Evaluating the Efficiency of Quantum Simulators using Practical Application Benchmarks

by

Rajesh Sathya Kumar

A Thesis Presented in Partial Fulfillment
of the Requirement for the Degree
Master of Science

Approved April 2023 by the
Graduate Supervisory Committee:

Andreas Spanias, Co-Chair
Arunabha Sen, Co-Chair
Gautam Dasarathy

ARIZONA STATE UNIVERSITY

May 2023

ABSTRACT

Quantum computing holds the potential to revolutionize various industries by solving problems that classical computers cannot solve efficiently. However, building quantum computers is still in its infancy, and simulators are currently the best available option to explore the potential of quantum computing. Therefore, developing comprehensive benchmarking suites for quantum computing simulators is essential to evaluate their performance and guide the development of future quantum algorithms and hardware.

This study presents a systematic evaluation of quantum computing simulators' performance using a benchmarking suite. The benchmarking suite is designed to meet the industry-standard performance benchmarks established by the Defense Advanced Research Projects Agency (DARPA) and includes standardized test data and comparison metrics that encompass a wide range of applications, deep neural network models, and optimization techniques. The thesis is divided into two parts to cover basic quantum algorithms and variational quantum algorithms for practical machine learning tasks.

In the first part, the run time and memory performance of quantum computing simulators are analyzed using basic quantum algorithms. The performance is evaluated using standardized test data and comparison metrics that cover fundamental quantum algorithms, including Quantum Fourier Transform (QFT), Inverse Quantum Fourier Transform (IQFT), Quantum Adder, and Variational Quantum Eigensolver (VQE). The analysis provides valuable insights into the simulators' strengths and weaknesses and highlights the need for further development to enhance their performance. In the second part, benchmarks are developed using variational quantum

algorithms for practical machine learning tasks such as image classification, natural language processing, and recommendation. The benchmarks address several unique challenges posed by benchmarking quantum machine learning (QML), including the effect of optimizations on time-to-solution, the stochastic nature of training, the inclusion of hybrid quantum-classical layers, and the diversity of software and hardware systems. The findings offer valuable insights into the simulators' ability to solve practical machine learning tasks and pinpoint areas for future research and enhancement.

In conclusion, this study provides a rigorous evaluation of quantum computing simulators' performance using a benchmarking suite that meets industry-standard performance benchmarks.

ACKNOWLEDGEMENTS

I am immensely grateful to my advisor, Dr. Andreas Spanias, for allowing me the opportunity to pursue my Master's Thesis under his guidance. I would also like to extend my appreciation to my committee members, Dr. Arunabha Sen and Dr. Gautam Dasarathy, for their valuable suggestions and comments. I am also grateful to my colleagues at SenSIP, particularly Glen Uehara, and other lab members for their support and constructive criticism. My sincere thanks to Dr. Gil Speyer for sharing his technical expertise in the area and steering me in the right direction. I would like to thank my family, especially my Mom and my Brother, for having faith in me and enabling me to pursue my masters without any hindrance. Finally, I want to express my heartfelt thanks to all my friends for their unwavering support throughout the journey, making it a memorable experience.

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

Quantum computing simulators are software programs that mimic the behavior of quantum algorithms on classical computers. They simulate the same mathematical operations of a quantum computer at an exponential computational cost, which means it is only possible to run quantum operations up to a certain number of qubits in these simulators. Despite this limitation, they are powerful tools that offer many benefits such as:

- Introspect the quantum states at different levels of computation.
- Easily accessible with a lot of open-source implementations from companies like Amazon, IonQ, Azure, Etc., which offers software SDK's and cloud access.
- Supports state vector, density matrices, and tensor network representations.
- Able to model quantum noise and decoherence for near-term algorithms.
- Provide easy way for researchers to experiment with old and novel algorithms.

However, due to the exponential increase in computational resources required by quantum state vector and density matrix simulators with the number of qubits or circuit width, it is challenging to optimize these quantum simulators beyond 40 qubits (Efthymiou *et al.* (2022)). Therefore, it is crucial to establish benchmarks (Coleman *et al.* (2019)) and metrics to evaluate the performance of these simulators effectively and identify bottlenecks. The proliferation of quantum computing hardware and software systems highlights the pressing need for a rigorous benchmarking framework.

Standard benchmarks have been proven to be instrumental in advancing the development of technology in various fields such as microprocessors and relational databases. The development of standard benchmarks for quantum computation was inspired by the success of organizations such as the Standard Performance Evaluation Corporation (SPEC)(Dixit (1991)) and the Transaction Processing Performance Council (TPC)(TPC (2010)).

Quantum computing has a vast array of potential applications, making it essential to evaluate the hardware and simulators in accordance with specific goals. This requires the development of a comprehensive suite of application benchmarks for a diverse range of use cases. The Quantum Economic Development Consortium (QEDC) has developed practical benchmarks and metrics for actual quantum computers, following a similar application-oriented approach. Atos, a company specializing in digital transformation, has developed a method to assess the effectiveness of quantum systems in solving optimization problems (Martiel *et al.* (2021)). These benchmarking techniques allow researchers to evaluate how well quantum computers perform relative to classical computers in solving specific problems. By comparing the results of the benchmarking, researchers can determine the potential impact of quantum computing on various fields such as finance, healthcare, and logistics. Meanwhile, a quantum software start-up, Super.tech, has introduced SupermarQ (Tomesh *et al.* (2022)), a suite of practical application benchmarks that includes error correction benchmarks. These benchmarks serve as a measure of the readiness of quantum computers to solve practical problems, indicating progress toward achieving fault tolerance. By developing practical benchmarks and incorporating error correction measures, Super.tech aims to accelerate the development of quantum computing applications. The ability to solve practical problems will be a crucial factor in determining the success of quan-

tum computing in the future. The benchmarks developed by Atos and Super.tech will play a critical role in evaluating the progress of quantum computing and its potential impact on various industries.

While these benchmarks computers are designed for real Quantum Computers, some of these techniques apply to quantum computing simulators as well. We will discuss the benchmarks that were developed during this study and their results in the later sections. In order to perform these simulations, we make use of a selected list of quantum software packages and we call them S1, S2, S3, and S4. Several quantum software packages support simulators with the emergence of quantum computation. Each of these simulators supports different mathematical representations, such as state vectors, density matrices, and tensor networks, which are widely used network is the Matrix Product State (MPS). Table 1.1 shows the list of quantum simulators supported by these software packages and their support for GPU execution and NVIDIA's quantum computing accelerator cuQuantum (cuq (2022)).

In the first section of the experiments, we conducted a comparative analysis of the different types of simulators supported by each software package by profiling the simulations against specific practical benchmarking algorithms. Three types of benchmarks were used in this paper: random square circuits, fundamental quantum algorithms, and Variational Quantum Algorithms (VQA) (Cerezo *et al.* (2020)). The fundamental quantum algorithms used in this paper are Quantum Fourier Transform (QFT), Inverse Quantum Fourier Transform (IQFT), and Quantum Adders. VQE and QNN are used as benchmarks for variational algorithms. Then, we evaluated the performance of these state-of-the-art simulators concerning their simulation time (statistically weighted). We also compared their performance when simulated on a

Quantum Software	Available Simulators	GPU	cuQuantum
Qiskit (IBM)	State Vector	✓	✓
	Density Matrix	✓	✓
	MPS	×	×
Cirq (Google)	State Vector	×	×
	Density Matrix	×	×
	qsim/qsimh	✓	✓
	Quimb	✓	×
Braket (Amazon)	State Vector	×	×
	Density Matrix	×	×
Pennylane (Xanadu)	State Vector	✓	✓
	Density Matrix	×	×

Table 1.1: Simulator Support in Quantum Software Frameworks (By the time of this study)

GPU or using the quantum computing accelerator cuQuantum. We will also discuss the observations from the results.

The second part of the experiments involved comparing different simulators for Quantum ML (QML) applications. However, developing benchmarks for QML poses unique challenges compared to classical ML, specifically with Deep Learning (DL) due to the various optimizations that can affect the final model (Auer *et al.* (1995), Gori and Tesi (1992), Choromanska *et al.* (2014)). DL is also stochastic, with multiple valid solutions, so clear definitions of solution classes are necessary for consistent results. Optimization of DL performance is further complicated by hyperparame-

ter adjustments and distributed systems. These challenges were addressed in this study, which developed a benchmarking suite with metrics, timing rules, and reference implementations for QML. This benchmarking suite was inspired by the widely used MLPerf benchmarking suite for classical ML. The results from popular quantum software simulators were presented, and future directions for benchmarking were discussed in the conclusion.

Chapter 2

RELATED WORK

Performance benchmarking is an essential process in evaluating various deep technologies. It helps in comparing the strengths of different platforms or systems and speeding up the development of quantum computing. In this process, designing practical, scalable, and comprehensive benchmarks is crucial for a radical and complex technology like quantum computing. Some firms are already making progress in developing these benchmarks, and more tools will likely emerge as the technology gets closer to the market. There are two types of benchmarks being developed for quantum computers: system and application benchmarks. System benchmarks focus on the hardware and its ability to execute basic operations. On the other hand, application benchmarks focus on the performance of quantum algorithms on real-world problems. These benchmarks can inform users about what they need to know and help them grow and adapt to emerging technologies.

In our goal to benchmark quantum computing simulators, we can adapt application benchmarks to evaluate the performance of quantum algorithms. These benchmarks can be used to assess the performance of quantum algorithms on a variety of tasks, such as optimization, machine learning, and cryptography. However, there are challenges in developing benchmarks for ML tasks using quantum computing and these challenges are discussed in this chapter. In classical machine learning, MLPerf is the de facto performance standard. These tests evaluate how well graphics processing units perform on particular tasks, such as speech recognition or picture classification, using a collection of medical data. We will also discuss how we can adapt these bench-

marks for benchmarking machine learning tasks for quantum simulators and review the results in the later chapters.

