0 ou 1. Através da manipulação desses bits os dispositivos são capazes de representar informações, operações e, em larga escala, aplicativos e serviços [20].

Um computador quântico também representa informação em uma espécie de *bit*, o *qubit*. O *qubit*, abreviação para *Quantum Bit*, também pode assumir o valor de 0 ou 1, mas não fica limitado a isso. Essa unidade de dados pode ficar em um estado de superposição, onde é 0 e 1 ao mesmo tempo. Um sistema com múltiplos *qubits* pode estar em todos os estados binários possíveis, levando em conta a quantidade de dígitos binários, ao mesmo tempo [97], [98], [99], [20]. Ao longo de um algoritmo quântico, manipulam-se as distribuições probabilísticas desses estados de superposição. Devido à natureza probabilística desses algoritmos, esses são repetidos um número de vezes, a fim de manter um resultado relativamente consistente [29]. Esse fenômeno é conhecido como interferência quântica e é fundamental para o poder de processamento dos computadores quânticos [100], [20].

Os *qubits* são sensíveis a interferências do ambiente, o que pode levar a erros no processamento. Para mitigar esse problema, os computadores quânticos utilizam um processo chamado correção de erros quânticos, onde os *qubits* são redundantes para tentar alcançar uma precisão maior nos cálculos [101]. É natural que um *qubit* decaia ao longo de sua manipulação, principalmente ao ser aplicado em circuitos maiores e mais complexos, perdendo o estado de superposição em que estava, invalidando seu uso [102].

Uma propriedade importante dos *qubits* é a chamada emaranhamento, do inglês *entanglement*. Quando *qubits* estão emaranhados, o estado de um *qubit* não pode ser descrito independentemente do estado de outros *qubits* emaranhados com ele [102], [103]. Essa característica é explorada em algoritmos quânticos num geral, assim como no Algoritmo de Shor [19].

É importante destacar que nem todos os problemas podem ser resolvidos de forma mais eficiente com computadores quânticos. Eles se destacam na resolução de problemas inerentemente quânticos, inteligência artificial e criptografia. Para as demais tarefas esses computadores podem não ser atraentes [20].

Visto que a computação quântica é uma ciência muito recente, até mesmo os últimos *surveys* sobre a tecnologia apresentam informações desatualizadas de seus ambientes de execução e ferramentas. Como destacado em [104], artigo intitulado "QUANTUM COMPUTING: SURVEY AND ANALYSIS"de 2019, a construção de dispositivos reais de computação quântica é um problema fundamental da física moderna e, até então,

existiam apenas implementações limitadas desses aparelhos. Essa limitação persiste até a atualidade, no entanto dispositivos com maior capacidade de *qubits* e correção de erros têm sido construídos.

Além dos recursos de simulação de computadores quânticos, existem dois tipos principais de implementações físicas desses aparelhos: a universal e a não-universal. O computador universal é aquele onde a implementação de algoritmos é livre, possibilitando a execução de operações e resolução de problemas arbitrários. Um dispositivo não-universal é aquele voltado para a resolução duma classe específica de algoritmos, como a otimização de certos modelos de aprendizado de máquina ou otimização combinatória. Um exemplar desse computador pode ser selecionado da coleção de dispositivos desenvolvidos pela empresa canadense D-Wave, como o computador de 16 *qubits* capaz de resolver sudokus. Quando se trata de computadores universais, três grandes empresas se destacam: IBM, Google e Intel.

A IBM conta com *chips* quânticos de acesso gratuito e pago. O acesso gratuito tem *runtime* mensal limitado a 10 minutos e oferece disponibilidade de 7 e 127 *qubits*. Os planos pagos não limitam o tempo de uso e contam com sistemas de 27 e 127 *qubits*. A empresa tem seu próprio *framework* de simulação, cujo uso não é limitado temporalmente. O Qiskit é baseado em Python e traz a mesma programação de alto nível dos algoritmos executados nos *chips* quânticos. A simulação disponibiliza acesso irrestrito aos *qubits* em detrimento de um tempo de execução mais alto [22].

O Google tem uma abordagem mais reservada sobre o acesso aos seus computadores quânticos. Para utilizá-los é necessário escrever uma proposta de pesquisa de 2 a 3 páginas, providenciando informações dos pesquisadores envolvidos, sumário da pesquisa, visão geral do projeto, recursos necessários para execução (tempo de máquina e acesso a *qubits*), experiências passadas com hardware quântico e colaborações para o meio científico e, por fim, o que se espera desse projeto em termos de resultados. Após a submissão da proposta, há uma fase de análise e, caso seja aprovado, o projeto pode ter acesso à *chips* de fila aberta (cuja execução ocorre juntamente a demais projetos aprovados). Conforme a pesquisa avança, acesso dedicado a *chips* pode ser pleiteado [23]. A empresa também aposta na simulação com o *framework* Cirq, de código aberto. O Cirq é uma biblioteca para Python que permite escrever, manipular e otimizar circuitos quânticos [105].

A Intel não disponibiliza acesso aos seus computadores quânticos para o público. A empresa incentiva a pesquisa na área, investindo milhões de dólares em universidades pela Fundação Nacional da Ciência dos Estados Unidos, e oferece vagas de emprego em diversas localidades do mundo [24].

Outros dois grandes *players* que têm apostado na computação quântica são Microsoft e Amazon. Seus *hardwares* quânticos são acessáveis via assinatura, onde o cliente paga conforme os utiliza. Ambos possuem abordagens de simulação, o Q# e o *Braket*

Local Simulator respectivamente [106].

A empresa canadense Xanadu, financiada por diversas outras empresas incluindo a montadora automobilística Porsche, tem por missão construir computadores quânticos acessíveis ao público. Desenvolveram o primeiro computador quântico nanofotônico, que manipula e mede fótons ao longo de seus circuitos, acessível por nuvem do mundo. Para ter acesso a esse *hardware* é necessário entrar em contato com a empresa via *e-mail*. Soluções de *software* também são do escopo da Xanadu, proporcionando a biblioteca *PennyLane* para Python. Essa biblioteca permite a integração de ferramentas de aprendizado de máquina e oferece suporte a um conjunto abrangente de recursos, incluindo as plataformas citadas previamente [107].

A empresa estadunidense Rigetti, premiada como uma das três empresas líderes em computação quântica juntamente ao Google e IBM em 2016, não apenas desenvolve os próprios *chips* (que compõem as máquinas de outras plataformas como as da Microsoft e Amazon) mas também os integra numa arquitetura de controle e, por meio de software proprietário, permite que programadores possam construir algoritmos para rodar nesses dispositivos [108]. Apesar do website da empresa afirmar o acesso completamente gratuito a *hardware* quântico através da plataforma parceira Strangeworks QC, o plano gratuito não contempla tempo de máquina em dispositivos quânticos reais, sendo sujeito a custos não especificados [109].

4.2 Passado, Presente e Futuro

Como visto anteriormente, o estudo da computação quântica é recente, tendo suas primeiras especulações na década de 90. A implementação de *hardware* quântico veio surgir por volta de 2001 com a empresa IBM, onde cientistas da empresa anunciaram teste bem sucedido de um computador quântico de 7 *qubits*. Esse dispositivo fora implementado com base no fenômeno de ressonância magnética nuclear e foi capaz de fatorar o número 15 pelo algoritmo de Shor [104].

De acordo com [104], a próxima implementação quântica veio apenas em 2007, pela empresa D-Wave, com um dispositivo de 16 *qubits*. Esse computador resolvia sudokus e outras tarefas de determinação de padrão. A empresa anunciou ainda o processador One, em maio de 2011, alegando ser o primeiro computador quântico comercial, operando com 128 *qubits*. No ano seguinte, um dispositivo com 512 *qubits* fora anunciado e, em agosto de 2015, um dispositivo com 1152 *qubits*. Em janeiro de 2017, a empresa anunciou a venda do seu futuro dispositivo D-Wave 2000Q, que operava com 2000 *qubits* e custava 15 milhões de dólares, para a empresa de segurança cibernética TDS. Em 2020, a D-Wave anunciou seu computador *Advantage*, com 5000 *qubits* [110].

A maioria dos pesquisadores não consideram os dispositivos desenvolvidos pela D-

Wave como dispositivos quânticos de fato [104]. Visto que sua implementação é adiabática, onde os *qubits* não são manipulados por portas lógicas mas sim por transformações termodinâmicas, alguns algoritmos quânticos podem ser implementados nesse modelo, mas nem sempre são viáveis, favorecendo algoritmos de otimização combinatória [104], [110]. A própria empresa D-Wave enfatizou, após o lançamento do seu modelo 2000Q, que o algoritmo de Shor não poderia ser implementado em seus dispositivos [104]. Em 2015 pesquisadores do Google alegaram que os dispositivos da D-Wave, apesar de apresentarem os fenômenos quânticos esperados, agrupavam os *qubits* em clusteres menores. As características desses computadores foram analisadas por Cho [111] e foi concluído que esses não seriam capazes de prover nenhuma vantagem computacional sobre os computadores clássicos.

Considerando dispositivos quânticos baseados em portas lógicas, no ano de 2007, um grupo de cientistas da Universidade de Queensland reportaram a fatoração bem sucedida do número 15 usando 7 *qubits*, num computador cujas portas lógicas eram baseadas em polarização de fótons. No mesmo ano, cientistas da Universidade de Ciência e Tecnologias da China também reportou a fatoração bem sucedida do número 15 usando apenas 6 *qubits*, numa implementação do algoritmo de Shor com portas lógicas baseadas em fótons [104].

Em 2009, a fatoração do número 15 por meio do algoritmo de Shor pôde ser implementada em um chip de silício usando 4 *qubits*. No ano de 2012, pesquisadores da Universidade da Califórnia reportaram a demonstração do algoritmo de Shor implementado com 4 *qubits* de fase (tipo específico de *qubit*) e ressonadores de ondas supercondutoras para criar e manipular os circuitos, fatorando o número 15. Nesse mesmo ano, E. Martin-Lopez *et al.* fatorou o número 21 usando uma implementação do mesmo algoritmo com apenas 2 *qubits* baseados em fótons [104].

Transformando o algoritmo de Shor em um algoritmo de otimização combinatória, em 2012, foi possível fatorar o número 143 usando 4 *qubits* na implementação adiabática. Em 2015, um grupo de pesquisadores liderados por T. Montz reportou uma nova demonstração experimental do algoritmo de Shor usando armadilha de íons para decompor o número 15. Nesse experimento, 5 íons $^{40}Ca^{+}$ numa armadilha linear de Paul foram usados, sendo cada íon traduzido para um *qubit* [104].

Em 2016, a IBM anunciou a criação de um computador quântico de 5 *qubits*, sendo um deles usado para correção de erros. Esse dispositivo é baseado num *chip* supercondutor com geometria estrela e implementação completa da álgebra de Clifford, sendo programável e permitindo criar portas lógicas, bem como modelar a operação das mesmas. Em maio de 2017, a empresa anunciou um novo dispositivo de 50 *qubits*, onde 20 *qubits* são usáveis e o restante é destinado à correção de erros, já que um *qubit* nesse computador fica em estado coerente por até 90 microssegundos, tornando-se incoerente após esse período

[104]. De acordo com o site da empresa [21], foram lançados em 2020 o *chip* Hummingbird de 65 *qubits*, em 2021 o Eagle de 127 *qubits*, 2022 o Osprey com 433 *qubits* e, por fim, o chip Condor com 1121 *qubits* em 2023.

Em janeiro de 2018, a Intel também se juntou à corrida dos computadores quânticos e declarou a criação de um *chip* quântico supercondutor de 49 *qubits*, chamado de Tangle Lake [104]. De acordo com o site da empresa [24], o *chip* Tunnel Falls é seu último lançamento. O Tunnel Falls é um *chip* de pesquisa e é baseado em *qubits* de *spin* de silício. Sua criação é, de acordo com a Intel, um progresso significante na criação de um *chip* quântico confiável e produzível em escala, uma vez que utiliza a tecnologia de transistores CMOS de silício. Esse chip possui 12 *qubits*.

Em março de 2018, o Google apresentou o *chip* Bristlecone com capacidade para 72 *qubits*, que aproveitava a construção de um *chip* de 9 *qubits* da própria empresa. O diferencial em relação ao modelo de 9 *qubits* é o uso de estruturas bi-dimensionais de $6x6$ *qubits* empilhadas verticalmente. De acordo com a empresa, essa construção permite que o dispositivo localize e corrija erros em tempo de execução [104]. De acordo com o site do Google [112], seu *chip* mais recente é o Sycamore, com 54 *qubits*. A empresa alega que esse *chip* pode executar uma computação de teste em 200 segundos, que no supercomputador clássico mais rápido do mundo demoraria 10000 anos.

Atualmente estamos na denominada *Noisy Intermediate-Scale Quantum Era* (NISQ Era), termo usado para expressar a situação atual dos computadores quânticos, de Escala Intermediária e com Ruídos [26], [25], [100]. A dita "vantagem quântica"só pode ser atingida em sua completude com alta disponibilidade de *qubits* e menor taxa de erros devido à decaimentos [25], sendo assim, a tecnologia que temos até então não é matura o suficiente.

A pesquisa e estudo na área é promissora e o futuro aguarda implementações mais estáveis e poderosas, como destaca o autor John Preskill, em 2018, sobre a então futura computação quântica [26]:

> NISQ devices will be useful tools for exploring many-body quantum physics, and may have other useful applications, but the 100-qubit quantum computer will not change the world right away — we should regard it as a significant step toward the more powerful quantum technologies of the future. Quantum technologists should continue to strive for more accurate quantum gates and, eventually, fully fault-tolerant quantum computing.

As empresas de tecnologia têm buscado a criação de *chips* quânticos proprietários e universais, devido à promissora perspectiva da computação quântica. A atualidade está limitada à poucos *qubits* utilizáveis. Por mais que diversos *chips* possuam uma porcenta-

gem grande dos seus *qubits* designados para a correção de erros, ainda é cedo para declarar em qualquer instância que a vantagem quântica já foi alcançada [104].

4.3 Arquitetura do Computador Quântico

Como destacado em 4.1, diversas implementações de computadores quânticos têm sido executadas por diferentes empresas. Diferentemente dos computadores clássicos, onde uma única abordagem é utilizada para a manufatura desses dispositivos, os computadores quânticos têm diferentes modelos teóricos que embasam suas implementações. Os modelos teóricos podem ser resumidos em *gate model*, *measurement model*, *adiabatic model*, *quantum annealing* e *topological quantum computing*, sendo que o último ainda não conta com exemplares físicos.

O chamado *gate model*, também conhecido como *circuit model*, é um modelo que se assemelha ao máximo com a computação clássica como conhecemos. Como o próprio nome diz, esse modelo teórico de computador quântico é baseado em circuitos e portas lógicas. Um circuito aplica portas lógicas em seus *qubits* a fim de manipular seus estados de superposição. Um algoritmo é um circuito de X *qubits* manipulados por operações arbitrárias e medidos ao final desse circuito. Vale destacar que uma extensão desse modelo permite a medição dos *qubits* em qualquer ponto de um circuito, não apenas ao final, sendo nomeada de "circuitos quânticos com medições intermediárias"[113].

O modelo *measurement model* é um caso especial dos circuitos com medições intermediárias. A principal diferença e conceito central desse modelo é que os *qubits* do circuito são inicializados com um estado emaranhado (*entangled*, como explicado em 4.1) e são medidos individualmente ao longo desse circuito. Esse modelo é matematicamente equivalente ao modelo *gate model*, e é capaz de simular diversos circuitos subsequentes em um grande circuito contínuo [113].

É no modelo adiabático que os conceitos da computação clássica começam a dar lugar aos conceitos da física pura e simples. No *adiabatic model* não existem circuitos nem portas lógicas, ao invés disso, esse modelo tira proveito do principio geral da física que postula que todo sistema sempre se move em direção ao estado de menor energia. O problema que se deseja resolver por um computador quântico adiabático deve ser construído de tal maneira que o estado de menor energia represente a resposta para esse problema. Um sistema quântico adiabático é iniciado com uma paisagem energética plana e é manipulada de forma que gradualmente são introduzidos os relevos energéticos do problema. Cada ponto na paisagem energética é um potencial *output* para o computador quântico e o ponto de menor energia representa a resposta ao problema. A chave para esse modelo funcionar é a lenta introdução da paisagem energética do problema no sistema [114].

Quantum annealing funciona da mesma maneira que o modelo adiabático, no en-

tanto é direcionado para a solução de problemas específicos e não oferece o mesmo grau de liberdade para a implementação de um algoritmo arbitrário, sendo considerado um esquema não-universal [115].

Os modelos baseados em circuitos e o modelo adiabático são modelos universais de computação quântica, sendo assim são capazes de implementar qualquer sistema quântico, e considerados equivalentes em termos de escopo de solução de problemas, visto que problemas podem ser modelados para ambos os esquemas [116].