2.1 System Benchmarks

Quantum computers are evaluated through system benchmarks that measure essential properties such as quality, speed, and scale, which are initially proposed by Wack *et al.* (2021), a group of researchers at IBM. They provide several system benchmarks, including Quantum Volume (QV) and Circuit Layer Operations per Second (CLOPS).

Definition 2.1.1 *Quantum Volume evaluates the quality of a system by measuring the largest square quantum circuit that a Quantum Computer can successfully implement for a given number of Qubits N and depth d(N). It is given by,*

$$V_Q = min[N, d(N)]$$

Another useful metric is the Algorithmic Quantum Volume.

Definition 2.1.2 *Algorithmic Quantum Volume specifies the number of qubits that are useful in successfully implementing a quantum circuit. It is given by,*

$$log_2 V_Q = \arg\max_{n<=N} \{min[n, d(n)]\}$$

CLOPS is another unit of measurement used to describe speed, first used in late 2021.

Definition 2.1.3 *Circuit Layer Operations per Second (CLOPS) indicates the number of Quantum Circuits that can be executed on a given hardware and time.*

These metrics offer a transparent history of development and allow evaluations of machine performance independent of specific use cases.

One benchmark for computational power is Shor's algorithm, which has inherent complexity and is frequently used to evaluate quantum computing. The best estimate predicts an eight-hour runtime for superconducting qubits, while trapped ion qubits, which have a lower CLOPS value, are estimated to take one year. This highlights the significant differences between alternative quantum computing technologies.

IBM has introduced the Quantum Volume metric, which measures the performance of a quantum computer by taking into account both the size of the system and the complexity of the quantum circuits that it can implement. While these benchmarks are useful for comparing the relative performance of different quantum computers, it is not definitive for quantum computing as a whole, and researchers are considering other potential metrics to better measure the performance of quantum computers.

2.2 Application Benchmarks

Application benchmarks play a critical role in assessing the performance of quantum computers in specific use cases. They enable end-users and investors to evaluate the system's overall performance, rather than just assessing quantum advantage. Although the development of such benchmarks is challenging, it provides significant value in measuring performance based on real-world use cases. Several attempts have been made in the past to develop quantum benchmarking suites (LaRose (2019)), including the Quantum LINPACK, Q-Score and application-oriented benchmarking

suite from QED-C.

2.2.1 Initial Attempts

One of the initial attempts at developing a quantum benchmarking suite was by the Berkeley Lab, where they replicated the popular classical computing benchmarking suite, Linear Algebra Package (LINPACK). However, instead of using classical computing, they developed a set of linear algebra operations using quantum circuits to measure the efficacy of quantum computing. Another team from Atos (Martiel *et al.* (2021)) developed Q-Score, a metric to measure the performance of standard optimization problems in quantum computers by assessing the maximum number of variables that can be optimized using the hardware. While these benchmarks are a good start, they only predict the efficacy of quantum computers for specific applications. Therefore, there is a need to develop more sophisticated benchmarks that consider a wider range of applications. This will help in determining the strengths and weaknesses of different quantum computers and selecting the one that best meets the specific needs of the user. SupermarQ is a novel set of application benchmarks developed by Tomesh *et al.* (2022), researchers from the quantum software start-up Super.tech. These benchmarks adopt traditional benchmarking methods and adapt them to the quantum world. SupermarQ employs four design principles, namely scalability, meaningfulness and diversity, full-system evaluation, and adaptivity, to select and evaluate applications. The applications cover a broad range of algorithmic techniques that are relevant to real-world use cases in various industries, such as finance, pharmaceuticals, energy, and chemistry. Most of the current use cases are covered by these algorithms. In addition, SupermarQ includes a benchmark for error correction, which serves as a proxy for practical readiness and indicates the extent to which a

device needs to advance toward fault tolerance. The QED-C's benchmarking suite forms the basis for the experiments of this thesis.

2.2.2 QED-C's Application-Oriented Benchmarks

The US Quantum Economic Development Consortium (QED-C) has created an open-source benchmarking suite, developed by Lubinski *et al.* (2021), consisting of quantum algorithms to assess practical applications of quantum computing. The QED-C was formed after the National Quantum Initiative Act of 2018 was passed. The quantum algorithms used in the benchmarking suite are classified into three types: Tutorial, Subroutine, and Functional. These algorithms are implemented using various quantum software frameworks like Qiskit, Cirq, and Braket. The benchmarking suite uses four metrics to evaluate the performance of the algorithms: Circuit Depth, Creation Time, Execution Time, and Fidelity.

Definition 2.2.1 ***Circuit Depth*** *refers to the length of the longest path from input to output, or the minimum amount of time taken to execute the circuit, assuming every gate operation is performed in the same time step.*

Definition 2.2.2 ***Transpiled Depth*** *indicates the depth of the circuit after transpilation using the supported gates of the quantum hardware.*

Definition 2.2.3 ***Circuit Creation Time*** *measures the time taken by the transpiler to create the quantum circuit.*

Definition 2.2.4 ***Circuit Execution Time*** *measures the time taken by the Quantum Backend to evaluate the circuit.*

Definition 2.2.5 ***Circuit Fidelity*** *is a metric that determines the closeness of two quantum states in a system. It is given by,*

$$F(\rho, \sigma) = tr\sqrt{\rho^{1/2}\sigma\rho^{1/2}}$$

where ρ and σ are Density Operators, which is a generalization of the Ket vector representation of the Wavefunction. Density operators are useful in representing both Pure States as well as Mixed States.

The benchmarking suite is designed to evaluate the practicality of quantum algorithms in various fields like chemistry, finance, logistics, and cybersecurity. Figure 2.1 shows the result of simulating a Quantum Fourier Transform (QFT) using state vector simulators of the software framework, S2. Figure 2.2 shows the fidelity of the circuit as we increase the circuit width and circuit depth in simulating QFT in an actual quantum computer**ibmq_quito**. The result indicates that the machine is capable of simulating a QFT up to only 4 Qubits. The fidelity of the circuit increases and does not yield the desired result if we increase the number of qubits beyond 4. This technique is called **Volumetric Benchmarking**, which is a work produced by Miller *et al.* (2022).

2.3 MLPerf

MLPerf is a consortium of academic institutions and technology companies that aim to establish a standardized set of benchmarks to evaluate the performance of machine learning (ML) systems and was initially developed by Mattson *et al.* (2019). The organization was founded in 2018 and has since developed a set of benchmarks for training and inference tasks. The benchmarks are designed to evaluate the performance of ML systems across different hardware platforms, software frameworks, and optimization techniques.

Benchmarking ML applications poses several challenges. Some of these challenges are:

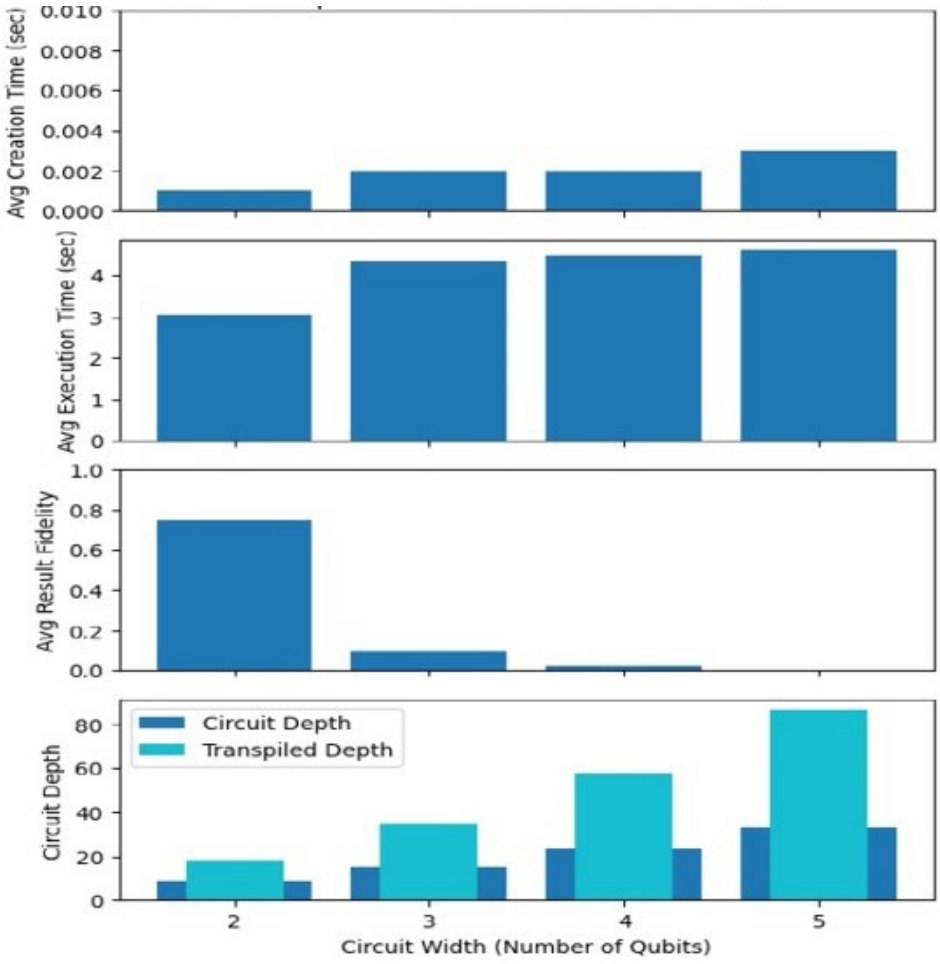

Figure 2.1: QED-C's Benchmarking Results: Depth, Creation Time, Execution Time, and Fidelity.

- **Impact of optimization:** Different optimization techniques can significantly impact the performance of an ML system (Zhu *et al.* (2016)). A benchmark must account for these optimization techniques to provide a fair evaluation.

- **Impact of distributed systems:** Distributed systems can be challenging to benchmark, as the performance can vary depending on the network configuration and communication protocols used (Strubell *et al.* (2019), Goyal *et al.* (2017)).

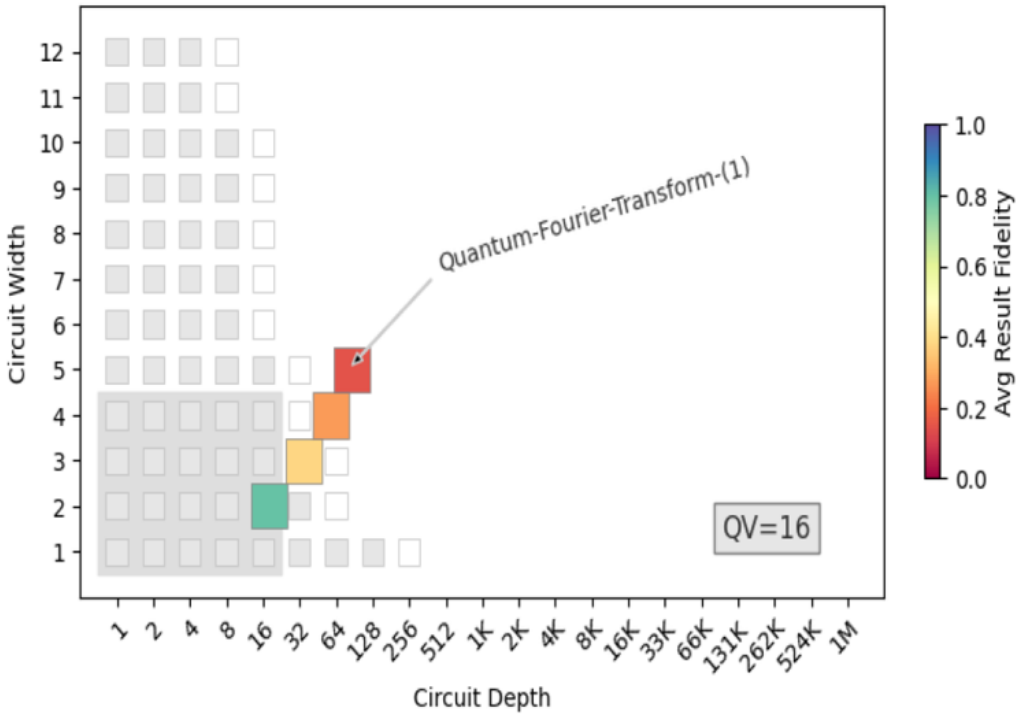

Figure 2.2: QED-C's Benchmarking Results: Volumetric Benchmarking.

- **Stochastic manifestations in QML:** In quantum machine learning, randomness is an inherent feature of the algorithms. This randomness can lead to variability in the results, making benchmarking more challenging.

- **Different software implementations:** ML systems can be implemented in various software frameworks, each with its own optimizations and trade-offs. Benchmarking should account for these differences to provide an accurate evaluation.

- **Involvement of both classical and quantum layers:** In quantum machine learning, classical preprocessing and postprocessing steps (Beer *et al.* (2020)) are often required. These classical steps can affect the performance of the overall system, making it more challenging to benchmark.

MLPerf has developed benchmarks for both training and inference tasks. Training benchmarks measure the time it takes to train a model on a given dataset. Inference benchmarks measure the time it takes to run a trained model on a new dataset. MLPerf provides several benchmarks and metrics to evaluate the performance of an ML system, including:

- **Image classification:** The benchmark evaluates the ability of an ML system to classify images into specific categories.
- **Object detection:** The benchmark evaluates the ability of an ML system to detect objects in images and videos.
- **Translation:** The benchmark evaluates the ability of an ML system to translate text from one language to another.
- **Speech recognition:** The benchmark evaluates the ability of an ML system to recognize speech in different languages.

In this second part of the study (described in later chapters), we have developed quantum benchmarks for practical ML tasks such as Image classification using the ImageNet dataset (Krizhevsky *et al.* (2017), Deng *et al.* (2009), Goyal *et al.* (2017)), NLP using the Wikipedia dataset (NLP (2018), He *et al.* (2017), Devlin *et al.* (2019),Brown *et al.* (2020)), and Recommendation systems using the MovieLens dataset (Cheng *et al.* (2016)).

Chapter 3

PROBLEM DEFINITION AND RESEARCH FOCI

This study focuses on developing a benchmarking suite to evaluate the performance of some quantum computing simulators (We named them S1, S2, S3, and S4). The goal is to develop a comprehensive benchmark suite that encompasses various applications and follows coding principles to enable ease of experimentation and adaptability for novel and near-term quantum algorithms and systems. To achieve this, the study has established timing guidelines to reduce the impact of randomness during comparisons, developed precise reference implementations, and made baseline implementations available to the public for examination, replication, and comparison by the quantum machine learning and systems communities.

The study evaluates the performance of these state-of-the-art simulators with respect to their simulation time, statistically weighted, and shows their performance when simulated on a CPU and a GPU. The study follows the code design principles of scalability with respect to parameters, practical and diverse algorithms, full system evaluation, adaptability, and enabling fair comparison metrics to ensure that the benchmarking suite is comprehensive and accurate. In summary, the goals of the study are as follows:

- Develop a comprehensive benchmark suite that is diverse.
- Develop reference implementations of the algorithms and models.
- Establish timing guidelines to reduce the impact of randomness.

- Make baseline implementations available to the public to allow for examination, replication, and comparison by the quantum machine learning and systems communities.
- Follow coding principles to enable ease of experimentation and adaptability for novel and near-term quantum algorithms and systems.

3.1 Suite of Fundamental Quantum Algorithms

In the first part of the study, we compare the performance of different types of quantum simulators supported by the above-mentioned software packages, including state vectors, density matrices, and tensor networks such as the Matrix Product State (MPS). The analysis is based on profiling the simulations against certain practical benchmarking algorithms, including random square circuits, fundamental quantum algorithms such as Quantum Fourier Transform (QFT), Inverse Quantum Fourier Transform (IQFT), and quantum adders, as well as Variational Quantum Algorithms (VQA) as described by Cerezo *et al.* (2020), such as Variational Quantum Eigensolver (VQE) and Quantum Neural Networks (QNN).

3.2 Quantum Benchmarking of Practical ML Tasks

This study presents a comprehensive quantum computing benchmark suite that focuses on practical machine learning (ML) workloads. The suite includes a carefully curated set of tasks from key areas such as vision (Sandler *et al.* (2018), He *et al.* (2015), Deng *et al.* (2009)), language, and recommendation, which were selected based on their commercial and research significance. The benchmark suite features a compact, yet representative set of three benchmarks and includes source code that will be made open-source to engage the QML Benchmarking community.

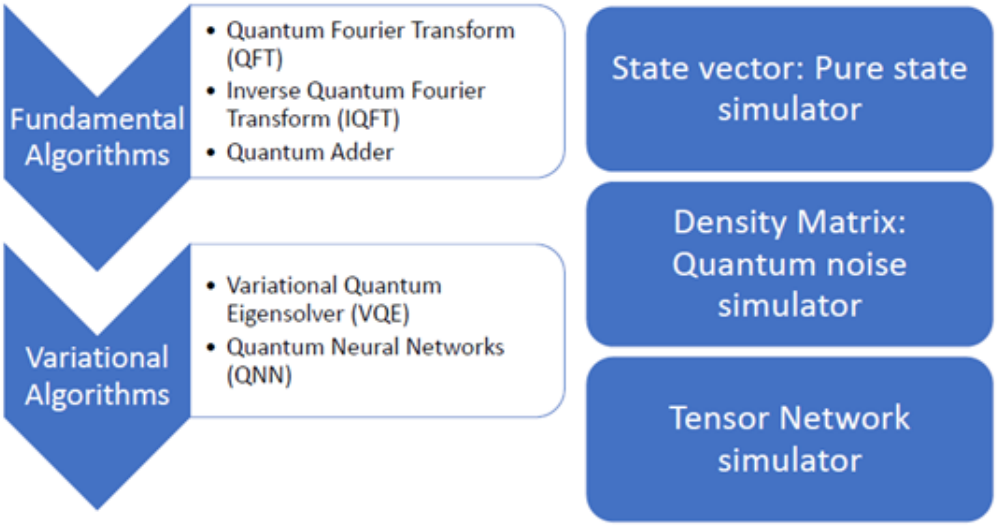

Figure 3.1: Benchmarks Using Quantum Algorithms.

The benchmarks featured in the suite are described in Table 1, along with the datasets, models, and metrics used in their implementation. The study also addresses common issues related to benchmarking QML applications, such as the effect of optimizations, distributed systems, different software implementations, classical and hybrid layers, and more.

Overall, this benchmarking suite aims to provide a valuable resource for the QML community, enabling researchers and practitioners to evaluate the performance of quantum computing platforms on practical ML workloads. By addressing common issues related to benchmarking QML applications, the study seeks to ensure that the benchmarking results are accurate and reliable and that they provide meaningful insights into the capabilities and limitations of quantum computing platforms for practical ML applications.