O modelo topológico de computação quântica é o mais teórico dos modelos existentes. Seus *qubits* são construídos a partir de uma entidade física chamada *majorana zero-mode quasi-particle*, um tipo de ânyon não-abeliano. Os ânyons são um tipo especial de partícula, denominada quasipartícula, não fundamental e bidimensional. Ou seja, não são partículas como átomos, elétrons ou fótons, são criadas através do comportamento de diversas partículas juntas e acabam tendo propriedades de partícula, apesar de não serem reais. O exemplo mais simples de quasipartícula é o buraco de elétron, onde uma grade de elétrons têm um buraco e esse buraco acaba se comportando como uma partícula, mesmo sendo apenas a ausência de um elétron. A quasipartícula *majorana zero-mode quasi-particle* foi teoricamente prevista e observada. Seu uso na computação quântica é promissor uma vez que é muito mais estável que os demais modelos de *qubits*, já que é formada por partes fisicamente separadas por um vão energético, o que as torna muito mais resistentes à ruído [117].

Devido à maior facilidade em discernir problemas para o modelo de circuitos, além do maior acesso à computadores e softwares de simulação do tipo *circuit model*, esse trabalho utilizará esse modelo para a implementação e futura revisão bibliográfica.

4.3.1 Qubit

O *qubit* foi apenas apresentado como uma unidade similar ao *bit* do computador clássico, porém capaz de estar num estado de superposição onde não é nem deterministicamente 0 nem 1, mas sim duas probabilidades distintas de estar em cada estado base. É apenas quando observamos o *qubit* que ele assume um valor determinístico de 0 ou 1 [97].

Uma partícula em superposição é representada por um estado quântico na mecânica quântica. Esse estado é a combinação linear de todos os seus possíveis resultados e suas respectivas probabilidades de ocorrência. Algumas partículas podem assumir infinitos possíveis resultados e seu estado quântico resultante seria uma combinação linear de infinitos termos [97], [118]. Outras partículas, como o átomo de hidrogênio, têm energias discretas e enumeráveis, sendo possível listar todas as possíveis medições segundo a formula de Rydberg. [118].

O estudo da física quântica deve contemplar tanto partículas com infinitos possí-

veis estados, como as de estados enumeráveis, garantindo universalidade. Trabalhar com infinidade é um tanto complicado, uma vez que o infinito não é uma grandeza ou um número de fato, mas sim um conceito matemático. É natural que uma sequência convergente de infinitos termos seja resolvida usando limite. Usar limite, no entanto, pode gerar um vetor resultante que não pertence ao espaço vetorial da sequência original. Usa-se, então, um espaço vetorial especial chamado espaço de Hilbert ($\mathcal{H}$) para trabalhar com estados quânticos, consequentemente *qubits* [118].

Um espaço de Hilbert é todo e qualquer espaço vetorial com a propriedade de produto escalar Cauchy completo, ou seja, toda sequência convergente de vetores converge para um elemento dentro desse mesmo espaço vetorial. Isso garante que, mesmo com N termos que se aproximam de infinito, o limite nunca será atingido e o vetor resultante dessa combinação linear permaneça no mesmo espaço vetorial [118].

Nesse estudo, a norma do vetor não importa. Dois vetores $\mathbf{a}$ e $\mathbf{b}$ diferenciados apenas pelo seu escalar complexo são estados quânticos $\boldsymbol{\alpha}$ e $\boldsymbol{\beta}$ equivalentes.

$$\alpha\mathbf{a} = \mathbf{b} \Rightarrow \boldsymbol{\alpha} \sim \boldsymbol{\beta}.$$

Os estados quânticos são tratados como "raios"que passam pela origem de $\mathcal{H}$. Uma representação mais amigável desses estados quânticos é a *complex projective sphere*, onde toda a coleção de raios é representada por raios equivalentes de comprimento unitário, formando uma superfície esférica em torno da origem de $\mathcal{H}$. Os pontos nessa superfície representam também os estados quânticos do *qubit*. A *complex projective sphere* não é um espaço vetorial, é apenas uma abstração do espaço que leva em contas vetores normalizados [118].

O estado quântico de um *qubit* é representado pelo espaço vetorial de Hilbert, onde $\mathcal{H} = \mathbb{C}^2$. O espaço é complexo bi-dimensional uma vez que o estado de um *qubit* é medido segundo o spin da partícula que o implementa fisicamente. Caso fosse optado por medir a posição ou o momento angular dessa partícula, o espaço $\mathcal{H}$ seria de dimensão infinita, dificultando a manipulação de informações [118].

Mas o que é o spin? O spin é uma propriedade intrínseca de partículas com carga, como um elétron, e não, como se acreditava quando fora descoberto, um movimento da partícula de girar em torno de um eixo. Definir o spin de maneira criteriosa demanda um conhecimento mais profundo de mecânica quântica, impossível de ser abordado em um trabalho de conclusão de curso. Para este escopo, basta compreender que o spin de um elétron (ou outra partícula subatômica) é uma propriedade de todas as partículas subatômicas que se assemelha a um imã. É como se cada elétron tivesse seu próprio campo magnético e agisse como uma pequena barra imantada, o motivo disso ocorrer não nos interessa [118]. Pode-se intuir um elétron como na figura abaixo.

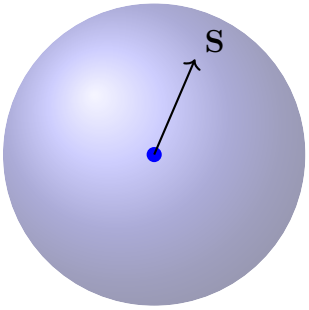

Figura 6 – Representação de um elétron e seu Spin.

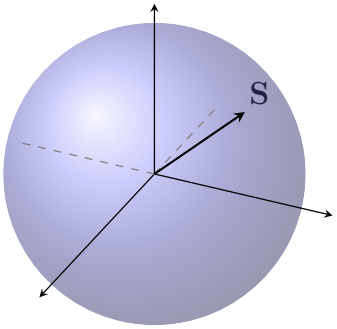

Figura 7 – Representação do Spin como um vetor no plano 3D.

O eixo do spin tem direção e magnitude, tal qual um vetor. Define-se sua magnitude como

$$S \equiv |\mathbf{S}| = \sqrt{S_x^2 + S_y^2 + S_z^2} \tag{4.1}$$

e direção na forma de vetor unitário

$$\hat{\mathbf{n}}_S = \hat{\mathbf{S}} \equiv \frac{\mathbf{S}}{|\mathbf{S}|}.$$

A magnitude do spin de um elétron é o valor constante $\frac{\sqrt{3}}{2}\hbar$, onde $\hbar$ é a constante de Plank. Portanto, tem-se

$$\mathbf{S} = \left(\frac{\sqrt{3}}{2}\hbar\right)\hat{\mathbf{n}}_S.$$

Restando apenas a direção do spin como única entidade que pode ter o valor modificado, sendo essa entidade a base para a modelagem do *qubit* [118].

A direção do spin definida pelo vetor $\mathbf{S}$ pode ser projetada nos eixos x, y e z, gerando as componentes S_x, S_y e S_z respectivamente, como visto em 4.1. Medir $\mathbf{S}$ não é tarefa trivial, sendo possível medir apenas uma componente por vez. Ao medir a componente S_z, por exemplo, é possível obter apenas os valores $+\frac{\hbar}{2}$ ou $-\frac{\hbar}{2}$. Mesmo com uma grande gama de possíveis valores para o spin, ao medi-lo, o mesmo colapsa em apenas

um dos dois valores descritos anteriormente. Essa propriedade binária é essencial para a computação quântica [118].

O estado $+\frac{\hbar}{2}$ projetado sobre o eixo z é nomeado $|+\rangle_z$ (*spin-up*) e o estado $-\frac{\hbar}{2}$ nomeado $|-\rangle_z$ (*spin-down*) [119]. Servindo de estados 0 e 1 para o nosso sistema quântico do ponto de vista computacional. Projetando o spin sobre os demais eixos, tem-se:

$$\begin{aligned} |+\rangle_z &\leftrightarrow |0\rangle_z & |-\rangle_z &\leftrightarrow |1\rangle_z \\ |+\rangle_x &\leftrightarrow |0\rangle_x & |-\rangle_x &\leftrightarrow |1\rangle_x \\ |+\rangle_y &\leftrightarrow |0\rangle_y & |-\rangle_y &\leftrightarrow |1\rangle_y . \end{aligned}$$

Para finalidade de notação,

$$\begin{aligned} |+\rangle_z &\leftrightarrow |0\rangle & |-\rangle_z &\leftrightarrow |1\rangle \\ |0\rangle_x &\leftrightarrow |+\rangle & |1\rangle_x &\leftrightarrow |-\rangle \\ |+\rangle_y &\leftrightarrow |0\rangle_y & |-\rangle_y &\leftrightarrow |1\rangle_y . \end{aligned}$$

Mantendo os vetores da componente y explicitamente enunciados.

Pode-se definir a base natural do espaço vetorial de Hilbert sobre uma componente genérica que descreve o *qubit*:

$$\begin{aligned} \mathcal{H} \equiv \left\{ \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \mid \alpha, \beta \in \mathbb{C} \right\} & \qquad com \\ |+\rangle \leftrightarrow \begin{pmatrix} 1 \\ 0 \end{pmatrix} & \qquad e \\ |-\rangle \leftrightarrow \begin{pmatrix} 0 \\ 1 \end{pmatrix} & \qquad . \end{aligned}$$

Logo, um *qubit* genérico $|\psi\rangle$ é descrito como a combinação linear

$$|\psi\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \alpha\,|+\rangle + \beta\,|-\rangle , \qquad onde\ |\alpha|^2 + |\beta|^2 = 1.$$

Justificando aqui a bi-dimensionalidade do espaço $\mathcal{H}$ e o uso da *projective sphere*. Uma vez que a soma de todas as probabilidades deve ser 1 (não faz sentido trabalhar com probabilidades somadas que passem de 100%), a soma do quadrado das amplitudes deve ser 1. Logo, o vetor que representa o estado quântico descrito é unitário e pertence à esfera de projeção [119].

Finalmente, tem-se a relação entre o bit clássico e o bit quântico:

$$\begin{aligned} Clássico: & \qquad \mathbf{x} = \alpha[0] + \beta[1], & \alpha^2 \oplus \beta^2 = 1 \\ Quântico: & \qquad |\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle , & |\alpha|^2 + |\beta|^2 = 1. \end{aligned}$$

4.3.2 Portas Lógicas Unárias

O circuito quântico, ao executar, deve realizar transformações nos *qubits* que armazenam as informações, por meio de operadores quânticos. Um Operador Lógico Quântico Unário, ou "Porta Lógica Unária", é definido como uma transformação linear em $\mathcal{H}$ que mapeia vetores normalizados (unitários) para outros vetores normalizados.

Um operador unário pode ser representado no formato de matriz $2x2$, já que o espaço $\mathcal{H}$ é bi-dimensional

$$\begin{pmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{pmatrix}.$$

Cada componente da matriz é um número complexo [120], [121], [122].

4.3.2.1 Quantum NOT, X

A porta lógica QNOT, ou Quantum NOT, inverte a amplitude de qualquer vetor de estado da base z ($|0\rangle_z$ e $|1\rangle_z$) e pode ser representado pelo símbolo X [120], [121], [122]. Esse operador corresponde à matriz

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

Aplicando o QNOT à um *qubit* genérico $|\psi\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$, obtém-se:

$$X|\psi\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \beta \\ \alpha \end{pmatrix}.$$

4.3.2.2 The Phase Flip, Z

Diferente do QNOT, que realiza a inversão das amplitudes, o operador Z realiza a máxima inversão relativa entre as amplitudes α e β do vetor de estado. Sua matriz é definida como

$$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix},$$

negando a segunda amplitude do vetor de estado

$$Z|\psi\rangle = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \alpha \\ -\beta \end{pmatrix}.$$

Esse resultado é de fato a máxima inversão uma vez que

$$-1 = e^{i\pi},$$

portanto multiplicar β por esse escalar causa uma rotação de π (180^o) dessa amplitude sobre o plano dos complexos [122].

A aplicação do operador Z não tem uma tradução para a lógica clássica e seus efeitos em um único *qubit* são nulos, uma vez que $|\alpha|^2$ e $|\beta|^2$ permanecem os mesmos [121]. Por mais que os estados tenham as mesmas probabilidades de medição não necessariamente são o mesmo estado. A diferença do operador Z pode ser percebida ao incorporá-lo numa expressão maior onde a tira-se proveito da superposição de mais estados [120], [121], [122].

4.3.2.3 O Operador Y

Esse operador é definido pela matriz

$$\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix},$$

gerando o seguinte efeito num *ket* genérico $|\psi\rangle$:

$$Y|\psi\rangle = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = -i \begin{pmatrix} \beta \\ -\alpha \end{pmatrix} = \begin{pmatrix} -i\beta \\ i\alpha \end{pmatrix} \cong \begin{pmatrix} \beta \\ -\alpha \end{pmatrix}.$$

Como é possível observar pela equação acima, esse operador realiza tanto o *bit flip*, a invertendo as amplitudes α e β em relação aos seus respectivos *kets*, quanto o *phase flip*, invertendo maximamente as amplitudes entre si. Essa transformação causada pelo operador gera uma medição do *qubit* igual à medição após a aplicação de uma QNOT [120], [121].

4.3.2.4 A Porta Hadamard, H

De acordo com [120], [121], [122], o operador Hadamard é definido pela matriz

$$\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}.$$

Ao aplicá-lo para as bases $|0\rangle$ e $|1\rangle$, nota-se o seguinte comportamento:

$$H|0\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle \quad e$$

$$H|1\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle.$$

Olhando mais atentamente, é possível notar que as probabilidades de ocorrência dos *kets* $|0\rangle$ e $|1\rangle$ tornam-se $\frac{1}{2}$, uma vez que $|\frac{1}{\sqrt{2}}|^2 = \frac{1}{2}$. Sobre a componente observável

S_z, base z, as amplitudes igualam-se, "rotacionando"o vetor de estado da base z para a base x, sendo

$$H|0\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle \equiv |+\rangle \quad e$$

$$H|1\rangle = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle \equiv |-\rangle .$$

Para um estado genérico $|\psi\rangle$ temos:

$$H|\psi\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\begin{pmatrix}\alpha\\ \beta\end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix}\alpha+\beta\\ \alpha-\beta\end{pmatrix} = \left(\frac{\alpha+\beta}{\sqrt{2}}\right)|0\rangle + \left(\frac{\alpha-\beta}{\sqrt{2}}\right)|1\rangle .$$

4.3.2.5 Portas Phase-Shift, S, T e R_θ

Os operadores de *Phase-Shift* são todos aqueles que realizam rotações do coeficiente do $|1\rangle$ em relação ao observável S_z em θ radianos. O operador *Phase-Flip* (Z) já visto é um caso particular de *Phase-Shift* [121], [122].

O operador R_θ realiza a rotação em qualquer ângulo θ desejado. Os operadores S e T são outros casos especiais com ângulos que são comumente usados, $\frac{\pi}{2}$ e $\frac{\pi}{4}$ respectivamente. A matriz que define R_θ é

$$\begin{pmatrix}1 & 0\\ 0 & e^{i\theta}\end{pmatrix},$$

onde θ é um número real qualquer. Sua aplicação num estado genérico $|\psi\rangle$ é:

$$R_\theta|\psi\rangle = \begin{pmatrix}1 & 0\\ 0 & e^{i\theta}\end{pmatrix}\begin{pmatrix}\alpha\\ \beta\end{pmatrix} = \begin{pmatrix}\alpha\\ e^{i\theta}\beta\end{pmatrix}.$$

Os operadores S e T são definidos em termos de R_θ,

$$S \equiv R_{\pi/2} = \begin{pmatrix}1 & 0\\ 0 & i\end{pmatrix},$$

e

$$T \equiv R_{\pi/4} = \begin{pmatrix}1 & 0\\ 0 & e^{i\pi/4}\end{pmatrix}.$$

4.3.3 Portas Lógicas Binárias

Diferentemente dos sistemas de um *qubit*, sistemas de mais *qubits* não podem residir no espaço $\mathcal{H}$ bi-dimensional. Isso se dá uma vez que *qubits* podem estar emaranhados, formando um estado único que reside num espaço vetorial maior. Esse crescimento é

exponencial, sendo 1 *qubit* pertencente à um $\mathcal{H}^2$, 2 *qubits* $\mathcal{H}^4$, 3 *qubits* $\mathcal{H}^8$ e assim por diante [120], [122].