Benchmark	**Dataset**	**Quality Target**	**Reference Model**
Image classification	ImageNet	74.9% Top-1 accuracy	ResNet-50 + Quantum Layers
NLP	Wikipedia 2020/01/01	0.72 Mask-LM accuracy	BERT-large + Quantum Layers
Recommendation	MovieLens-20M	0.635 HR@10	NCF + Quantum Layers

Table 3.1: QML Benchmarking Suite

3.3 Disclaimer

The comparative results in this study were obtained in the Fall of 2022 and spring of 2023. Given that the Quantum simulators examined are constantly evolving, comparisons and concluding remarks may change considerably over time. In addition, results are platform dependent. Hence the presence of additional GPUs or other hardware and software accelerators in certain computing platforms may change significantly the outcomes of the comparative simulations presented in this document. For this reason, hardware and software developers should not use these results without extensive verification across different platforms and with the most current simulators available.

Chapter 4

SIMULATIONS USING FUNDAMENTAL QUANTUM ALGORITHMS

In this section, we will discuss the experimental setup, statistics, and metrics used for the first set of simulations using fundamental algorithms. The results presented in this study are based on experiments that were executed on different systems.

4.1 Experimental Setup

It is important to note that the experiments were conducted using specific software packages, namely S1, S2, S3, and S4. It is essential to keep in mind that new versions of the simulators might yield different results. Therefore, it is crucial to compare the systems on a common ground.

Moreover, the simulation of different systems may lead to varying results since each software may be optimized for the corporation's systems. For example, we found that S2 is optimized for Tesla T4 GPU and may provide an unfair advantage. Hence, it is necessary to run all the quantum simulators on the same machine to compare them fairly.

The benchmarks and code serve as reference implementations, and the results shown in this study are based on the systems mentioned in each experiment. Thus, the results are only useful if the reference implementations are simulated on different quantum simulators that run on the same machine for comparison, even in the future.

4.2 Statistics and Metrics

The statistics and metrics used in this study are based **time.perf_counter()** in python and which has the highest available resolution in python. This tool was used to measure execution and running times. It measures wall clock time with the highest available resolution to measure a short duration accurately. The experiment was repeated 50 times, and the execution times were averaged. The loop was executed over five runs, and the minimum average metric was used for visualization.

The circuit creation time represents the parse time of the simulators from OpenQASM v.2 or v.3. The circuit execution time, on the other hand, represents the circuit simulation time over 1024 shots. Both metrics were measured over the square circuits with depth/width ranging from 2-24.

It is noteworthy that the results may vary if the experiments were performed on a different machine with a GPU or a TPU. Therefore, it is essential to keep in mind the experimental setup when comparing the results obtained in this study to those obtained from other experiments.

System	**Specifications**
System 1	Windows 11 OS, with 11th Gen Intel(R) Core (TM) i7-1165G7 @2.80GHz CPU and 12.0 GB RAM
System 2	Mac OS, with M1 processor @3.2GHz with 16 GB RAM
System 3	Ubuntu 20.04.4, with Intel(R) Xeon(R) CPU @ 2.20GHz and 32 GB RAM.

Table 4.1: Systems Used in the Experiments

Chapter 5

RESULTS USING FUNDAMENTAL QUANTUM ALGORITHMS

In this chapter, we will discuss the results and observations of the experiments performed using the fundamental quantum algorithm suite.

5.1 Experiment 1: System Comparison

We simulated the same randomly generated quantum circuit ("Quantum Volume Circuit", which has the same width and depth) with each of the software (S1, S2, S3, and S4). We simulated each of those circuits using state vector simulators by increasing the qubits from 2-24 in a step of 2. The same simulation was repeated by different systems that are indicated in Chapter 4 and table 4.1. Figures 5.1 show a sample of randomly generated quantum circuits using a quantum simulator. From the result shown in figure 5.2, we can see that for the same circuits, the simulation times vary slightly and increase exponentially as we increase the number of qubits, and S1 performs slightly better than other software frameworks.

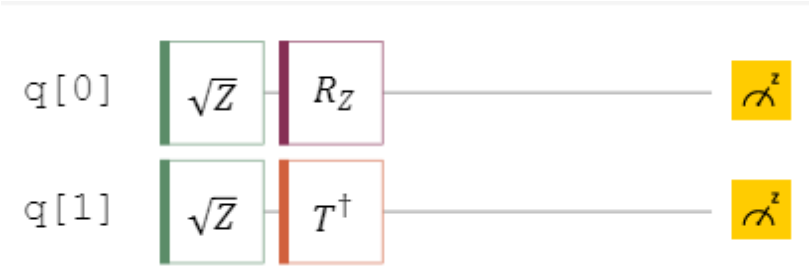

Figure 5.1: Sample Random Circuit Generated from a Simulator

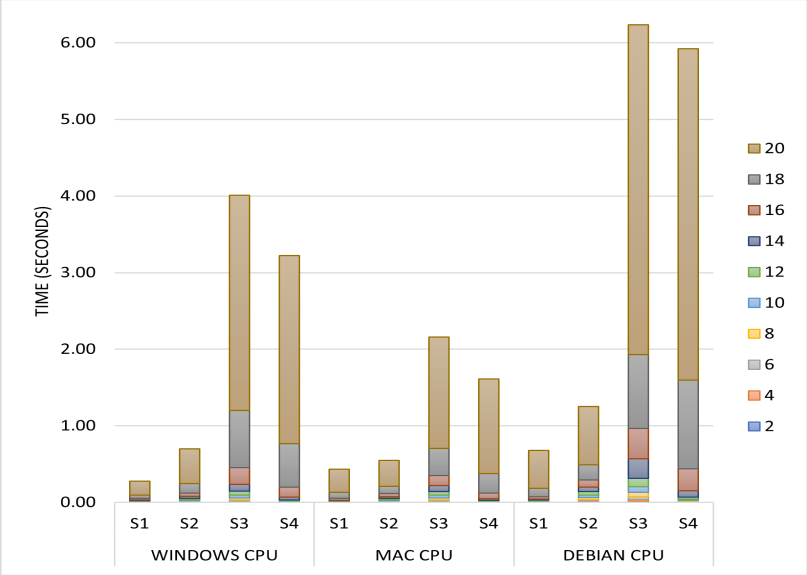

Figure 5.2: Result - System Comparison Using Random Quantum Circuits.

5.2 Experiment 2: CPU vs GPU Comparison

In this experiment, we simulated state vector simulators with S1 and S2 but ran the experiments in System 3 (which is a CPU) and a Tesla T4 GPU (Google Colab). From figure 5.3, we can see that GPU simulations far outperform CPU simulations. Hence, it is advisable to run the simulations on a GPU wherever possible.

5.3 Experiment 3: Simulation on Cloud Platforms

In this experiment, we simulate the randomly generated circuits in native cloud platforms such as IBM Cloud, Google Colab, and AWS. We can see from figure 5.4 that S2 is performing slightly better on IBM's cloud and Google Colab. S1 is performing better on AWS. Hence, in order to avoid giving any unfair advantage, we decided not to simulate other experiments using native platforms.

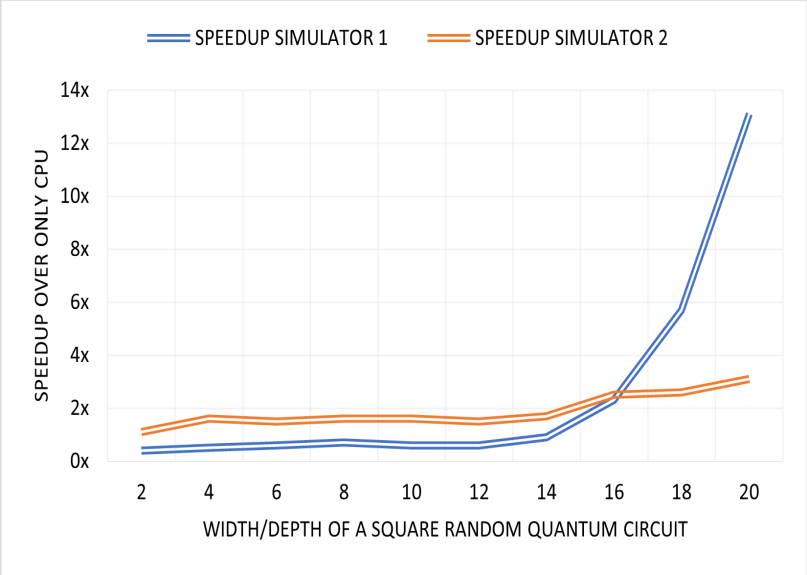

Figure 5.3: CPU vs GPU Comparison.

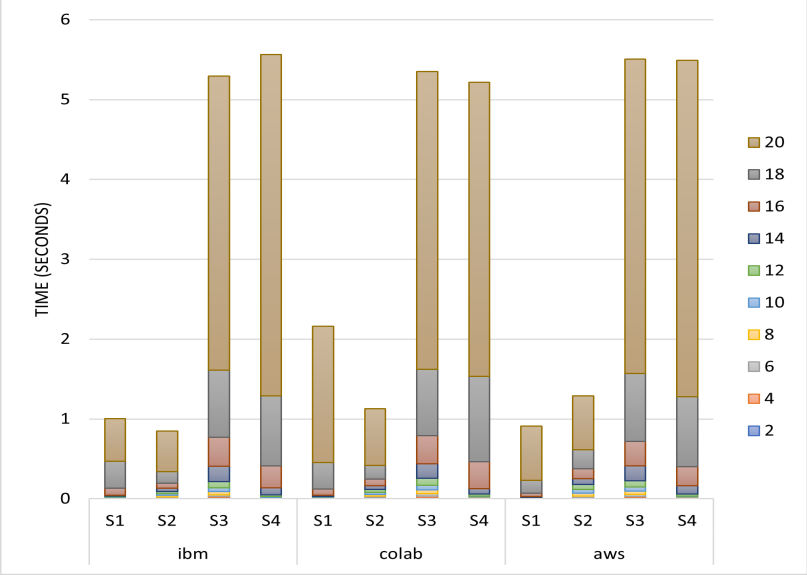

Figure 5.4: Simulation on Cloud Platforms.

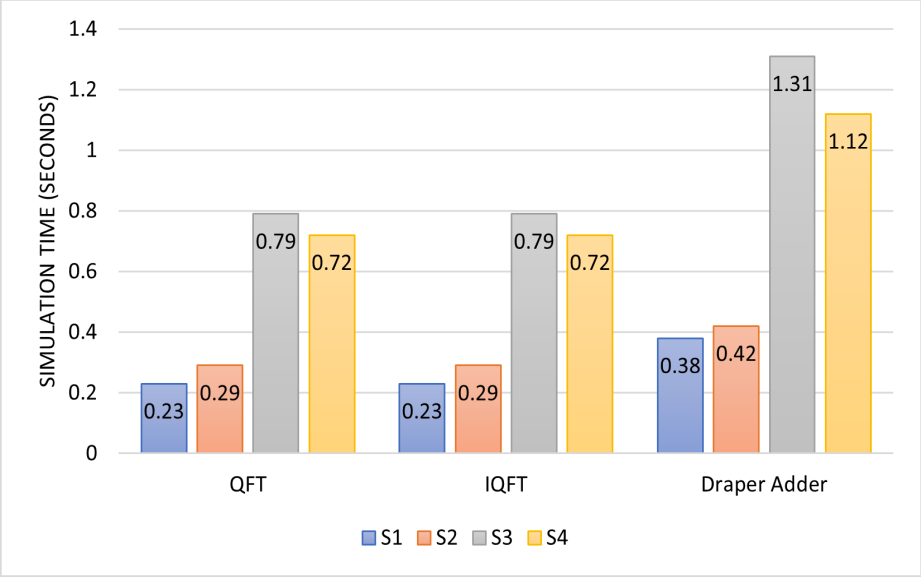

Figure 5.5: Density Matrix Simulations - Execution Time.

5.4 Experiment 4: Density Matrix Simulations

Here, we compared the density matrix simulators (noisy) (Harper *et al.* (2019)) using the quantum algorithms, QFT, IQFT, and Quantum Adder. This simulation was performed in System 1 (from table 4.1). The result shown in 5.5 and 5.6 shows that the density matrix scales twice as fast as the state vector simulators and is limited to a lesser number of qubits. We can see, S1 outperforms other software frameworks. In the case of density matrix simulators, we used a 5% bit flip error and a phase error of 0.05.

5.5 Experiment 5: Tensor Network Simulations

In this experiment, we used variational quantum algorithms (Cerezo *et al.* (2020)) such as the Variational Quantum Eigensolver (VQE) and a Quantum Neural Network (QNN) which are suitable for the implementation of a tensor network-based circuit. Figure 5.7 shows a sample Matrix-Product State tensor network-based quantum cir-

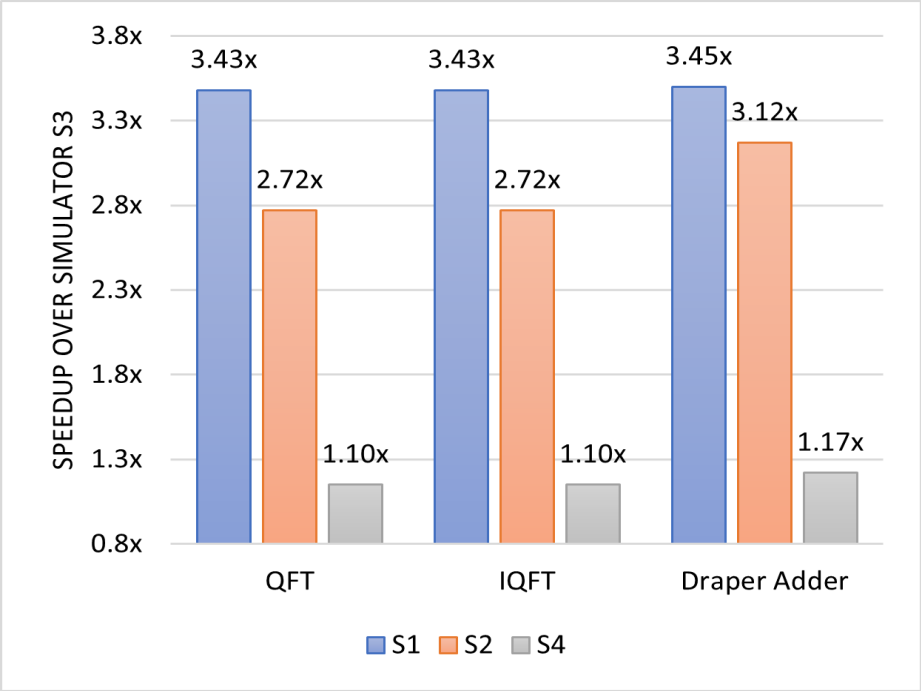

Figure 5.6: Density Matrix Simulations- Speedup over S3.

cuit. It is shown in figure 5.8 and 5.9 that S2 is performing better than S1 in the Tensor network simulation. Please note that tensor network simulators can extend beyond the 50 qubit limit of State Vector simulation, and there are only a handful of algorithms that can be implemented using tensor networks.

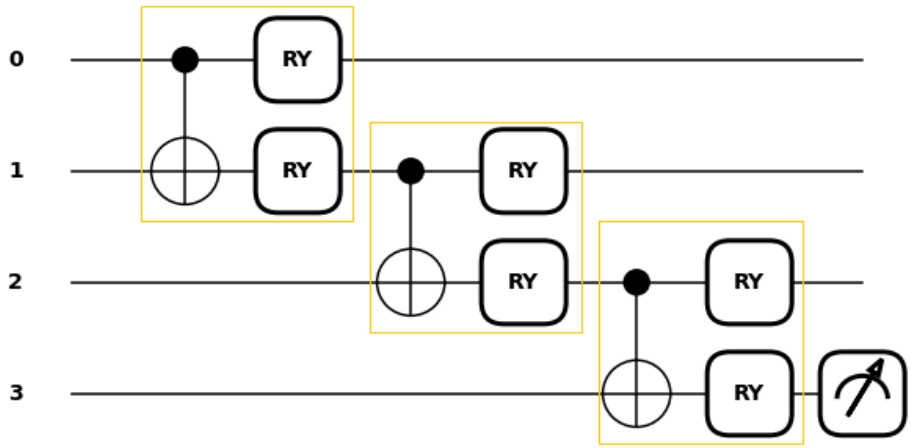

Figure 5.7: Tensor Network Quantum Circuit (MPS).

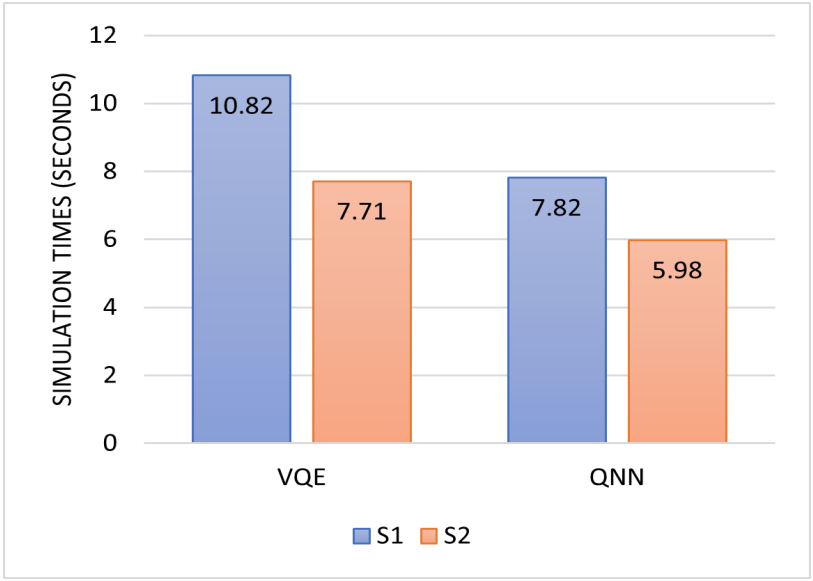

Figure 5.8: Tensor Network Simulations - Execution Time.

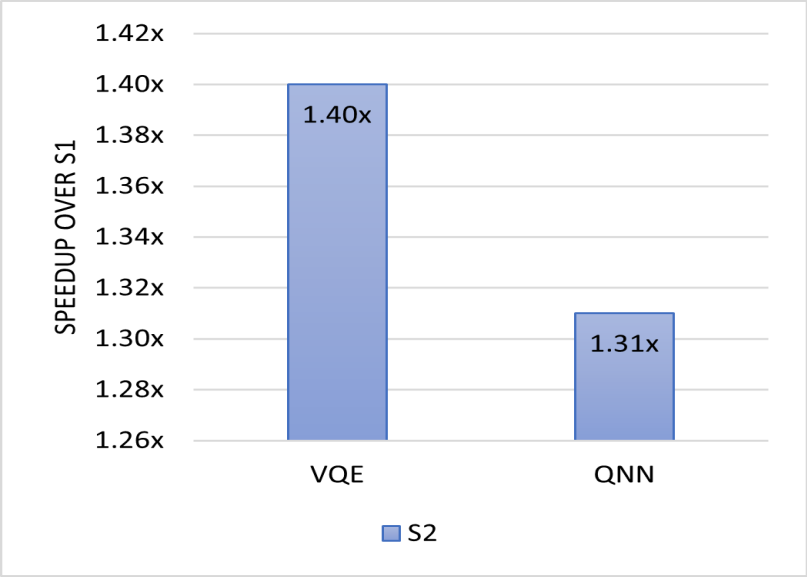

Figure 5.9: Tensor Network Simulations - Speedup.