Para alcançar esse comportamento usamos produto tensorial. Para dois *qubits* genéricos $|a\rangle$ e $|b\rangle$, tem-se o produto tensorial $|a\rangle \otimes |b\rangle$:

$$\begin{pmatrix} a_0 \\ a_1 \end{pmatrix} \otimes \begin{pmatrix} b_0 \\ b_1 \end{pmatrix} = \begin{pmatrix} a_0 \begin{pmatrix} b_0 \\ b_1 \end{pmatrix} \\ a_1 \begin{pmatrix} b_0 \\ b_1 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} a_0 b_0 \\ a_0 b_1 \\ a_1 b_0 \\ a_1 b_1 \end{pmatrix}.$$

Para os *kets* base do $\mathcal{H}^2$, tem-se:

$$|0\rangle \otimes |0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix},$$

$$|0\rangle \otimes |1\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix},$$

$$|1\rangle \otimes |0\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \quad e$$

$$|1\rangle \otimes |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}.$$

A fim de encurtar a notação, os estados descritos acima são denominados $|00\rangle$, $|01\rangle$, $|10\rangle$ e $|11\rangle$ respectivamente. Esses estados formam a base do espaço $\mathcal{H}^4$ e um estado genérico $|\psi\rangle^2$ que reside nesse espaço vetorial é representado da seguinte maneira:

$$|\psi\rangle^2 = \alpha\,|00\rangle + \beta\,|01\rangle + \gamma\,|10\rangle + \delta\,|11\rangle\,.$$

Combinar mais *qubits* é realizar o produto tensorial de mais vetores de estado, gerando estados que residem em espaços vetoriais de maior dimensão [120], [122].

Portas lógicas de dois *qubits* são representadas por matrizes $4x4$

$$\begin{pmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{pmatrix}.$$

4.3.3.1 Porta Controlled-NOT, CNOT

O intuito desta porta lógica é utilizar um *qubit* de controle para negar outro *qubit* com base em seu valor. Considerando um *qubit* $|x\rangle$ como o de controle e um $|y\rangle$ como o alvo, temos para a base binária e sobre a componente observável S_z:

$$|y\rangle \mapsto \begin{cases} |y\rangle\,, & se \quad x = 0 \\ |\neg\, y\rangle\,, & se \quad x = 1 \end{cases}.$$

Computa-se a matriz dessa porta lógica por:

$$\begin{aligned} M_{CNOT} &= (CNOT\,|00\rangle\,,\ CNOT\,|01\rangle\,,\ CNOT\,|10\rangle\,,\ CNOT\,|11\rangle) \\ &= (|00\rangle\,,\ |01\rangle\,,\ |11\rangle\,,\ |10\rangle) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}. \end{aligned}$$

Aplicando em um estado genérico $|\psi\rangle^2$, tem-se:

$$\begin{aligned} CNOT\,|\psi\rangle^2 &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \\ \gamma \\ \delta \end{pmatrix} = \begin{pmatrix} \alpha \\ \beta \\ \delta \\ \gamma \end{pmatrix} \\ &= \alpha\,|00\rangle\ + \beta\,|01\rangle\ + \delta\,|10\rangle\ + \gamma\,|11\rangle\ . \end{aligned}$$

Estados separáveis que entram na porta CNOT não necessariamente saem separáveis. Para demonstrar esse comportamento, considere o estado

$$|\psi\rangle^2 = |0\rangle_x \otimes |0\rangle = |+\rangle \otimes |0\rangle = \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right)|0\rangle$$

como *input* da CNOT. Sendo $\left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right)$ o controle e $|0\rangle$ o alvo. Os estados separáveis devem ser resolvidos na base tensorial,

$$\left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right)|0\rangle = \frac{1}{\sqrt{2}}\,|00\rangle + \frac{1}{\sqrt{2}}\,|10\rangle\,.$$

Aplicando a porta CNOT usando linearidade,

$$\begin{aligned} CNOT(\ \frac{1}{\sqrt{2}}\ |00\rangle + \frac{1}{\sqrt{2}}\ |10\rangle\ ) &= \frac{1}{\sqrt{2}}\,CNOT(\ |00\rangle\ )\ +\ \frac{1}{\sqrt{2}}\,CNOT(\ |10\rangle\ ) \\ &= \frac{1}{\sqrt{2}}\ |00\rangle\ +\ \frac{1}{\sqrt{2}}\ |11\rangle \\ &= \frac{|00\rangle\ +\ |11\rangle}{\sqrt{2}}. \end{aligned}$$

O *output* $\frac{|00\rangle + |11\rangle}{\sqrt{2}}$ representa um estado de emaranhamento dos *qubits* de controle e alvo. O emaranhamento não impede de medir um *qubit* separadamente do outro, de fato eles são entidades separadas no espaço. Ao medir uma das duas entidades, devido à sua relação de emaranhamento, ambos os *qubits* colapsam para seus respectivos resultados.

Um exemplo de separabilidade no *output* é dada pelo *input*

$$|\psi\rangle^2 = |1\rangle \otimes |0\rangle_x = |1\rangle \otimes |+\rangle = |1\rangle \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right),$$

sendo $|1\rangle$ o registrador de controle e $\left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right)$ o alvo. Dessa maneira, tem-se a representação na base tensorial:

$$|1\rangle \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right) = \frac{1}{\sqrt{2}} |10\rangle + \frac{1}{\sqrt{2}} |11\rangle .$$

E a aplicação da CNOT:

$$\begin{aligned} CNOT(\ \frac{1}{\sqrt{2}}\ |10\rangle + \frac{1}{\sqrt{2}}\ |11\rangle\ ) &= \frac{1}{\sqrt{2}}\, CNOT(\ |10\rangle\ )\ +\ \frac{1}{\sqrt{2}}\, CNOT(\ |11\rangle\ ) \\ &= \frac{1}{\sqrt{2}}\ |11\rangle\ +\ \frac{1}{\sqrt{2}}\ |10\rangle \\ &= |1\rangle\ \left(\frac{|1\rangle + |0\rangle}{\sqrt{2}}\right). \end{aligned}$$

A parcela $|1\rangle$ é referente ao *output* do registrador de controle e a parcela $\left(\frac{|1\rangle + |0\rangle}{\sqrt{2}}\right)$ referente ao *output* do registrador alvo.

Sua representação visual, sendo o *qubit* 0 o controle e o *qubit* 1 o alvo, é:

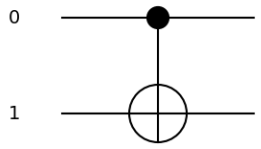

Figura 8 – Porta Lógica CNOT.

Aplicar a CNOT para a base x, sobre a componente observável S_x, gera o seguinte comportamento:

$$\begin{aligned} CNOT\ |00\rangle_x &= |00\rangle_x , \\ CNOT\ |01\rangle_x &= |11\rangle_x , \\ CNOT\ |10\rangle_x &= |10\rangle_x , \\ CNOT\ |11\rangle_x &= |01\rangle_x . \end{aligned}$$

Nota-se que usando a base x, o segundo registrador (segundo valor numérico do *ket*) trabalha como de controle e o primeiro como alvo [120], [121], [122].

4.3.4 Portas Lógicas Ternárias

Assim como porta binárias residem no $\mathcal{H}^4$, portas ternárias residem no espaço $\mathcal{H}^8$, dado pelo produto tensorial $\mathcal{H} \otimes \mathcal{H} \otimes \mathcal{H}$. Os estados base desse espaço de Hilbert também são fruto do produto tensorial de três *qubits* [120], [122], sendo:

$$
\begin{aligned}
|0\rangle \otimes |0\rangle \otimes |0\rangle &\leftrightarrow |0\rangle\,|0\rangle\,|0\rangle \leftrightarrow |000\rangle \leftrightarrow |0\rangle^3 \\
|0\rangle \otimes |0\rangle \otimes |1\rangle &\leftrightarrow |0\rangle\,|0\rangle\,|1\rangle \leftrightarrow |001\rangle \leftrightarrow |1\rangle^3 \\
|0\rangle \otimes |1\rangle \otimes |0\rangle &\leftrightarrow |0\rangle\,|1\rangle\,|0\rangle \leftrightarrow |010\rangle \leftrightarrow |2\rangle^3 \\
|0\rangle \otimes |1\rangle \otimes |1\rangle &\leftrightarrow |0\rangle\,|1\rangle\,|1\rangle \leftrightarrow |011\rangle \leftrightarrow |3\rangle^3 \\
|1\rangle \otimes |0\rangle \otimes |0\rangle &\leftrightarrow |1\rangle\,|0\rangle\,|0\rangle \leftrightarrow |100\rangle \leftrightarrow |4\rangle^3 \\
|1\rangle \otimes |0\rangle \otimes |1\rangle &\leftrightarrow |1\rangle\,|0\rangle\,|1\rangle \leftrightarrow |101\rangle \leftrightarrow |5\rangle^3 \\
|1\rangle \otimes |1\rangle \otimes |0\rangle &\leftrightarrow |1\rangle\,|1\rangle\,|0\rangle \leftrightarrow |110\rangle \leftrightarrow |6\rangle^3 \\
|1\rangle \otimes |1\rangle \otimes |1\rangle &\leftrightarrow |1\rangle\,|1\rangle\,|1\rangle \leftrightarrow |111\rangle \leftrightarrow |7\rangle^3
\end{aligned}
$$

4.3.4.1 Porta Toffoli

A porta Toffoli, também conhecida como CCNOT (*Controlled-Controlled-NOT*, utiliza dois *qubits* de controle e um *qubit* alvo. O *qubit* alvo é negado caso ambos os estados de controle estejam no estado 1 [120], [121], [122]:

$$
Toffoli = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0
\end{pmatrix}
$$

Sua representação visual, sendo os *qubits* 0 e 1 os controles e o *qubit* 2 o alvo, é:

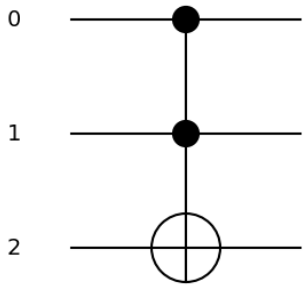

Figura 9 – Porta Lógica Toffoli.

4.3.4.2 Porta Fredkin, CSWAP

A porta Fredkin, comumente chamada de *Controlled-SWAP*, tem um *qubit* de controle e dois de alvo. Caso o controle seja 1, os *qubits* alvo são trocados entre si [120], [121]:

$$Fredkin = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Sua representação visual, sendo o *qubit* 0 o controle e os *qubits* 1 e 2 os alvos, é:

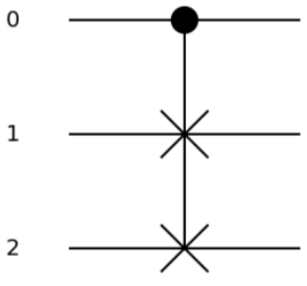

Figura 10 – Porta Lógica Fredkin.

4.3.5 Resultado Não-determinístico

Medir um *qubit* faz seu estado colapsar para uma das bases binárias 0 ou 1. Os *qubits* emaranhados têm propriedades não usuais. Nenhum *qubit* que faz parte de um

sistema onde os estados estão emaranhados tem um estado próprio. Medir qualquer parte desse sistema afeta diretamente o estado associado às demais parcelas desse sistema. Isso se dá pelo conceito de correlação, um relacionamento quantitativo importante para a análise estatística dos dados armazenados nos *qubits* [122].

Como exemplifica o livro [122], se um sistema possui dois cubos coloridos, um laranja e um verde, e Bob recebe um dos cubos, deixando Alice com o cubo sobressalente, ao Bob olhar para seu cubo e ver que ele recebeu o verde é determinístico que Alice recebeu o laranja. Essa correlação é chamada de determinística, uma vez que as observações individuais são perfeitamente correlatas [122].

Outra forma muito importante de correlação é a que ocorre entre variáveis estocásticas. Variáveis estocásticas são aquelas que perante repetitivas observações têm valores que flutuam, como observações da altura de uma população ou temperatura ao longo do ano. Essas observações individuais não mostram uma correlação óbvia dos fatores, mas as médias das observações mostram [122]. Ainda como exemplifica [122], um artigo de 2012 da *New England Journal of Medicine* mostrou a correlação entre o consumo de chocolate per capita de um país e o número de cientistas desse país que ganharam prêmios Nobel. A correlação não é precisa para todos os países, mas na média ela parece existir. Como conclusão do artigo, o consumo de chocolate supostamente aumentava a capacidade cognitiva. Correlação, no entanto, não é o mesmo que causalidade. Talvez o consumo de chocolate não tenha diretamente a ver com habilidades cognitivas mas é apenas reflexo da riqueza de um país, o que leva a mais investimento em pesquisas científicas e consequentemente mais prêmios Nobel.

As correlações descritas pelos estados emaranhados, bem como os estados independentes, são fruto da relação entre a média das aferições/medições dos *qubits* [122]. Seguindo o exemplo de [122], supõe-se N aferições da orientação do *spin* de um *qubit* independente. Cada aferição retorna $+1$ (spin-up) ou -1 (spin-down). A média das aferições sobre a componente S_z é dada por

$$\langle Z \rangle = (+1 + 1 - 1 - 1 - 1 + 1 \cdots + 1)/N.$$

Para compreender a correlação entre as aferições no caso de dois estados emranhados, utiliza-se duas variáveis F e G fictícias, substituidas posteriormente pelas médias de aferições de cada *qubit*. Serão medidas três médias: $\langle F \rangle$, $\langle G \rangle$ e $\langle FG \rangle$. O nível de correlação entre essas variáveis é frequentemente expresso pela covariância:

$$Covariância(F, G) = \langle FG \rangle - \langle F \rangle \langle G \rangle.$$

Se a covariância é igual a zero, ou próxima de zero, diz-se que as variáveis não são correlatas. Uma vez que as aferições dos estados quânticos envolvem probabilidade e aleatoriedade, a covariância pode ser usada para expressar a correlação entre as aferições dos *qubits* [122].

4.4 Solução de Problemas

Em que os computadores quânticos são bons? De acordo com o livro [121], assim como qualquer forma de computação, são bons na solução de problemas. Esses computadores são bons em alguns problemas do mundo quântico, que tratam o comportamento de sistemas quânticos, e na modelagem de soluções melhores para problemas clássicos, tornando viável a solução de problemas que exigem uma quantidade exorbitante de recursos no paradigma clássico.

4.4.1 Problemas Quânticos

De acordo com o livro [121], um sistema quântico não pode ser eficientemente simulado por um computador clássico. Isso ocorre pelo comportamento exponencial dos sistemas quânticos. Para um *qubit* evoluindo de acordo com a equação de Schrödinger, um sistema de duas equações diferenciais deve ser resolvido; para dois *qubits*, quatro equações diferenciais; três *qubits*, oito equações; e para n *qubits*, 2^n equações. Algumas aproximações podem ser feitas, reduzindo o número efetivo de equações envolvidas, o que torna a simulação do sistema quântico viável dentro da computação clássica. Existem muitos sistemas quânticos onde não são conhecidas tais aproximações, tornando impossível a modelagem desses sistemas na computação clássica.

Nem todas as evoluções Hamiltonianas podem ser simuladas de maneira eficiente na computação quântica. Processos de equilibração, especialmente aqueles com Hamiltonianos arbitrários, que são exponencialmente difíceis para computadores clássicos, ainda são um problema em aberto para os computadores quânticos. Outros sistemas quânticos são eficientemente simuláveis pelos computadores quânticos.

Ao contrário do mundo clássico, onde podemos facilmente distinguir e acompanhar objetos diferentes, a mecânica quântica nos impõe limitações quando se trata de partículas idênticas. Enquanto podemos diferenciar partículas distintas, como prótons e elétrons, medindo propriedades como carga, partículas idênticas, como dois elétrons, são indistinguíveis.

A indistinguibilidade de partículas resulta em duas categorias principais: bósons e férmions. Bósons, como os fótons, mantêm seu vetor de estado inalterado quando as partículas têm suas posições trocadas, refletindo sua indistinguibilidade fundamental. Férmions, como elétrons, experimentam uma mudança de sinal em seu vetor de estado sob a permutação. A simulação eficiente desses sistemas quânticos, levando em consideração tais características, é possível em computadores quânticos.

O processo de simulação envolve respeitar as simetrias específicas desses sistemas. Se um estado inicial não possui a simetria correta, ele pode ser ajustado antes da simulação começar. Os operadores utilizados na simulação são construídos para respeitar a simetria

desejada, mesmo considerando os efeitos de erros de ordem superior. Essa capacidade de simular sistemas quânticos eficientemente em computadores quânticos, especialmente aqueles que envolvem simetrias globais relacionadas às estatísticas de partículas, destaca a promessa dessas máquinas para resolver problemas específicos da física quântica.

4.4.2 Problemas Clássicos

São conhecidas duas amplas classes de algoritmos quânticos que solucionam problemas clássicos computacionalmente inviáveis pelo paradigma clássico. A primeira classe de algoritmos é baseada na transformada quântica de Fourier de Shor (*Shor's quantum Fourier transform*), essa classe inclui algoritmos notáveis para resolver os problemas de fatoração e logaritmo discreto, proporcionando uma notável aceleração exponencial em relação aos melhores algoritmos clássicos conhecidos. A segunda classe de algoritmos é baseada no algoritmo de Grover para realizar buscas quânticas, proporcionando uma aceleração quadrática em relação aos melhores algoritmos clássicos possíveis. O algoritmo de busca quântica deriva sua importância do uso generalizado de técnicas baseadas em busca em algoritmos clássicos [121].