Chapter 6

SIMULATIONS USING QML BENCHMARKS

The study presents a comprehensive benchmark suite for quantum computing that focuses on practical workloads in key areas such as vision, language, and recommendation. The benchmarks were carefully selected based on their commercial and research significance, and the suite includes a compact, yet representative set of three benchmarks. The source code for the benchmarks will be made open-source to engage the QML Benchmarking community.

This benchmark suite aims to provide a valuable resource for the quantum computing community, enabling researchers and practitioners to evaluate the performance of quantum computing platforms on practical workloads. By addressing the most significant areas of practical application, the study seeks to ensure that the benchmarks are relevant and meaningful for the quantum computing community.

The study also aims to contribute to the advancement of quantum computing by providing a benchmark suite that is designed to meet the unique challenges and requirements of quantum computing platforms. By providing a standardized set of benchmarks, the study seeks to enable fair and meaningful comparisons of quantum computing platforms and to help advance the state of the art in quantum computing research and development.

6.1 QML Benchmarking Suite

The study presents a comprehensive benchmarking suite for practical workloads. We adopt state-of-the-art models such as ResNet-50 v1.5, BERT, and NCF, with the addition of quantum layers, adjustment of parameter initialization, optimizer schedule, and data augmentation to further enhance performance.

- **Image Classification:** The task of image classification (Krizhevsky and Inc (2014)) is essential in computer vision, as it lays the foundation for other tasks such as object detection and segmentation. The ResNet-50 architecture is considered a state-of-the-art model for image classification, using deep residual networks that have set new performance records. Our study utilizes the ResNet-50 v1.5 pre-trained model, which employs quantum layers and adjusts parameter initialization, optimizer schedule, and data augmentation to further enhance performance.

- **Natural Language Processing:** Natural Language Processing (NLP) is an emerging field with practical importance, and the Wikipedia 2020/01/01 dataset provides a valuable resource for research and development. The BERT model has achieved state-of-the-art results in various NLP tasks, utilizing transformer-based architecture with pre-training on massive amounts of text data. Our study employs the BERT model, with 12/24 transformer layers, a hidden size of 768/1024, and a total of 110M/340M parameters.

- **Recommendation Systems:** Recommendation systems are crucial commercial workloads for internet companies, and the Neural Collaborative Filtering (NCF) approach is commonly used for benchmarking. NCF predicts user-item interactions and has been implemented in various domains, such as movie rec-

ommendations. Our study utilizes the MovieLens-20M public dataset made available by Mov (2016), which tends to be smaller than industrial ones, impacting the compute characteristics of the recommender.

6.2 Hybrid Classical-Quantum Layers

This paper adopts a hybrid approach that combines classical deep learning with variational quantum algorithms. The technique involves incorporating quantum layers into pre-trained transfer learning models such as ResNet-50, BERT, and NCF. This method takes advantage of the benefits of both classical and quantum machine learning. Figure 6.1 shows the block diagram of the development of the QML benchmark and how quantum layers are sandwiched.

6.3 Time-to-Train Metric

To assess the performance of ML systems, the time-to-train metric measures the duration required to achieve a specified quality target for each of the ML applications. The metric considers both system speed and accuracy and captures auxiliary operations. It is flexible enough to be applied to various training schemes. Time-to-train balances the trade-off between system-scale and software flexibility while preventing quality-reducing optimizations. We will also separately measure the time it takes to train the classical and quantum layers.

6.4 Timing Rules

We included only the model creation, system initialization, and data reformatting are excluded. The metric aims to capture the time it takes to achieve a certain quality target while accounting for both system speed and accuracy. To ensure the stability of timing results, multiple runs of each benchmark are mandated due to the stochastic

nature of deep learning methods. The number of runs required varies depending on the task, with vision tasks requiring 5 runs and other tasks requiring 10 runs, in order to ensure that 90% of entries from the same system is within a certain margin of deviation. The result is obtained by discarding the fastest and slowest runs and reporting the arithmetic mean of the remaining runs.

6.5 Quality Thresholds

We selected quality metrics that are near-state-of-the-art for each benchmark, based on experiments with reference implementations. We opted for higher quality thresholds to prevent optimizations and minimize run-to-run variation, as accuracy tends to be more variable early in training.

6.6 Reference Implementations

References of benchmark models using Quantum software packages serve as a basis for defining benchmark training procedures and hyperparameter restrictions. The restrictions balance the need for system optimization while limiting the hyperparameter search space to ensure fairness.

6.7 Carefully Chosen Parameters

For each of the ML tasks, I have carefully chosen the number of qubits and the ansatz to improve the metrics. Table 6.1 shows the hyperparameters that are made modifiable in the reference code. Developers/researchers using the benchmarks will be able to tune these hyperparameters to improve the efficiency of their system or model.

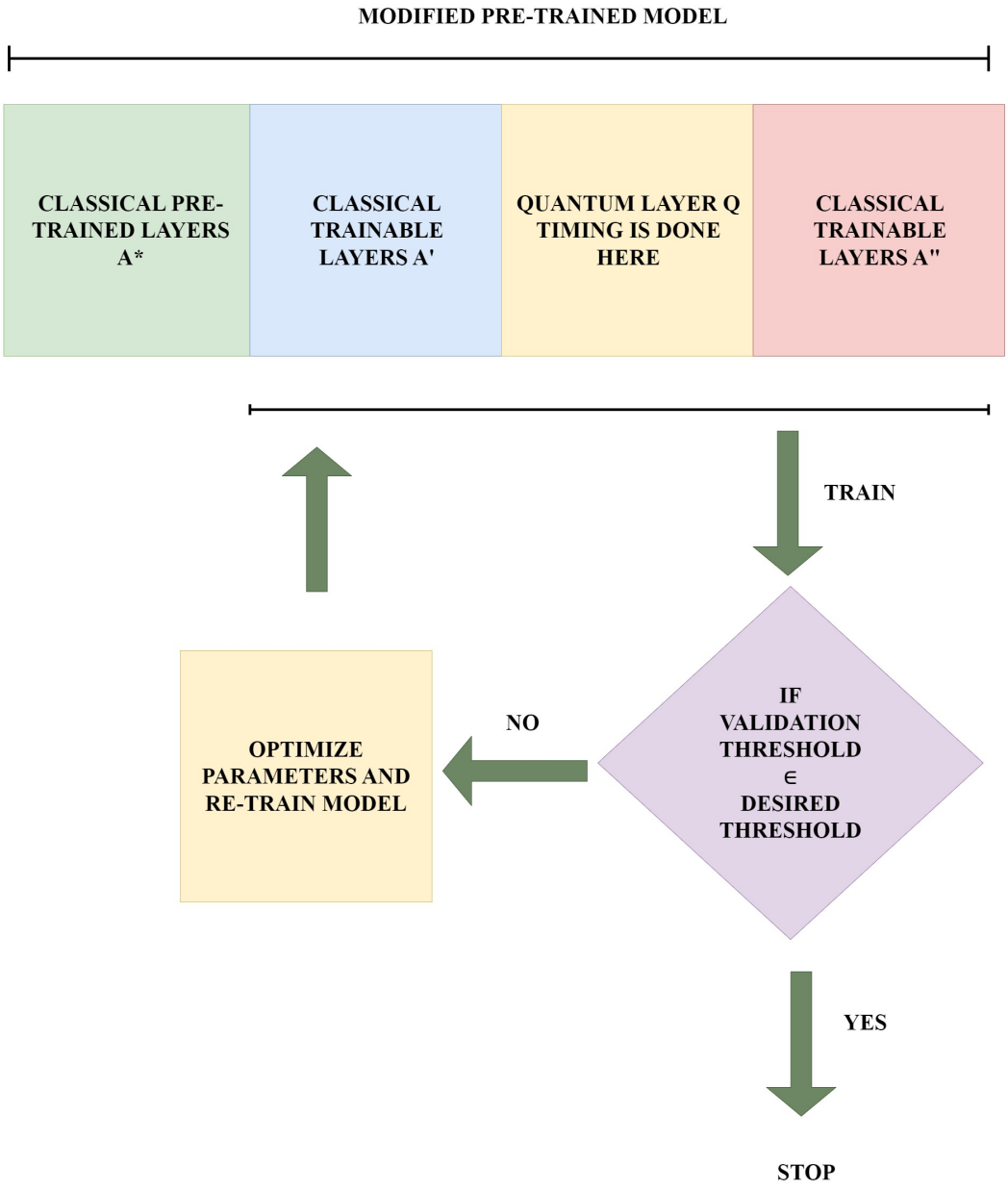

Figure 6.1: Block Diagram for the Implementation of Quantum Circuit Blocks for ML Tasks

Model	**Modifiable Hyperparameters**
All that use SGD	Batch size, Learning-rate schedule parameters
ResNet-50 v1.5	-
BERT	Learning rate, $\beta 1$, $\beta 2$
NCF	Optimizer: Adam or Lazy Adam, Learning rate, $\beta 1$, $\beta 2$

Table 6.1: Modifiable Hyperparameters

Chapter 7

RESULTS USING QML BENCHMARKS

Our benchmarking suite aims to advance the field of quantum machine learning (QML) by promoting innovation through constructive competition. The present study features the implementation of Quantum Neural Network (QNN) models(Farhi and Neven (2018)) on a standard hardware infrastructure, comprised of an Intel(R) Xeon(R) CPU with a clock speed of 2.20GHz and a memory capacity of 32GB RAM. The development of these QNN models was accomplished through the utilization of the Tensorflow Machine Learning framework. Experiments were carried out utilizing state vector simulators from S1, S2, S3, and S4, revealing that S1 has outperformed its counterparts in all the ML tasks by a small margin. The exact benchmark timings (in minutes) are shown in Figure 7.2. However, the rapidly evolving field of quantum computation mandates acknowledging that these results may alter with software advancements and optimizations.