O algoritmo de busca quântica tem potencial para muitas aplicações. Esse pode ser usado para extrair estatísticas, como o elemento mínimo de um conjunto de dados não ordenados, mais rapidamente do que é possível em um computador clássico. Pode acelerar algoritmos para alguns problemas em NP, especificamente aqueles problemas para os quais uma busca direta por uma solução é o melhor algoritmo conhecido. E pode ser usado para acelerar a busca por chaves de sistemas de criptografia simétrica [121].

A transformada quântica de Fourier pode ser usada para resolver os problemas de logaritmo discreto e fatoração, permitindo que um computador quântico quebre muitos sistemas de criptografia populares, como o RSA. A transformada de Fourier também está intimamente relacionada a um problema importante em matemática, encontrar um subgrupo oculto, um caso mais geral do problema de encontrar o período de uma função periódica [121].

Poucos algoritmos quânticos, por mais que sejam melhores que seus homólogos clássicos, são conhecidos. Isso se dá pela dificuldade em criar algoritmos, sejam clássicos ou quânticos, melhores que aqueles já existentes, demandando uma grande engenhosidade para superar o que há de melhor. Outra dificuldade na concepção de algoritmos quânticos é a propensão natural do homem à lógica clássica. É de fato contra intuitivo pensar de forma quântica, requerendo *insights* e abordagens especiais para superar essa barreira intuitiva [121].

4.5 Aprendizado de Máquina Híbrido

Como destacado na revisão sistemática [29], diversos estudos exploram a ideia de usar a vantagem quântica para melhorar os modelos de Aprendizado de Máquina. Esforço está sendo colocado em desenvolver modelos completos de Redes Neurais em suas versões quânticas, geralmente baseados numa perspectiva biológica, no entanto nenhum avanço significativo foi alcançado. Alguns autores desenvolvem algoritmos completamente quânticos para reconhecimento de padrões.

Outras propostas sugerem o uso da computação quântica em sub-rotinas clássicas com o propósito de ganhar velocidade, o uso da computação quântica adiabática para problemas de otimização combinatória e até a implementação elegante de modelos estocásticos como a Teoria de Decisão Bayesiana e Modelos de Markov ocultos.

Este trabalho implementará um modelo de *Support Vector Machine* com uma sub-rotina designada à computação quântica. O modelo é considerado híbrido uma vez que combina a computação clássica e quântica, visando tirar proveito do melhor de ambos os paradigmas.

4.5.1 Quantum Enhanced Support Vector Machine

O modelo clássico de *Support Vector Machine*, como explicado em 2.4.4.1, funciona encontrando um hiperplano que seja capaz de melhor discriminar ambas as classes, servindo de fronteira para classificar futuras observações [29]. Na seção 2.4.4.1, o conceito de *kernel* também foi explicado, bem como sua representação em formato matricial.

A matriz que representa o *kernel* pode ser chamada de matriz de similaridade. Ela recebe esse nome uma vez que, para todas as combinações de instâncias do conjunto de dados, essa matriz mostra a similaridade dessas instâncias como um valor entre 0 e 1. Quanto mais próximo de 1 mais similares, sendo 1 instâncias iguais. Para o valor 0 essas são ortonormais e completamente distintas.

O cálculo dessa matriz é designado à parcela quântica do modelo híbrido, mantendo o SVM inalterado, e pode ser realizado de maneiras diferentes. Esse trabalho tem por objetivo implementar e comparar dois métodos: Produto Escalar [87], [84], [123] e *Controlled Swap Test* [84], [1].

O método de Produto Escalar [87], [84], [123] aproveita a codificação dos dados nos *qubits* por meio de *feature maps* para calcular o produto interno dos dados par a par, populando a matriz de similaridade, uma matriz de covariância. Um *feature map* é uma operação unitária genérica $U(x)$ responsável por mapear um valor clássico x para um *qubit*, manipulando seu *spin* por meios das portas lógicas. Para um dado genérico x,

tem-se:

$$|\phi(x)\rangle = U(x)\,|0\rangle\,.$$

Como visto na seção 2.4.4.1, basta calcular o produto interno dos pares de instâncias de dados x_i e x_j, dado por

$$k(x_i, x_j) = |\,\langle\phi(x_j)|\phi(x_i)\rangle\,|^2.$$

Para alcançar $\langle\phi(x_j)|$ é necessário construir o *adjoint*, conjugado transposto, de $U(x)$, o $U^\dagger(x)$. Representa-se o produto interno das duas instâncias em termos da operação U como:

$$|\,\langle\phi(x_j)|\phi(x_i)\rangle\,|^2 = |\,\langle 0|\,U^\dagger(x_j)U(x_i)\,|0\rangle\,|^2. \tag{4.2}$$

Para o método de *Controlled Swap Test* [84], [1], a equivalência de dois estados quânticos puros de *n-qubit* $|\psi\rangle$ e $|\phi\rangle$ é medida a partir da aplicação do circuito da figura:

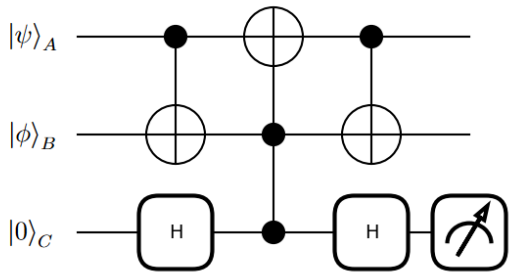

Figura 11 – Swap Test implementado usando portas Toffoli e CNOT. Adaptado de [1].

Sua implementação para estados de dois *qubits* é:

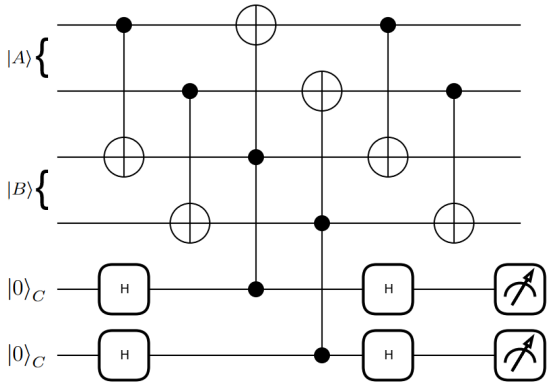

Figura 12 – Swap Test implementado usando portas Toffoli e CNOT para estados de 2 *qubits*. Adaptado de [1].

Por equivalência, a porta *Controlled SWAP* pode ser escrita como CNOT, *Toffoli* e CNOT [124], mudando o circuito da figura anterior para uma implementação mais simples:

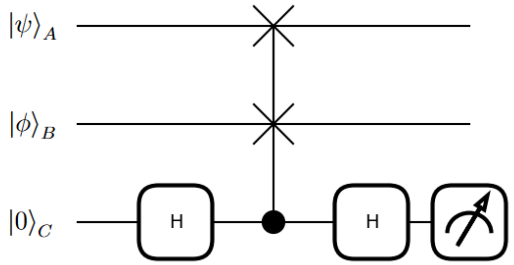

Figura 13 – Swap Test implementado usando porta CSWAP.

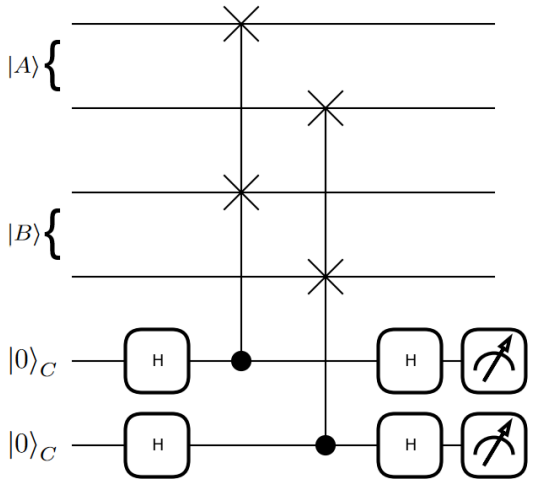

Figura 14 – Swap Test implementado usando porta CSWAP para estados de 2 *qubits*.

A composição inicial $|\Psi\rangle = |\psi\rangle_A |\phi\rangle_B |0\rangle_C$, dos estados puros e do estado de controle, após a aplicação do circuito, fica:

$$|\Psi\rangle = \frac{1}{2}[(|\phi\rangle_A |\psi\rangle_B + |\psi\rangle_A |\phi\rangle_B) |0\rangle_C + (|\phi\rangle_A |\psi\rangle_B - |\psi\rangle_A |\phi\rangle_B) |1\rangle_C].$$

É trivial, portanto, que se $|\phi\rangle = |\psi\rangle$, o *qubit* de controle será $|0\rangle_C$ com total certeza.

Medir a probabilidade do *qubit* de controle ser $|0\rangle$ retorna a probabilidade dos estados $|\phi\rangle$ e $|\psi\rangle$ serem iguais. Essa probabilidade popula a matriz de similaridade, gerando o *kernel* para nosso modelo híbrido.

A Computação Quântica apresenta um potencial computacional capaz de quebrar a Lei de Moore, vigente desde 1965. Essa capacidade é justificada pela física moderna, uma vez que a máquina quântica tira proveito da superposição de partículas sub-atômicas para realizar seus cálculos. O *qubit* em superposição representa simultaneamente ambos os valores binários, possuindo uma probabilidade α de ser 0 e β de ser 1. Quando combinado com outros x *qubits*, juntos representam todos os estados binários possíveis de $x + 1$ dígitos. Somente quando medidos os *qubits* colapsam para um valor determinístico. Sendo assim, durante a computação do algoritmo quântico todas as possibilidades de resultado são tratadas simultaneamente. É evidente que essa característica traz uma nova forma de modelar problemas computacionalmente, que pode ser aproveitada para solucionar desafios intratáveis na computação clássica, como faz o algoritmo de Shor na fatoração de números grandes. Os empecilhos desse paradigma, no entanto, evidenciam que a computação quântica está longe de substituir os computadores atuais.

O não-determinismo deste modelo computacional exige que diversas execuções aconteçam para que uma média entre os resultados seja extraída e somente assim obter o resultado final de dado algoritmo. Essa característica pode fazer com que problemas já tratáveis na computação clássica não sejam otimizados pelo uso do computador quântico. Outra dificuldade em utilizar essas máquinas é a instabilidade dos *qubits*, dificultando a computação de algoritmos mais extensos. Tanto interferência quanto colapsamento espontâneo acabam invalidando um *qubit* durante uma execução. Para contornar esse problema a redundância pode ser utilizada, mas sistemas com um número elevado de *qubits* é mais custoso e difícil de construir. Modelar problemas nativamente quânticos, porém, não enfrenta as dificuldades provenientes do não-determinismo, uma vez que comportam-se como o computador quântico. Por esse motivo a computação quântica é atrativa para o estudo de ambientes quânticos, simulando sem dificuldade os trejeitos do mundo sub-atômico.

5 ESTUDO DE CASO

A partir deste ponto, o trabalho mostra a implementação dos modelos de SVM clássico e híbridos explicados anteriormente, analisando a viabilidade de ambas as implementações como um Sistema de Detecção de Intrusão de Redes.

5.1 Cenário de Dados

Para a realização dos estudo acerca da implementação desses modelos, foram utilizados dois conjuntos de dados gerados pelo Grupo Orion de pesquisa em redes da Universidade Estadual de Londrina. Cada conjunto de dados conta com fluxos IP referentes ao tráfego da rede em 24 horas. Ambos os dias apresentam fluxos normais da rede e ataques DDoS e *Portscan* combinados.

Os dados foram gerados utilizando a ferramenta *Mininet* a fim de emular uma rede SDN com 6 *switches*, 60 *hosts* e 1 *controller*, a biblioteca *Scapy* para gerar o tráfego da rede e uma combinação do software *hping3* e *Scapy* para simular os ataques DDoS e *Portscan*. A topologia da rede é como mostra a figura 15.

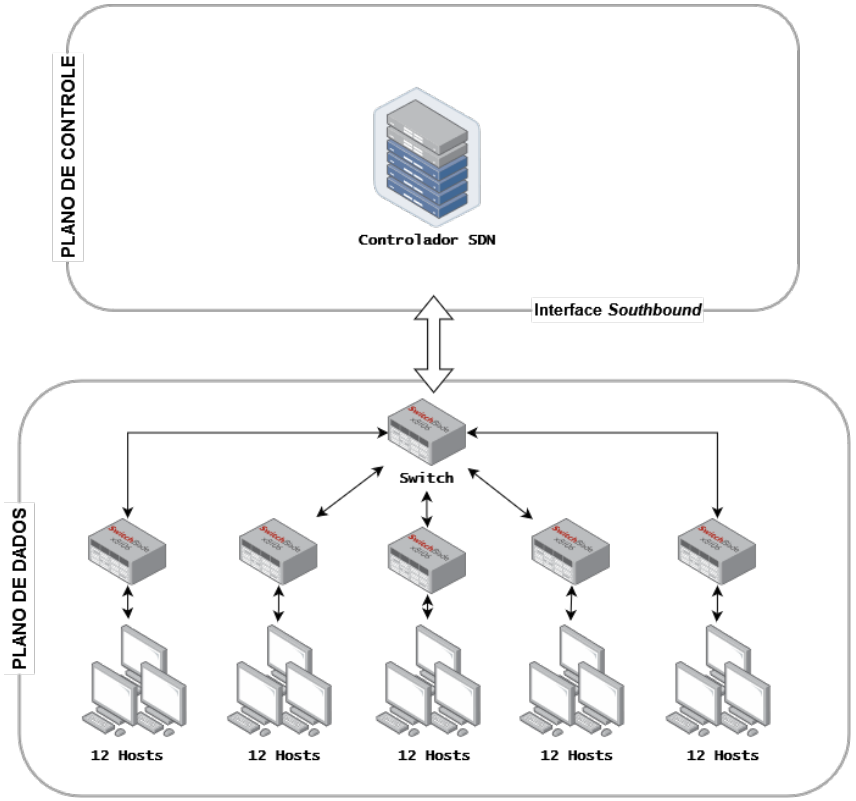

Figura 15 – Topologia da rede emulada no *Mininet*.

Os fluxos IP coletados são irregulares e requerem um pré-processamento para

tornarem-se utilizáveis pelo modelo de Aprendizado de Máquina. Os fluxos são agrupados em intervalos de 1 segundo, condensando os atributos de *timestamp* de início do fluxo, endereço IP e porta de origem, endereço IP e porta de destino, quantidade de pacotes e bits transferidos em observações estatísticas de mesma duração.

Os atributos numéricos dos fluxos, como quantidade de pacotes e bits transferidos, são diretamente operáveis, enquanto atributos nominais como os endereços IP e portas de origem e destino precisam ser condensados em atributos quantitativos por meio da Entropia de Shannon [125].

A Entropia de Shannon [125] transforma os endereços IP e portas em valores que representam a concentração ou dispersão de sua ocorrência, representando a desordem desses atributos. Para o histograma da quantidade de ocorrências dos diferentes possíveis valores do atributo qualitativo A no período t, x_t^A, temos $x_t^A = \{x_1, x_2, \ldots, x_N\}$, onde x_i indica a quantidade de ocorrências do valor i, S o total de valores observados para o atributo A no intervalo e $\frac{x_i}{S}$ a probabilidade do valor i ocorrer. Sendo assim, temos a fórmula:

$$H(x_t^A) = -\sum_{i=1}^{N}(\frac{x_i}{S})log_2(\frac{x_i}{S})$$

Cada dia analisado é dividido em 86.400 janelas de 1 segundo, sendo cada janela composta por seis atributos: total de bits transferidos, total de pacotes transferidos, entropia de IP de origem, entropia de IP de destino, entropia de porta de origem e entropia de porta de destino.

Os dois dias contendo ataques são ilustrados segundo cada um dos seis atributos a seguir nas figuras 16 e 17:

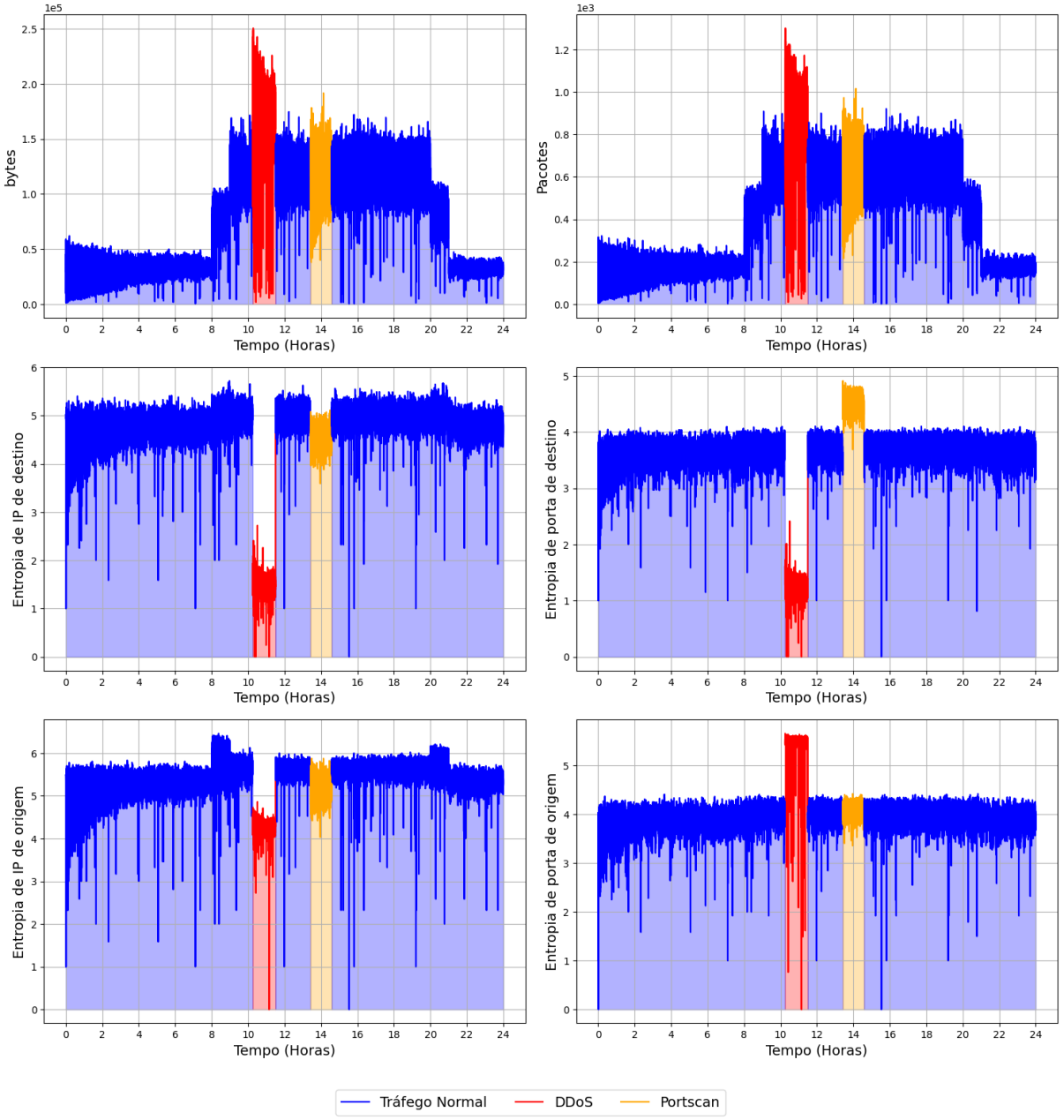

Figura 16 – Primeiro dia sob ataque.

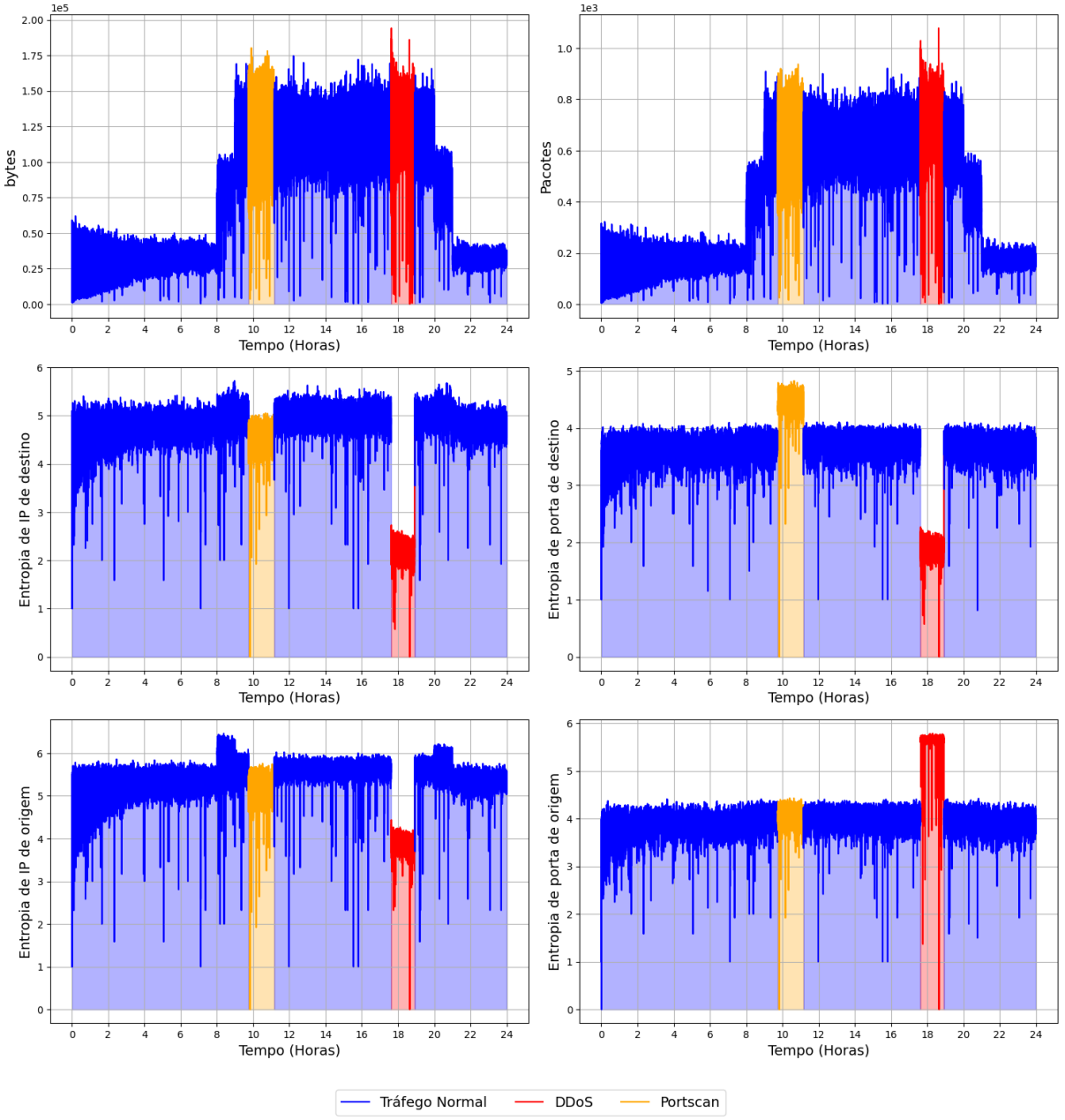

Figura 17 – Segundo dia sob ataque.

Antes de alimentar os dados ao modelo, uma última etapa de processamento é necessária. Os dados são normalizados em valores entre 0 e 1 a partir de seus mínimos e máximos. Para tal, fora utilizado o *MinMaxScaler* da biblioteca *scikit-learn*. De acordo com [126], o *MinMaxScaler* é descrito da seguinte maneira, sendo X_i o i-ésimo elemento do conjunto de dados X, e as funções *min()* e *max()* retornando o menor e maior valor de X respectivamente:

$$X_{escalado} = (X_i - min(X))/(max(X) - min(X)).$$

5.2 Sistemas de Detecção de Intrusão de Rede

O Sistema de Detecção de Intrusão tira proveito da interface *Northbound* e opera na camada de aplicação, implementando o modelo de Aprendizado de Máquina e alimentando-o com os dados coletados do tráfego da rede que passam pelo controlador SDN. O modelo é treinado com dados históricos e novos dados precisam ser relacionados por similaridade com os dados do treinamento para então serem alimentados ao modelo. Isso ocorre uma vez que o modelo foi treinado usando uma matriz de similaridade dos dados par-a-par.

5.2.1 *Support Vector Machine* Clássico

O SVM clássico contempla uma matriz de *kernel* calculada usando a biblioteca *numpy* para a linguagem *python*. Assim como os demais *kernels* que serão apresentados, o objetivo deste é relacionar os dados par-a-par por similaridade e alimentá-lo ao modelo.

A matriz é calculada por meio de produto escalar dos dados como mostra o pseudo-código:

Algorithm 1: Cálculo da matriz de kernel

```
1 Input: amostras_treino
2 Output: matriz_kernel
3 matriz_kernel = [ ][ ]
4 for i = 0 ; i < numero_amostras_treino ; i++ do
5     for j = i ; j < numero_amostras_treino ; j++ do
6         matriz_kernel[i][j] = <amostras_treino[i] , amostras_treino[j]>
7         matriz_kernel[j][i] = matriz_kernel[i][j]
8 return matriz_kernel
```

O pseudo-código 1 calcula a diagonal superior da matriz e a espelha na diagonal inferior a fim de salvar tempo de processamento. Já que produto escalar é comutativo, calcular $\langle X, Y \rangle$ é o mesmo que calcular $\langle Y, X \rangle$, portanto essa otimização pode ser feita.

Para os dados novos a serem classificados pelo modelo, o seguinte deve ser calculado:

Algorithm 2: Cálculo da matriz de similaridade dos novos dados

```
Input: amostras_treino, amostras_novas
Output: matriz_similaridade
matriz_similaridade = [ ][ ]
for i = 0 ; i < numero_amostras_novas ; i++ do
    for j = 0 ; j < numero_amostras_treino ; j++ do
        matriz_similaridade[i][j] =
         <amostras_novas[i] , amostras_treino[j]>
return matriz_similaridade
```

A matriz de similaridade dos dados novos com os dados de treino é alimentada no modelo e o mesmo retorna a sua predição.

5.2.2 *Support Vector Machine* Quântico - Kernel 1

Para o SVM híbrido com *kernel* computado por Produto Escalar, as matrizes continuam as mesmas, no entanto, o produto escalar é calculado pela aplicação da equação 4.2. Essa equação é atingida pela aplicação do *Feature Map* para o x_i (circuito da figura 18) e, logo em seguida, do *Feature Map adjoint* para o x_j (circuito da figura 19).

Dessa maneira, temos os pseudo-códigos:

Algorithm 3: Cálculo da matriz de kernel - Produto Escalar

```
Input: amostras_treino
Output: matriz_kernel
matriz_kernel = [ ][ ]
for i = 0 ; i < numero_amostras_treino ; i++ do
    for j = i ; j < numero_amostras_treino ; j++ do
        matriz_similaridade[i][j] = Produto(amostras_treino[i],
         amostras_treino[j])
        matriz_kernel[j][i] = matriz_kernel[i][j]
return matriz_kernel
```

Algorithm 4: Cálculo da matriz de similaridade dos novos dados

```
Input: amostras_treino, amostras_novas
Output: matriz_similaridade
matriz_similaridade = [ ][ ]
for i = 0 ; i < numero_amostras_novas ; i++ do
    for j = 0 ; j < numero_amostras_treino ; j++ do
        matriz_similaridade[i][j] = Produto(amostras_novas[i],
         amostras_treino[j])
return matriz_similaridade
```

E a função *Produto*() é definida como:

Algorithm 5: Produto Escalar Quântico

```
Input: amostra_a, amostra_b
Output: produto_escalar
FeatureMap (amostra_a, amostra_b)
FeatureMapAdjoint (amostra_a, amostra_b)
return Probabilidade do estado ser 0
```

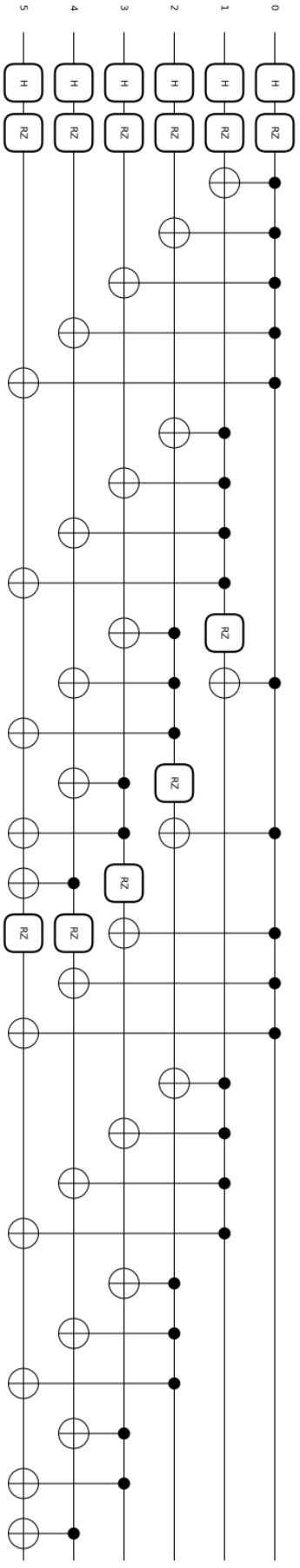

Figura 18 – *Feature Map*.

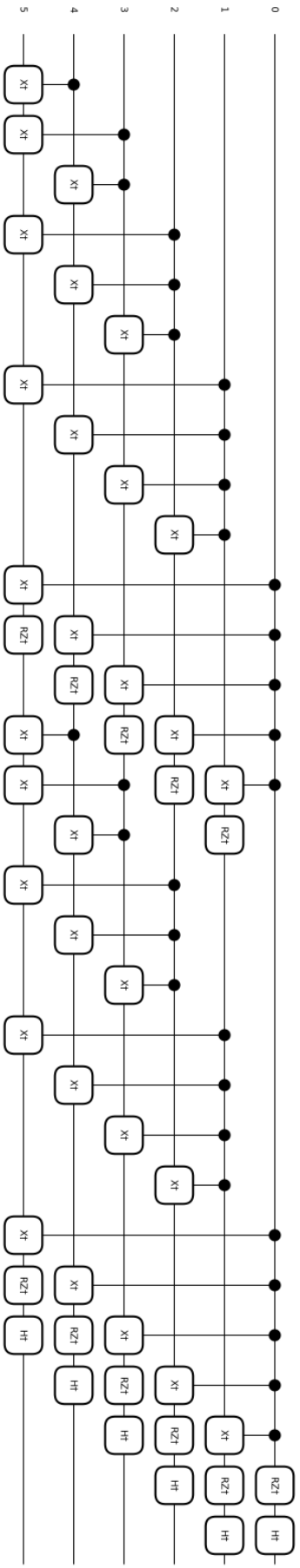

Figura 19 – *Feature Map adjoint.*

5.2.3 *Support Vector Machine* Quântico - Kernel 2

Como explicado em 4.5.1, o cálculo da similaridade entre dois estados quânticos utilizando o *Swap Test* pode ser descrito com os seguintes pseudo-códigos:

Algorithm 6: Cálculo da matriz de kernel - Swap Test

```
1 Input: amostras_treino
2 Output: matriz_kernel
3 matriz_kernel = [ ][ ]
4 for i = 0 ; i < numero_amostras_treino ; i++ do
5     for j = i ; j < numero_amostras_treino ; j++ do
6         matriz_similaridade[i][j] = Produto(amostras_treino[i],
           amostras_treino[j])
7         matriz_kernel[j][i] = matriz_kernel[i][j]
8 return matriz_kernel
```

Algorithm 7: Cálculo da matriz de similaridade dos novos dados

```
1 Input: amostras_treino, amostras_novas
2 Output: matriz_similaridade
3 matriz_similaridade = [ ][ ]
4 for i = 0 ; i < numero_amostras_novas ; i++ do
5     for j = 0 ; j < numero_amostras_treino ; j++ do
6         matriz_similaridade[i][j] = Produto(amostras_novas[i],
           amostras_treino[j])
7 return matriz_similaridade
```

E a função *Produto*() é definida como:

Algorithm 8: Produto Swap Test Quântico

```
1 Input: amostra_a, amostra_b
2 Output: produto_escalar
3 FeatureMap(amostra_a, fios = [0, 1, 2, 3, 4, 5])
4 FeatureMap(amostra_b, fios = [6, 7, 8, 9, 10, 11])
5 * Aplicação do circuito do Swap Test *
6 return Probabilidade do estado ser 0
```

5.3 Resultados e Discussão

As três variantes do Sistema de Detecção de Intrusão foram executadas a partir da base de dados Orion descrita na seção anterior. Uma vez que os *kernels* quânticos foram executados em simulação utilizando o *framework PennyLane* numa máquina clássica, a velocidade de processamento torna-se um impedimento. Foram selecionadas 21.600 das 86.400 instâncias de dados em ambos os conjuntos de dados. Essa seleção contempla todos as janelas de fluxo anômalas, completando o restante das instâncias de tráfego normal.

Os dados selecionados foram reagrupados em janelas de 60, 30 e 15 segundos, sendo cada nova janela a média dos valores registrados nas observações desses períodos. As 21.600 amostras selecionadas foram reduzidas para 360, 720 e 1.440 amostras.

Foram conduzidos 18 testes envolvendo as combinações das 3 variações do modelo SVM, 2 conjuntos de dados, sendo cada conjunto reagrupado em 3 variações de janelas. A execução foi feita no servidor remoto do Grupo Orion com a seguinte configuração:

Sistema Operacional	Ubunutu 22.04.2 LTS
RAM	64 GB DDR3
Processador	24 x AMD Opteron™ Processor 6344 @ 2.6GHz x 12
Python	3.10.12
PennyLane	0.34.0

Tabela 2 – Configuração Máquina ORION.