Our benchmarks can also be used to determine the performance of real quantum computers for QML applications. Our results indicate improvements in performance and scaling are can be incorporated into the underlying software infrastructure before passing on to users. We believe that the benchmark suite has the potential to drive further hardware innovation in the QML field, just as it has done for classical machine learning. As with any benchmark, the purpose of our benchmark is also to encourage constructive competition and measure progress in the field. We hope that the suite can drive advancements in QML and improve the performance and capabilities of quantum computing systems.

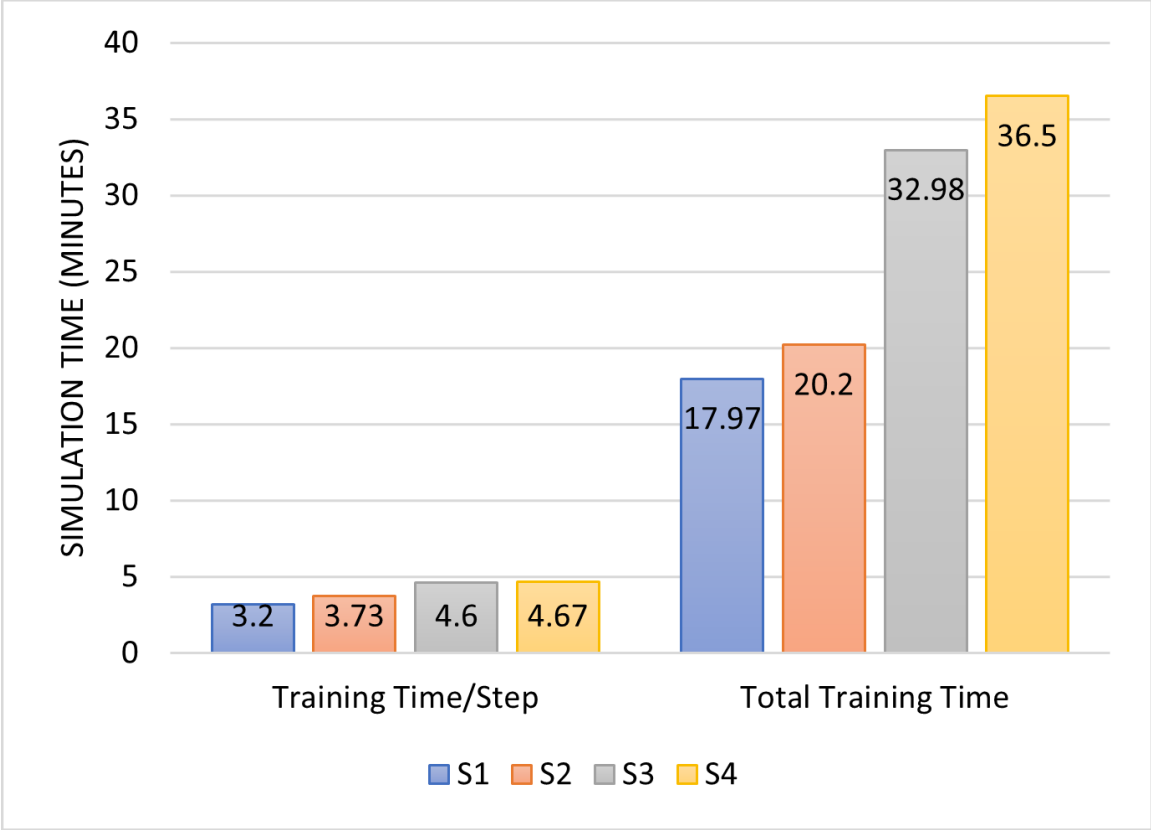

Figure 7.1: Benchmarking using simple Image Classification MNIST dataset - Execution Time

7.1 Experiment 1: Benchmarking Image classification task using MNIST dataset

In this simulation, we compared the state vector simulators from different software vendors such as S1, S2, S3, and S4. The dataset we used is the MNIST dataset. Figure 7.1 shows the total training time and training time for each step in the quantum layers. It can be seen that S1 takes lower time and S3 is almost double that. Although MNIST was useful and fast for our experiment, we decided not to include it in the benchmarking suite as the MNIST dataset is not really practical and is only used for research and teaching purposes for its simplicity.

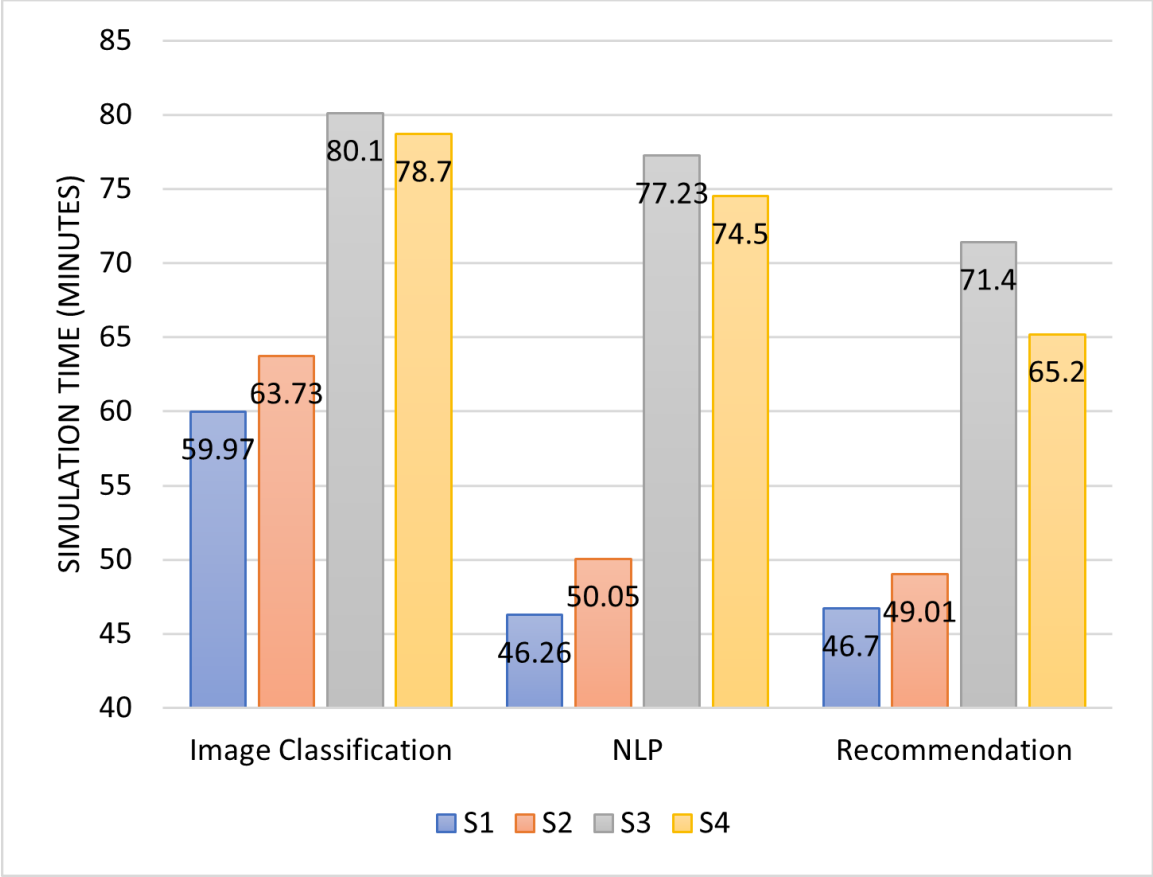

Figure 7.2: Benchmarking using practical datasets - Execution Time

7.2 Experiment 2: State Vector Simulation of Practical Datasets

In this experiment, we perform simulations using the state vector simulators of S1, S2, S3, and S4. From Figures 7.2 and 7.3, we can see that S1 was much faster in converging to the desired metrics/accuracy with all the ML tasks such as Image classification, NLP, and Recommendation.