5.3.1 Métricas

Ao comparar as classificações feitas pelo Sistema de Detecção de Intrusão com os resultados esperados para os tráfegos analisados, existem quatro resultados possíveis. Um verdadeiro positivo (VP) ocorre quando o modelo classifica corretamente um tráfego anômalo como anômalo. Um falso positivo (FP) acontece quando o modelo classifica erroneamente um tráfego benigno como anômalo. Um falso negativo (FN) ocorre quando o modelo classifica erroneamente um tráfego anômalo como benigno. Por fim, um verdadeiro negativo (VN) acontece quando o modelo classifica corretamente um tráfego benigno como benigno. O agrupamento dessas situações forma a chamada matriz de confusão, como mostra a figura 20. A partir da matriz de confusão, derivam-se as métricas *Accuracy*, *Precision*, *Recall*, *F1-score* e *Matthew's Correlation Coefficient* [127], [128].

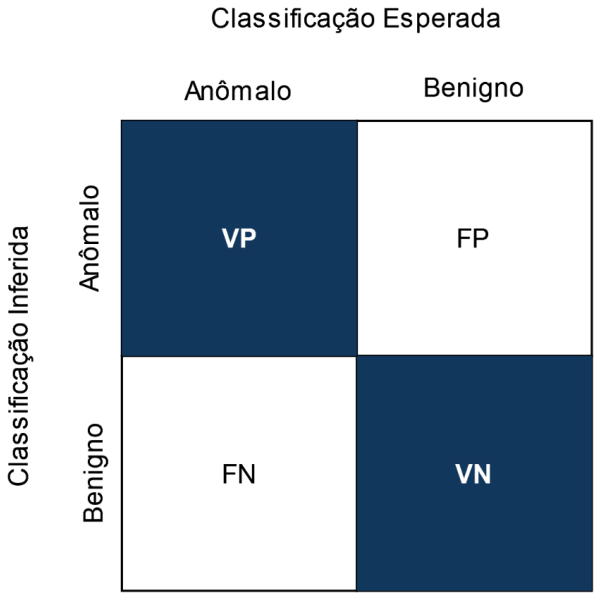

Figura 20 – Matriz de confusão.

Accuracy é a mais simples e direta das métricas utilizadas. Seu objetivo é representar a porcentagem de classificações corretas do modelo, sendo descrita pela fórmula:

$$Accuracy = \frac{VP + VN}{VP + VN + FP + FN}$$

Precision, também conhecida como *positive predictive value*, é a métrica que representa a parcela das observações classificadas corretamente como anômalas pelo sistema. Quanto menor a taxa de falsos positivos, melhor a *Precision*.

$$Precision = \frac{VP}{VP + FP}$$

O *Recall*, também denominada de *true positive rate*, indica a parcela total das observações anômalas corretamente classificadas pelo sistema. Quanto menos falsos negativos, melhor o *Recall*.

$$Recall = \frac{VP}{VP + FN}$$

A média harmônica entre *Precision* e *Recall* resulta na métrica *F1-score*. Essa métrica é distribuída em valores entre 0 e 1 e representa o quão poucos falsos positivos e negativos o sistema possui.

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

A métrica *Matthew's Correlation Coefficient* (MCC) relaciona toda a matriz de confusão, gerando um resultado entre -1 e 1. O valor 1 representa um modelo que classifica

todas as observações corretamente, enquanto -1 representa modelos que classificam as instâncias de maneira invertida de acordo com o esperado. Um MCC com valor 0 mostra que o modelo não sabe classificar as observações. Quanto menos falsos positivos e negativos o sistema apresentar, melhor seu MCC.

$$MCC = \frac{VP * VN - FP * FN}{\sqrt{(VP + FP)(VP + FN)(VN + FP)(VN + FN)}}$$

O *kernel* do modelo será uma matriz de covariância entre as instâncias de dados par-a-par. Será comparado o tempo médio para a execução do cálculo da similaridade entre dois pontos do conjunto em cada método. Esse cálculo representa uma posição na matriz de *kernel* e é determinado pela média de tempo para cada similaridade individual dentre 100. Mesmo utilizando simulação para executar os modelos híbridos clássico-quânticos, pode-se estabelecer um comparativo entre as abordagens quanto ao custo computacional.

5.3.2 Comparação dos Sistemas

Ambas as variações do sistema foram executadas na máquina descrita na tabela 2. Os conjuntos de dados foram seccionados como descrito anteriormente e alimentaram o modelo que realizou o treinamento com 80% do conjunto e teste com 20% do conjunto. As amostras que compuseram o sub-conjunto de treino e teste foram aleatorizadas por meio da ferramenta *split* da biblioteca *scikit-learn* e mantidas entre as execuções pelo emprego de uma *seed*.

5.3.2.1 *Support Vector Machine* Clássico

O tempo médio para executar o cálculo de uma similaridade por esse método é de $4,966 \times 10^{-6}$ segundos.

Os resultados para o modelo clássico está tabulado em 3.

	Dia 1			**Dia 2**		
Janela em segundos	**15**	**30**	**60**	**15**	**30**	**60**
Accuracy	1,000	1,000	1,000	1,000	1,000	1,000
Precision	1,000	1,000	1,000	1,000	1,000	1,000
Recall	1,000	1,000	1,000	1,000	1,000	1,000
F1-score	1,000	1,000	1,000	1,000	1,000	1,000
MCC	1,000	1,000	1,000	1,000	1,000	1,000

Tabela 3 – Testes modelo clássico.

As matrizes de confusão para os testes da tabela 3 são:

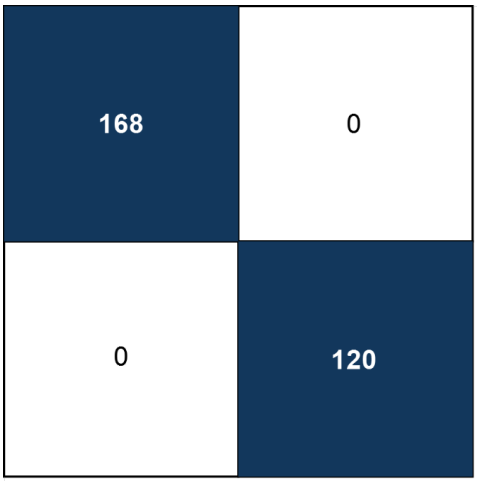

Figura 21 – *Kernel* clássico - Dia 1 - Janela 15s.

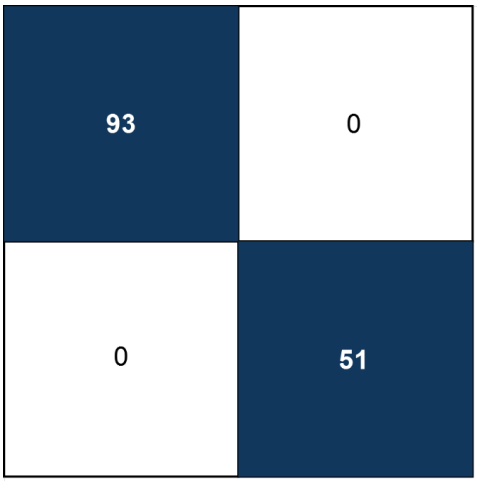

Figura 22 – *Kernel* clássico - Dia 1 - Janela 30s.

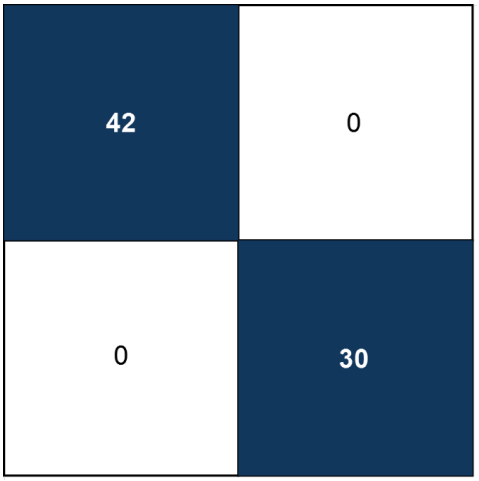

Figura 23 – *Kernel* clássico - Dia 1 - Janela 60s.

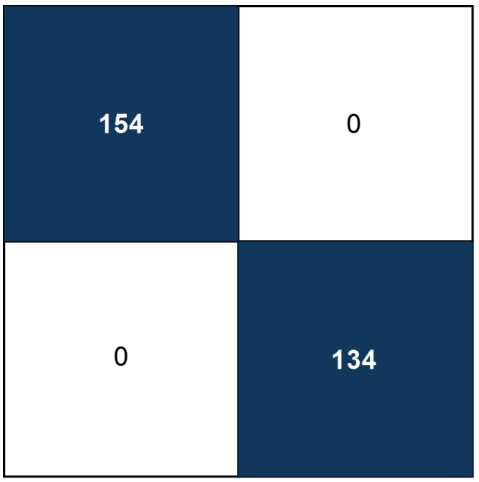

Figura 24 – *Kernel* clássico - Dia 2 - Janela 15s.

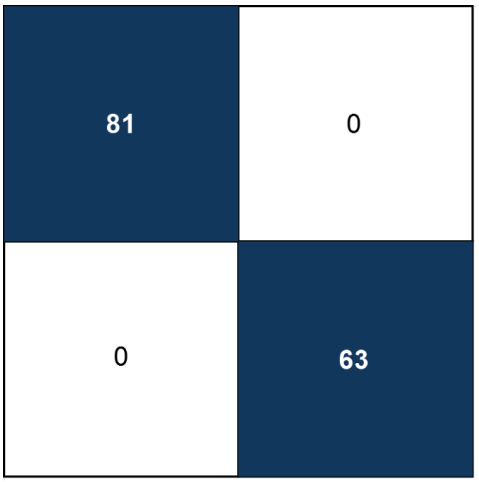

Figura 25 – *Kernel* clássico - Dia 2 - Janela 30s.

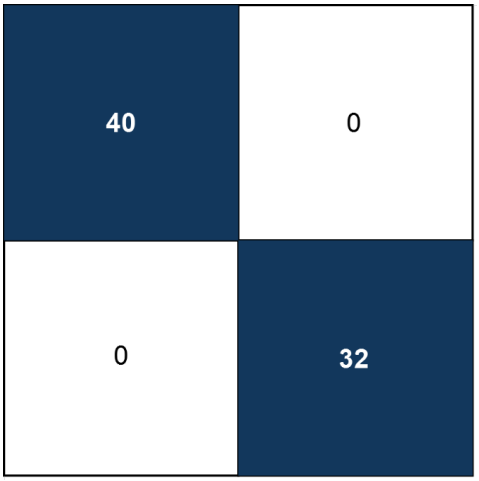

Figura 26 – *Kernel* clássico - Dia 2 - Janela 60s.

5.3.2.2 *Support Vector Machine* Quântico - Kernel 1

O tempo médio para executar o cálculo de uma similaridade por esse método é de $2,853 \times 10^{-2}$ segundos.

Os resultados para o modelo híbrido com *kernel* gerado por produto escalar está tabulado em 4.

	Dia 1			**Dia 2**		
Janela em segundos	**15**	**30**	**60**	**15**	**30**	**60**
Accuracy	1,000	1,000	1,000	1,000	0,993	1,000
Precision	1,000	1,000	1,000	1,000	1,000	1,000
Recall	1,000	1,000	1,000	1,000	0,984	1,000
F1-score	1,000	1,000	1,000	1,000	0,992	1,000
MCC	1,000	1,000	1,000	1,000	0,986	1,000

Tabela 4 – Testes modelo quântico *kernel* 1.

As matrizes de confusão para os testes da tabela 4 são:

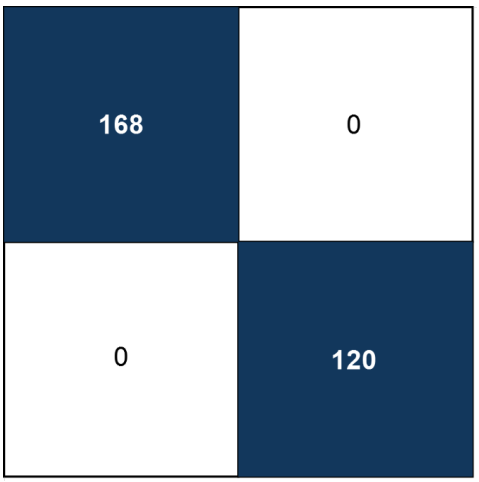

Figura 27 – Quântico *Kernel* 1 - Dia 1 - Janela 15s.

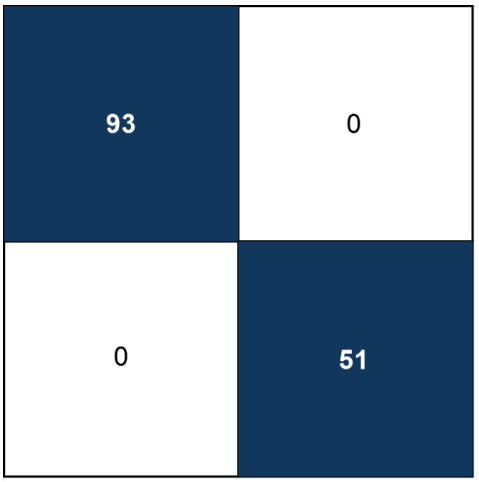

Figura 28 – Quântico *Kernel* 1 - Dia 1 - Janela 30s.

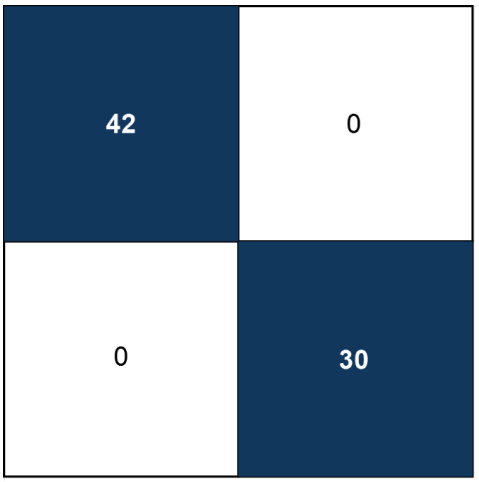

Figura 29 – Quântico *Kernel* 1 - Dia 1 - Janela 60s.

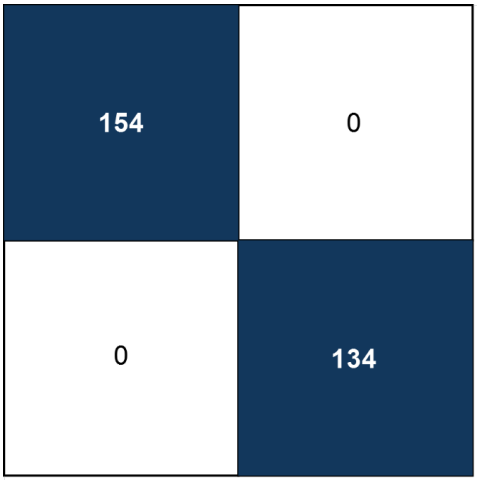

Figura 30 – Quântico *Kernel* 1 - Dia 2 - Janela 15s.

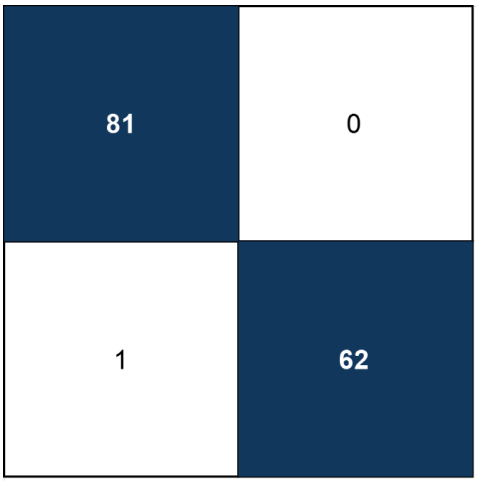

Figura 31 – Quântico *Kernel* 1 - Dia 2 - Janela 30s.

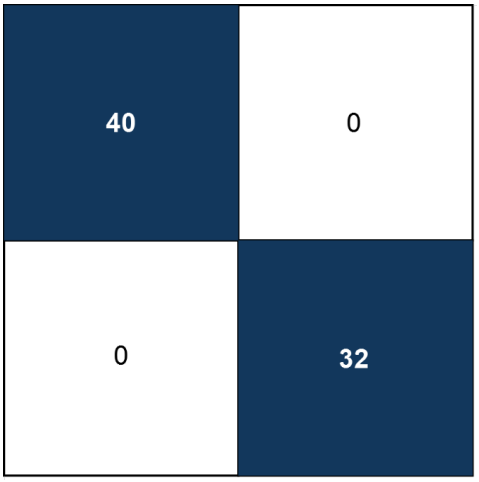

Figura 32 – Quântico *Kernel* 1 - Dia 2 - Janela 60s.

5.3.2.3 *Support Vector Machine* Quântico - Kernel 2

O tempo médio para executar o cálculo de uma similaridade por esse método é de $6,668 \times 10^{-1}$ segundos.

Os resultados para o modelo híbrido com *kernel* gerado por *swap test* está tabulado em 5.

	Dia 1			**Dia 2**		
Janela em segundos	**15**	**30**	**60**	**15**	**30**	**60**
Accuracy	1,000	1,000	1,000	1,000	1,000	1,000
Precision	1,000	1,000	1,000	1,000	1,000	1,000
Recall	1,000	1,000	1,000	1,000	1,000	1,000
F1-score	1,000	1,000	1,000	1,000	1,000	1,000
MCC	1,000	1,000	1,000	1,000	1,000	1,000

Tabela 5 – Testes modelo quântico *kernel* 2.

As matrizes de confusão para os testes da tabela 5 são:

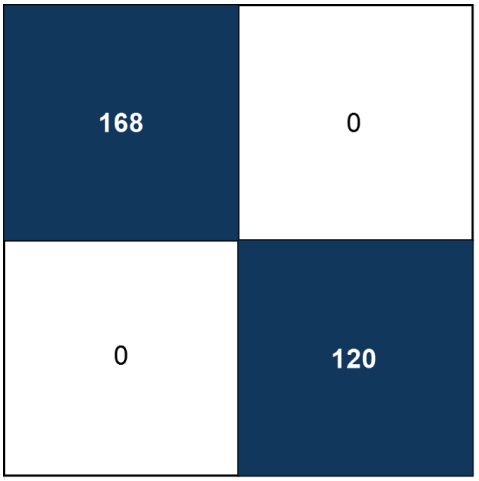

Figura 33 – Quântico *Kernel* 2 - Dia 1 - Janela 15s.

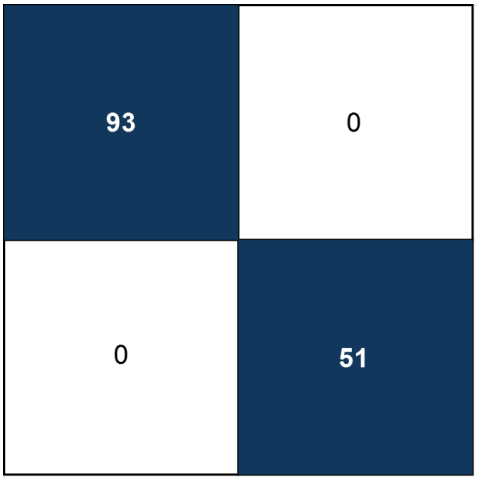

Figura 34 – Quântico *Kernel* 2 - Dia 1 - Janela 30s.

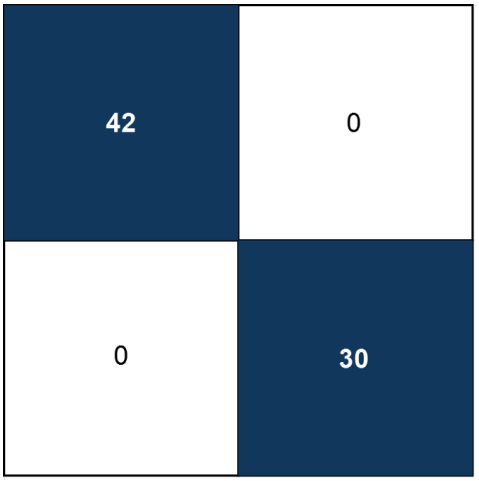

Figura 35 – Quântico *Kernel* 2 - Dia 1 - Janela 60s.

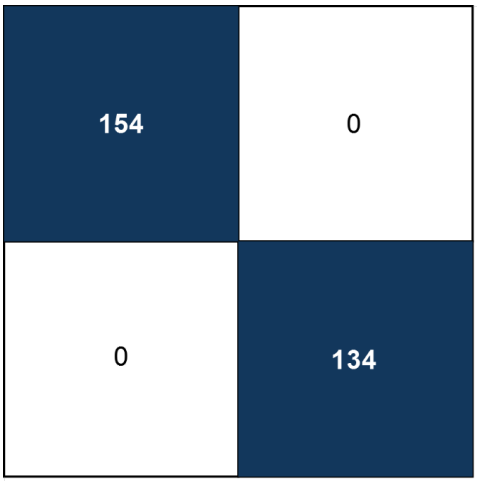

Figura 36 – Quântico *Kernel* 2 - Dia 2 - Janela 15s.

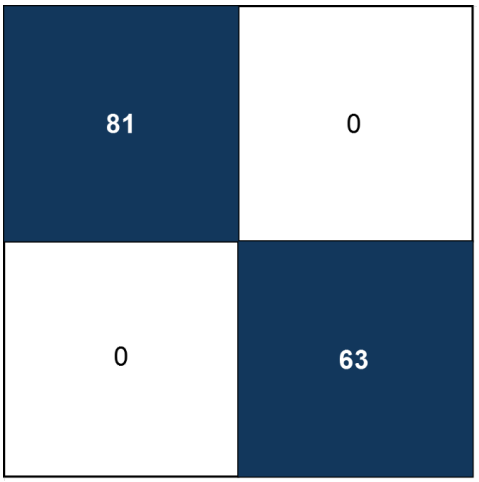

Figura 37 – Quântico *Kernel* 2 - Dia 2 - Janela 30s.

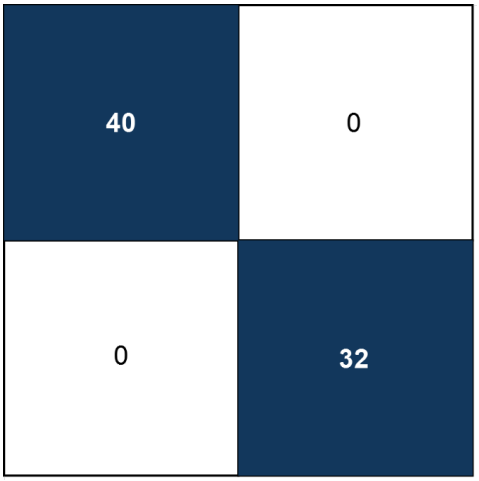

Figura 38 – Quântico *Kernel* 2 - Dia 2 - Janela 60s.

6 CONCLUSÃO

A Computação Quântica apresenta uma nova forma de resolver problemas. Com ela vêm o desafio de compreender um paradigma totalmente novo, baseado na mecânica quântica. O Computador Quântico manipula um sistema fechado de partículas sub-atômicas, podendo ser implementado de diversas maneiras. Do modelo adiabático ao orientado por portas lógicas, uma nova forma de programar é introduzida.

Cada implementação de Máquina Quântica tem suas particularidades e pode ser melhor aproveitada quando utilizada para resolver desafios mais específicos. Problemas nativamente quânticos são o foco desse paradigma. No entanto, algoritmos como o de Shor são alternativas quânticas para dilemas clássicos até então intratáveis.

O modelo quântico mais similar ao computador clássico é o baseado em portas lógicas, sendo o escolhido para este trabalho. Os dados clássicos são mapeados para os *qubits* e manipulados por meio de circuitos bem estabelecidos. Esses alteram fisicamente a orientação do *bit* quântico para obter a computação desejada, tirando proveito da eficiência proveniente do não-determinismo quântico.

Combinando a Computação Quântica e o modelo de Aprendizado de Máquina Raso *Support Vector Machine*, o problema clássico de Detecção de Intrusão foi abordado. O sistema é alimentando com uma matriz de similaridade dos dados, aprendendo os padrões do tráfego de pacotes. Três variações foram exploradas para o cálculo dessa similaridade: produto escalar clássico, produto escalar quântico e *swap test*. Ambos os métodos são eficazes em determinar a similaridade de duas amostras de dados, no entanto sua viabilidade deve ser determinada a partir dos testes executados.

Vale reiterar que os modelos híbridos de computação quântica foram executados em simulação e, como é de se esperar, demandaram um custo computacional alto. Esse custo computacional se dá principalmente pela característica intrínseca das máquinas utilizada e que se deseja simular ser diferente. O computador clássico trabalha com *bits* determinísticos enquanto o quântico aproveita a superposição de uma partícula sub-atômica para realizar sua computação. Dessa forma, os *bits* clássicos, por meio de diversas etapas, simulam os *bits* quânticos.

Dessa maneira, uma iteração do sistema clássico é da ordem de 10^{-6} segundos, enquanto os modelos quânticos são da ordem 10^{-2} e 10^{-1} segundos para os *kernels* 1 e 2, respectivamente. Tendo em vista a diferença do custo computacional das implementações e sua justificação, cabe uma extensão da análise para um computador quântico hipotético. A eficiência dos modelos quânticos quanto ao tempo que demandam, até mesmo em uma máquina quântica, não é promissor para nosso estudo. Devido à natureza não-

determinística do computador quântico, um circuito deve ser executado diversas vezes a fim de determinar uma média de "resultados quânticos"e, só assim, determinar a resposta mais provável para determinado *input*. Computações simples como o produto escalar são de ordem desprezível e não apresentam nenhuma vantagem ao executar em computação quântica.

Os resultados das classificações dos modelos foram muito similares. Ambos os modelos foram excelentes em distinguir as janelas de fluxos como anômalas ou benignas. O único caso onde as métricas não atingiram resultado perfeito foi ao executar o modelo Quântico de *Kernel* 1 no segundo dia sob ataque e janela de 30 segundos. Como é possível observar na matriz de confusão 31, referente à esse teste, houve um falso negativo, onde o modelo inferiu a amostra como benigna mas em realidade era anômala. Isso pode se dar pela característica probabilística da implementação quântica.

De maneira geral, as variações clássicas e híbridas dos modelos performaram de forma similar e a eficiência de ambos os modelos em classificar tráfego de rede é excelente. O custo computacional, como discutido, deve ser levado em conta e é justificável tanto para simulação quanto execução em máquina quântica real. A viabilidade da computação quântica para a detecção de intrusão de rede, no entanto, é questionável. É evidente que para outras tarefas como a fatoração de número primos gigantescos a computação quântica se sai melhor. Isso pode ocorrer pela capacidade computacional do paradigma quântico, a supremacia quântica, ou simplesmente pela modelagem algorítmica mais eficiente empregada. De toda forma, a computação quântica tem muito a oferecer tanto na solução de problemas intrinsecamente quânticos quanto clássicos e seu estudo é de alta importância para o estado-da-arte.

REFERÊNCIAS

[1] FOULDS, S.; KENDON, V.; SPILLER, T. The controlled swap test for determining quantum entanglement. *Quantum Science and Technology*, IOP Publishing, v. 6, n. 3, p. 035002, 2021.

[2] MIORANDI, D. et al. Internet of things: Vision, applications and research challenges. *Ad Hoc Networks*, v. 10, n. 7, p. 1497–1516, 2012. ISSN 1570-8705. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1570870512000674>.

[3] WHITMORE, A.; AGARWAL, A.; XU, L. D. The internet of things—a survey of topics and trends. *Information systems frontiers*, Springer, v. 17, p. 261–274, 2015.

[4] ASSIS, M. V. D. et al. Fast defense system against attacks in software defined networks. *IEEE Access*, IEEE, v. 6, p. 69620–69639, 2018.

[5] MASOUDI, R.; GHAFFARI, A. Software defined networks: A survey. *Journal of Network and computer Applications*, Elsevier, v. 67, p. 1–25, 2016.

[6] CARVALHO, L. F. et al. An ecosystem for anomaly detection and mitigation in software-defined networking. *Expert Systems with Applications*, Elsevier, v. 104, p. 121–133, 2018.

[7] LATHA, S.; PRAKASH, S. J. A survey on network attacks and intrusion detection systems. In: *2017 4th International Conference on Advanced Computing and Communication Systems (ICACCS)*. [S.l.: s.n.], 2017. p. 1–7.

[8] DASTRES, R.; SOORI, M. A review in recent development of network threats and security measures. *International Journal of Information Sciences and Computer Engineering*, 2021.

[9] FERNANDES, G. et al. A comprehensive survey on network anomaly detection. *Telecommunication Systems*, Springer, v. 70, p. 447–489, 2019.

[10] PENA, E. H. et al. Anomaly detection using the correlational paraconsistent machine with digital signatures of network segment. *Information Sciences*, Elsevier, v. 420, p. 313–328, 2017.

[11] KHAN, F. A. et al. A novel two-stage deep learning model for efficient network intrusion detection. *IEEE Access*, IEEE, v. 7, p. 30373–30385, 2019.

[12] DONG, B.; WANG, X. Comparison deep learning method to traditional methods using for network intrusion detection. In: *2016 8th IEEE International Conference on Communication Software and Networks (ICCSN)*. [S.l.: s.n.], 2016. p. 581–585.

[13] SHONE, N. et al. A deep learning approach to network intrusion detection. *IEEE Transactions on Emerging Topics in Computational Intelligence*, v. 2, n. 1, p. 41–50, 2018.

[14] WANG, H.; GU, J.; WANG, S. An effective intrusion detection framework based on svm with feature augmentation. *Knowledge-Based Systems*, Elsevier, v. 136, p. 130–139, 2017.

[15] KIM, D. E.; GOFMAN, M. Comparison of shallow and deep neural networks for network intrusion detection. In: IEEE. *2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)*. [S.l.], 2018. p. 204–208.

[16] NOVAES, M. P. et al. Adversarial deep learning approach detection and defense against ddos attacks in sdn environments. *Future Generation Computer Systems*, Elsevier, v. 125, p. 156–167, 2021.

[17] NOVAES, M. P. et al. Long short-term memory and fuzzy logic for anomaly detection and mitigation in software-defined network environment. *IEEE Access*, IEEE, v. 8, p. 83765–83781, 2020.

[18] ASSIS, M. V. de et al. Near real-time security system applied to sdn environments in iot networks using convolutional neural network. *Computers & Electrical Engineering*, Elsevier, v. 86, p. 106738, 2020.

[19] EKERT, A.; JOZSA, R. Quantum computation and shor's factoring algorithm. *Reviews of Modern Physics*, APS, v. 68, n. 3, p. 733, 1996.

[20] National Academies of Sciences, Engineering, and Medicine and others. Quantum computing: progress and prospects. National Academies Press, 2019.

[21] CHOW, J.; DIAL, O.; GAMBETTA, J. *IBM quantum breaks the 100-qubit processor barrier*. IBM, 2022. Disponível em: <https://research.ibm.com/blog/127-qubit-quantum-processor-eagle>.

[22] IBM quantum computing. 2015. Disponível em: <https://www.ibm.com/quantum>.

[23]

[24] QUANTUM computing and systems with Intel Labs: Intel®. Disponível em: <https://www.intel.com/content/www/us/en/research/quantum-computing.html>.

[25] GHEORGHE-POP, I.-D. et al. Quantum devops: Towards reliable and applicable nisq quantum computing. In: *2020 IEEE Globecom Workshops (GC Wkshps*. [S.l.: s.n.], 2020. p. 1–6.

[26] PRESKILL, J. Quantum computing in the nisq era and beyond. *Quantum*, Verein zur Förderung des Open Access Publizierens in den Quantenwissenschaften, v. 2, p. 79, 2018.

[27] GARCÍA, D. P.; CRUZ-BENITO, J.; GARCÍA-PEÑALVO, F. J. Systematic literature review: Quantum machine learning and its applications. *arXiv preprint arXiv:2201.04093*, 2022.

[28] BIAMONTE, J. et al. Quantum machine learning. *Nature*, Nature Publishing Group UK London, v. 549, n. 7671, p. 195–202, 2017.

[29] SCHULD, M.; SINAYSKIY, I.; PETRUCCIONE, F. An introduction to quantum machine learning. *Contemporary Physics*, Taylor & Francis, v. 56, n. 2, p. 172–185, 2015.

[30] PAYARES, E.; MARTÍNEZ-SANTOS, J. C. Quantum machine learning for intrusion detection of distributed denial of service attacks: a comparative overview. *Quantum Computing, Communication, and Simulation*, SPIE, v. 11699, p. 35–43, 2021.

[31] KALININ, M.; KRUNDYSHEV, V. Security intrusion detection using quantum machine learning techniques. *Journal of Computer Virology and Hacking Techniques*, Springer, v. 19, n. 1, p. 125–136, 2023.

[32] LIANG, J.-M. et al. Quantum anomaly detection with density estimation and multivariate gaussian distribution. *Physical Review A*, APS, v. 99, n. 5, p. 052310, 2019.

[33] KREUTZ, D. et al. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, Ieee, v. 103, n. 1, p. 14–76, 2014.

[34] SAHAY, R.; MENG, W.; JENSEN, C. D. The application of software defined networking on securing computer networks: A survey. *Journal of Network and Computer Applications*, Elsevier, v. 131, p. 89–108, 2019.

[35] LI, W.; MENG, W.; KWOK, L. F. A survey on openflow-based software defined networks: Security challenges and countermeasures. *Journal of Network and Computer Applications*, Elsevier, v. 68, p. 126–139, 2016.

[36] TANK, G. et al. Software defined networks: The new norm for networks. *International Journal of Science and Research (IJSR)*, 2017.

[37] PALIWAL, M.; SHRIMANKAR, D.; TEMBHURNE, O. Controllers in sdn: A review report. *IEEE Access*, v. 6, p. 36256–36270, 2018.

[38] THOTTAN, M.; JI, C. Anomaly detection in ip networks. *IEEE Transactions on signal processing*, IEEE, v. 51, n. 8, p. 2191–2204, 2003.

[39] LENT, D. M. B. et al. A gated recurrent unit deep learning model to detect and mitigate distributed denial of service and portscan attacks. *IEEE Access*, IEEE, v. 10, p. 73229–73242, 2022.

[40] ZHANG, T. et al. A survey of network anomaly visualization. *Science China Information Sciences*, Springer, v. 60, p. 1–17, 2017.

[41] CHANDOLA, V.; BANERJEE, A.; KUMAR, V. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, ACM New York, NY, USA, v. 41, n. 3, p. 1–58, 2009.

[42] MALL, R. et al. Stacking ensemble approach for ddos attack detection in software-defined cyber–physical systems. *Computers and Electrical Engineering*, v. 107, p. 108635, 2023. ISSN 0045-7906. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0045790623000605>.

[43] ALDWEESH, A.; DERHAB, A.; EMAM, A. Z. Deep learning approaches for anomaly-based intrusion detection systems: A survey, taxonomy, and open issues. *Knowledge-Based Systems*, Elsevier, v. 189, p. 105124, 2020.

[44] LEE, S.-W. et al. Towards secure intrusion detection systems using deep learning techniques: Comprehensive analysis and review. *Journal of Network and Computer Applications*, Elsevier, v. 187, p. 103111, 2021.

[45] YANG, Z. et al. A systematic literature review of methods and datasets for anomaly-based network intrusion detection. *Computers & Security*, Elsevier, v. 116, p. 102675, 2022.

[46] OTOUM, Y.; NAYAK, A. As-ids: Anomaly and signature based ids for the internet of things. *Journal of Network and Systems Management*, Springer, v. 29, p. 1–26, 2021.

[47] PROENÇA, M. L.; ZARPELÃO, B. B.; MENDES, L. S. Anomaly detection for network servers using digital signature of network segment. In: IEEE. *Advanced Industrial Conference on Telecommunications/Service Assurance with Partial and Intermittent Resources Conference/E-Learning on Telecommunications Workshop (AICT/SAPIR/ELETE'05)*. [S.l.], 2005. p. 290–295.

[48] HAMAMOTO, A. H. et al. Network anomaly detection system using genetic algorithm and fuzzy logic. *Expert Systems with Applications*, Elsevier, v. 92, p. 390–402, 2018.

[49] SCARANTI, G. F. et al. Unsupervised online anomaly detection in software defined network environments. *Expert Systems with Applications*, Elsevier, v. 191, p. 116225, 2022.

[50] CARVALHO, L. F. et al. A novel anomaly detection system to assist network management in sdn environment. In: IEEE. *2017 IEEE international conference on communications (ICC)*. [S.l.], 2017. p. 1–6.

[51] SCARANTI, G. F. et al. Artificial immune systems and fuzzy logic to detect flooding attacks in software-defined networks. *IEEE Access*, IEEE, v. 8, p. 100172–100184, 2020.

[52] ASSIS, M. V. et al. A gru deep learning system against attacks in software defined networks. *Journal of Network and Computer Applications*, Elsevier, v. 177, p. 102942, 2021.

[53] PROENCA JR, MARIO LEMES et al. Digital signature to help network management using flow analysis. *International Journal of Network Management*, Wiley Online Library, v. 26, n. 2, p. 76–94, 2016.

[54] PROENCA JR, MARIO LEMES et al. The hurst parameter for digital signature of network segment. In: SPRINGER. *Telecommunications and Networking-ICT 2004: 11th International Conference on Telecommunications, Fortaleza, Brazil, August 1-6, 2004. Proceedings 11*. [S.l.], 2004. p. 772–781.

[55] ALPAYDIN, E. *Introduction to machine learning*. [S.l.]: MIT press, 2020.

[56] MOHAMMED, M.; KHAN, M. B.; BASHIER, E. B. M. *Machine learning: algorithms and applications*. [S.l.]: Crc Press, 2016.

[57] TABAK, M. A. et al. Machine learning to classify animal species in camera trap images: Applications in ecology. *Methods in Ecology and Evolution*, Wiley Online Library, v. 10, n. 4, p. 585–590, 2019.

[58] HATCHER, W. G.; YU, W. A survey of deep learning: Platforms, applications and emerging research trends. *IEEE Access*, IEEE, v. 6, p. 24411–24432, 2018.

[59] XU, L.; WHITE, M.; SCHUURMANS, D. Optimal reverse prediction: A unified perspective on supervised, unsupervised and semi-supervised learning. In: *Proceedings of the 26th Annual International Conference on Machine Learning*. New York, NY, USA: Association for Computing Machinery, 2009. (ICML '09), p. 1137–1144. ISBN 9781605585161. Disponível em: <https://doi.org/10.1145/1553374.1553519>.

[60] SULTANA, N. et al. Survey on sdn based network intrusion detection system using machine learning approaches. *Peer-to-Peer Networking and Applications*, Springer, v. 12, p. 493–501, 2019.

[61] XIN, Y. et al. Machine learning and deep learning methods for cybersecurity. *Ieee access*, IEEE, v. 6, p. 35365–35381, 2018.

[62] XIE, J. et al. A survey of machine learning techniques applied to software defined networking (sdn): Research issues and challenges. *IEEE Communications Surveys & Tutorials*, IEEE, v. 21, n. 1, p. 393–430, 2018.

[63] BAKSHI, S. et al. Enhancing scalability and accuracy of recommendation systems using unsupervised learning and particle swarm optimization. *Applied Soft Computing*, Elsevier, v. 15, p. 21–29, 2014.

[64] OSUNA, E. E. *Support vector machines: Training and applications*. Tese (Doutorado) — Massachusetts Institute of Technology, 1998.

[65] YIN, X.-C. et al. Shallow classification or deep learning: An experimental study. In: IEEE. *2014 22nd International Conference on Pattern Recognition*. [S.l.], 2014. p. 1904–1909.

[66] JANABI, A. H.; KANAKIS, T.; JOHNSON, M. Convolutional neural network based algorithm for early warning proactive system security in software defined networks. *IEEE Access*, v. 10, p. 14301–14310, 2022.

[67] AHMAD, I.; WAN, Z.; AHMAD, A. A big data analytics for ddos attack detection using optimized ensemble framework in internet of things. *Internet of Things*, v. 23, p. 100825, 2023. ISSN 2542-6605. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2542660523001488>.

[68] DING, S.; ZHU, Z.; ZHANG, X. An overview on semi-supervised support vector machine. *Neural Computing and Applications*, Springer, v. 28, p. 969–978, 2017.

[69] HUANG, C.; DAVIS, L.; TOWNSHEND, J. An assessment of support vector machines for land cover classification. *International Journal of remote sensing*, Taylor & Francis, v. 23, n. 4, p. 725–749, 2002.

[70] MAMMONE, A.; TURCHI, M.; CRISTIANINI, N. Support vector machines. *Wiley Interdisciplinary Reviews: Computational Statistics*, Wiley Online Library, v. 1, n. 3, p. 283–289, 2009.

[71] OSUNA, E.; FREUND, R.; GIROSIT, F. Training support vector machines: an application to face detection. In: IEEE. *Proceedings of IEEE computer society conference on computer vision and pattern recognition*. [S.l.], 1997. p. 130–136.

[72] KALITA, D. J.; SINGH, V. P.; KUMAR, V. A survey on svm hyper-parameters optimization techniques. In: SPRINGER. *Social Networking and Computational Intelligence: Proceedings of SCI-2018*. [S.l.], 2020. p. 243–256.

[73] BURGES, C. J. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, Springer, v. 2, n. 2, p. 121–167, 1998.

[74] SHRESTHA, A.; MAHMOOD, A. Review of deep learning algorithms and architectures. *IEEE access*, IEEE, v. 7, p. 53040–53065, 2019.

[75] BORISOV, V. et al. Deep neural networks and tabular data: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, p. 1–21, 2022.

[76] GOODFELLOW, I. et al. Generative adversarial nets. *Advances in neural information processing systems*, v. 27, 2014.

[77] YU, B.; WEI, H.; WANG, W. Gan-based day and night image cross-domain conversion research and application. In: *2022 11th International Conference of Information and Communication Technology (ICTech)*. [S.l.: s.n.], 2022. p. 230–235.

[78] ZHANG, H. et al. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In: *Proceedings of the IEEE international conference on computer vision*. [S.l.: s.n.], 2017. p. 5907–5915.

[79] LEE, C.-K.; CHEON, Y.-J.; HWANG, W.-Y. Studies on the gan-based anomaly detection methods for the time series data. *IEEE Access*, IEEE, v. 9, p. 73201–73215, 2021.

[80] SCHLEGL, T. et al. f-anogan: Fast unsupervised anomaly detection with generative adversarial networks. *Medical image analysis*, Elsevier, v. 54, p. 30–44, 2019.

[81] LI, Y. et al. The theoretical research of generative adversarial networks: an overview. *Neurocomputing*, Elsevier, v. 435, p. 26–41, 2021.

[82] NAVIDAN, H. et al. Generative adversarial networks (gans) in networking: A comprehensive survey & evaluation. *Computer Networks*, Elsevier, v. 194, p. 108149, 2021.

[83] SAJEEDA, A.; HOSSAIN, B. M. Exploring generative adversarial networks and adversarial training. *International Journal of Cognitive Computing in Engineering*, Elsevier, v. 3, p. 78–89, 2022.

[84] REBENTROST, P.; MOHSENI, M.; LLOYD, S. Quantum support vector machine for big data classification. *Physical review letters*, APS, v. 113, n. 13, p. 130503, 2014.

[85] LIU, N.; REBENTROST, P. Quantum machine learning for quantum anomaly detection. *Physical Review A*, APS, v. 97, n. 4, p. 042315, 2018.

[86] GOUVEIA, A.; CORREIA, M. Towards quantum-enhanced machine learning for network intrusion detection. In: *2020 IEEE 19th International Symposium on Network Computing and Applications (NCA)*. [S.l.: s.n.], 2020. p. 1–8.

[87] HUBREGTSEN, T. et al. Training quantum embedding kernels on near-term quantum computers. *Phys. Rev. A*, American Physical Society, v. 106, p. 042431, Oct 2022. Disponível em: <https://link.aps.org/doi/10.1103/PhysRevA.106.042431>.

[88] M., G.; SETHURAMAN, S. C. A comprehensive survey on deep learning based malware detection techniques. *Computer Science Review*, v. 47, p. 100529, 2023. ISSN 1574-0137. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1574013722000636>.

[89] GONG, C. et al. Network intrusion detection based on variational quantum convolution neural network. *The Journal of Supercomputing*, Springer, p. 1–28, 2024.

[90] KUKLIANSKY, A. et al. Network anomaly detection using quantum neural networks on noisy quantum computers. *IEEE Transactions on Quantum Engineering*, IEEE, 2024.

[91] MOORE, G. E. Cramming more components onto integrated circuits. *Proceedings of the IEEE*, Ieee, v. 86, n. 1, p. 82–85, 1998.

[92] MOORE, G. Moore's law. *Electronics Magazine*, v. 38, n. 8, p. 114, 1965.

[93] YODER, M.; ORSAK, G. Engineering: our digital future. 2004.

[94] BUSCH, P.; HEINONEN, T.; LAHTI, P. Heisenberg's uncertainty principle. *Physics reports*, Elsevier, v. 452, n. 6, p. 155–176, 2007.

[95] BERNSTEIN, E.; VAZIRANI, U. Quantum complexity theory. In: *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*. [S.l.: s.n.], 1993. p. 11–20.

[96] BEAUREGARD, S. Circuit for shor's algorithm using 2n+ 3 qubits. *arXiv preprint quant-ph/0205095*, 2002.

[97] HEY, T. Quantum computing: an introduction. *Computing & Control Engineering Journal*, IET, v. 10, n. 3, p. 105–112, 1999.

[98] AMIRI, P. K. Quantum computers. *IEEE Potentials*, IEEE, v. 21, n. 5, p. 6–9, 2003.

[99] DEUTSCH, D. Quantum theory, the church–turing principle and the universal quantum computer. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, The Royal Society London, v. 400, n. 1818, p. 97–117, 1985.

[100] BHARTI, K. et al. Noisy intermediate-scale quantum algorithms. *Reviews of Modern Physics*, APS, v. 94, n. 1, p. 015004, 2022.

[101] NIELSEN, M. A.; CHUANG, I. L. Quantum computation and quantum information. *Phys. Today*, v. 54, n. 2, p. 60, 2001.

[102] TERHAL, B. M. Quantum error correction for quantum memories. *Reviews of Modern Physics*, APS, v. 87, n. 2, p. 307, 2015.

[103] LALOë, F. Quantum entanglement. In: _____. *Do We Really Understand Quantum Mechanics?* 2. ed. [S.l.]: Cambridge University Press, 2019. p. 189–222.

[104] SAVCHUK, M.; FESENKO, A. Quantum computing: Survey and analysis. *Cybernetics and Systems Analysis*, Springer, v. 55, p. 10–21, 2019.

[105] CIRQ : google quantum ai. Disponível em: <https://quantumai.google/cirq>.

[106] ERNST, D. A. *An overview of quantum computing frameworks — ginkgo-analytics.com.* <https://www.ginkgo-analytics.com/an-overview-of-quantum-computing-frameworks/>.

[107] XANADU | Welcome to Xanadu — xanadu.ai. <https://www.xanadu.ai/>.

[108] HTTPS://WWW.FACEBOOK.COM/48576411181. *Rigetti Launches Full-Stack Quantum Computing Service and Quantum IC Fab — spectrum.ieee.org.* <https://spectrum.ieee.org/rigetti-launches-fullstack-quantum-computing-service-and-quantum-ic-fab>.

[109] PRICING — strangeworks.com. <https://strangeworks.com/pricing>.

[110] SOUZA, P. J. et al. Computação quântica adiabática: Do teorema adiabático ao computador da d-wave. *Revista Brasileira de Ensino de Física*, SciELO Brasil, v. 43, 2021.

[111] CHO, A. *Quantum or not, controversial computer yields no speedup.* [S.l.]: American Association for the Advancement of Science, 2014.

[112] GOOGLE : Quantum computing hardware. Disponível em: <https://quantumai.google/hardware>.

[113] RAUSSENDORF, R. Measurement-based quantum computation with cluster states. *International Journal of Quantum Information*, World Scientific, v. 7, n. 06, p. 1053–1203, 2009.

[114] AMBAINIS, A.; REGEV, O. An elementary proof of the quantum adiabatic theorem. *arXiv preprint quant-ph/0411152*, 2004.

[115] DAS, A.; CHAKRABARTI, B. K. Colloquium: Quantum annealing and analog quantum computation. *Reviews of Modern Physics*, APS, v. 80, n. 3, p. 1061, 2008.

[116] AHARONOV, D. et al. Adiabatic quantum computation is equivalent to standard quantum computation. *SIAM review*, SIAM, v. 50, n. 4, p. 755–787, 2008.

[117] NAYAK, C. et al. Non-abelian anyons and topological quantum computation. *Reviews of Modern Physics*, APS, v. 80, n. 3, p. 1083, 2008.

[118] GRIFFITHS, D. J.; SCHROETER, D. F. *Introduction to quantum mechanics.* [S.l.]: Cambridge university press, 2018.

[119] HARVEY, S. P. *Quantum Dots/Spin Qubits.* Oxford University Press, 2022. Disponível em: <http://dx.doi.org/10.1093/acrefore/9780190871994.013.83>.

[120] BERNHARDT, C. *Quantum computing for everyone.* [S.l.]: Mit Press, 2019.

[121] NIELSEN, M. A.; CHUANG, I. L. *Quantum computation and quantum information*. [S.l.]: Cambridge university press, 2010.

[122] FLAREND, A.; HILBORN, R.; HILBORN, R. *Quantum Computing: from Alice to Bob*. Oxford University Press, 2022. ISBN 9780192857989. Disponível em: <https://books.google.com.br/books?id=mGTmzgEACAAJ>.

[123] FANIZZA, M. et al. Beyond the swap test: optimal estimation of quantum state overlap. *Physical review letters*, APS, v. 124, n. 6, p. 060503, 2020.

[124] DRAGNE, L. Modeling the controlled swap gate with quantum circuits. *Annals of DAAAM & Proceedings*, DAAAM International Vienna, p. 1713–1715, 2009.

[125] SHANNON, C. E. A mathematical theory of communication. *The Bell system technical journal*, Nokia Bell Labs, v. 27, n. 3, p. 379–423, 1948.

[126] PEDREGOSA, F. et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, JMLR. org, v. 12, p. 2825–2830, 2011.

[127] JAYASWAL, V. Performance metrics: Confusion matrix, precision, recall, and f1 score. *Medium, Towards Data Science*, v. 15, 2020.

[128] CHICCO, D.; TÖTSCH, N.; JURMAN, G. The matthews correlation coefficient (mcc) is more reliable than balanced accuracy, bookmaker informedness, and markedness in two-class confusion matrix evaluation. *BioData mining*, BioMed Central, v. 14, n. 1, p. 1–22, 2021.