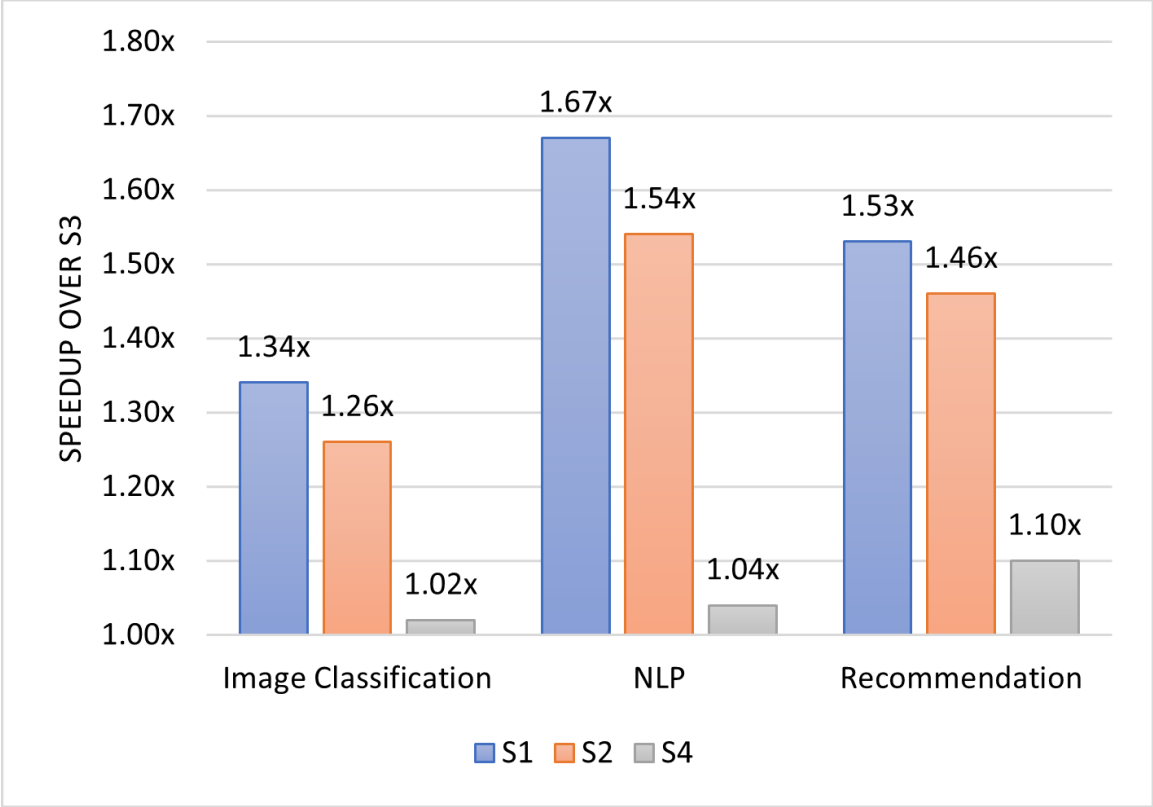

Figure 7.3: Benchmarking using practical datasets - Speedup

Chapter 8

CONCLUSION AND FUTURE WORK

The paper presents an analysis of various quantum simulators, comparing their performance across different CPU and GPU architectures. The aim is to provide a benchmark for future evaluation, as the performance of simulators is expected to vary due to the constantly evolving nature of the quantum field. The analysis includes the implementation of quantum algorithms to evaluate the performance of state vectors, density matrices, and tensor network simulators. The development of the code follows software development principles, allowing for the accommodation of new devices, simulators, and benchmarks in the future.

The paper also introduces the Quantum ML benchmarking suite, which aims to provide a comprehensive representation of both industrial and academic use cases in the field of machine learning. The suite includes precise definitions of model architectures and training procedures for each benchmark, allowing for equivalent workload comparisons. Reference implementations and rule definitions address the unique challenges of benchmarking QML training, such as stochastic processes and the need for workload variation at different system scales. Results from the suite demonstrate the importance of realistic dataset size and the impact of small hyperparameter changes, as well as highlighting the existence of subtle optimizer-algorithm variations in frameworks that affect convergence.

Moving forward, potential future work includes the addition of more quantum algorithms for evaluating simulator performance, as well as updating the ML tasks

with reference implementations using quantum algorithms and state-of-the-art ML models and datasets.

Other scopes of development in the domain of quantum benchmarking are:

- Developing new metrics and tools for measuring the efficiency and accuracy of quantum algorithms, such as quantum volume or entanglement metrics.
- Exploring different benchmarking approaches for different types of quantum systems, such as noisy intermediate-scale quantum (NISQ) devices or fault-tolerant quantum computers.
- Collaborating with industry partners to ensure the relevance and practicality of quantum benchmarking efforts and apply the results to real-world applications.
- Investigating the potential use of quantum computers for benchmarking classical algorithms and systems, such as machine learning models or cryptography protocols.

Overall, the paper contributes to the ongoing efforts to advance the field of quantum computing and machine learning, providing valuable insights and benchmarks for future research and development.

REFERENCES

"T.p.p. transaction processing performance council, website", URL https://www.tpc.org/ (2010).

"Movielens 20m dataset by grouplens research", URL https://grouplens.org/datasets/movielens/20m/ (2016).

"Ai and compute", URL https://openai.com/research/ai-and-compute (2018).

"cuquantum sdk — nvidia", URL https://developer.nvidia.com/cuquantum-sdk (2022).

Auer, P., M. Herbster and M. K. Warmuth, "Exponentially many local minima for single neurons", (1995).

Beer, K., D. Bondarenko, T. Farrelly, T. J. Osborne, R. Salzmann, D. Scheiermann and R. Wolf, "Training deep quantum neural networks", Nature Communications 2020 11:1 **11**, 1–6, URL https://www.nature.com/articles/s41467-020-14454-2 (2020).

Brown, T. B., B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. Mccandlish, A. Radford, I. Sutskever and D. Amodei, "Language models are few-shot learners", Advances in Neural Information Processing Systems **33**, 1877–1901, URL https://commoncrawl.org/the-data/ (2020).

Cerezo, M., A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio and P. J. Coles, "Variational quantum algorithms", Nature Reviews Physics **3**, 625–644, URL http://arxiv.org/abs/2012.09265 http://dx.doi.org/10.1038/s42254-021-00348-9 (2020).

Cheng, H.-T., L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir, R. Anil, Z. Haque, L. Hong, V. Jain, X. Liu and H. S. G. Inc, "Wide deep learning for recommender systems", URL https://arxiv.org/abs/1606.07792v1 (2016).

Choromanska, A., M. Henaff, M. Mathieu, G. B. Arous and Y. LeCun, "The loss surfaces of multilayer networks", Journal of Machine Learning Research **38**, 192–204, URL https://arxiv.org/abs/1412.0233v3 (2014).

Coleman, C., D. Kang, D. Narayanan, L. Nardi, T. Zhao, J. Zhang, P. Bailis, K. Olukotun, C. Ré, M. Zaharia and S. Dawn, "Analysis of dawnbench, a time-to-accuracy machine learning performance benchmark", (2019).

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

Devlin, J., M.-W. Chang, K. Lee, K. T. Google and A. I. Language, "Bert: Pre-training of deep bidirectional transformers for language understanding", Proceedings of the 2019 Conference of the North pp. 4171–4186, URL https://aclanthology.org/N19-1423 (2019).

Dixit, K. M., "The spec benchmarks", Parallel Computing **17**, 1195–1209 (1991).

Efthymiou, S., M. Lazzarin, A. Pasquale and S. Carrazza, "Quantum simulation with just-in-time compilation", Quantum **6**, 814, URL https://quantum-journal.org/papers/q-2022-09-22-814/ (2022).

Farhi, E. and H. Neven, "Classification with quantum neural networks on near term processors", URL https://arxiv.org/abs/1802.06002v2 (2018).

Gori, M. and A. Tesi, "On the problem of local minima in backpropagation", IEEE Transactions on Pattern Analysis and Machine Intelligence **14**, 76–86 (1992).

Goyal, P., P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia and K. H. Facebook, "Accurate, large minibatch sgd: Training imagenet in 1 hour", URL https://arxiv.org/abs/1706.02677v2 (2017).

Harper, R., S. T. Flammia and J. J. Wallman, "Efficient learning of quantum noise", Nature Physics **16**, 1184–1188 (2019).

He, K., X. Zhang, S. Ren and J. Sun, "Deep residual learning for image recognition", Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition **2016-December**, 770–778, URL https://arxiv.org/abs/1512.03385v1 (2015).

He, X., L. Liao, H. Zhang, L. Nie, X. Hu and T. S. Chua, "Neural collaborative filtering", 26th International World Wide Web Conference, WWW 2017 pp. 173–182, URL https://arxiv.org/abs/1708.05031v2 (2017).

Krizhevsky, A. and G. Inc, "One weird trick for parallelizing convolutional neural networks", URL https://arxiv.org/abs/1404.5997v2 (2014).

Krizhevsky, A., I. Sutskever and G. E. Hinton, "Imagenet classification with deep convolutional neural networks", Communications of the ACM **60**, 84–90, URL https://dl.acm.org/doi/10.1145/3065386 (2017).

LaRose, R., "Overview and comparison of gate level quantum software platforms", Quantum **3**, 130, URL

Lubinski, T., S. Johri, P. Varosy, J. Coleman, L. Zhao, J. Necaise, C. H. Baldwin, K. Mayer and T. Proctor, "Application-oriented performance benchmarks for quantum computing", URL https://arxiv.org/abs/2110.03137 (2021).

Martiel, S., T. Ayral and C. Allouche, "Benchmarking quantum co-processors in an application-centric, hardware-agnostic and scalable way", IEEE Transactions on Quantum Engineering **2**, URL http://arxiv.org/abs/2102.12973 http://dx.doi.org/10.1109/TQE.2021.3090207 (2021).

Mattson, P., C. Cheng, C. Coleman, G. Diamos, P. Micikevicius, D. Patterson, H. Tang, G.-Y. Wei, P. Bailis, V. Bittorf, D. Brooks, D. Chen, D. Dutta, U. Gupta, K. Hazelwood, A. Hock, X. Huang, A. Ike, B. Jia, D. Kang, D. Kanter, N. Kumar, J. Liao, G. Ma, D. Narayanan, T. Oguntebi, G. Pekhimenko, L. Pentecost, V. J. Reddi, T. Robie, T. S. John, T. Tabaru, C.-J. Wu, L. Xu, M. Yamazaki, C. Young and M. Zaharia, "Mlperf training benchmark", URL https://arxiv.org/abs/1910.01500v3 (2019).

Miller, K., C. Broomfield, A. Cox, J. Kinast and B. Rodenburg, "An improved volumetric metric for quantum computers via more representative quantum circuit shapes", URL https://arxiv.org/abs/2207.02315v1 (2022).

Sandler, M., A. Howard, M. Zhu, A. Zhmoginov and L. C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks", Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition pp. 4510–4520, URL https://arxiv.org/abs/1801.04381v4 (2018).

Schuld, M., V. Bergholm, C. Gogolin, J. Izaac and N. Killoran, "Evaluating analytic gradients on quantum hardware", (2018).

Stokes, J., J. Izaac, N. Killoran and G. Carleo, "Quantum natural gradient", Quantum **4**, 269, URL https://quantum-journal.org/papers/q-2020-05-25-269/ (2020).

Strubell, E., A. Ganesh and A. McCallum, "Energy and policy considerations for deep learning in nlp", Journal of International Marketing **53**, 3645–3650, URL https://arxiv.org/abs/1906.02243v1 (2019).

Tomesh, T., P. Gokhale, V. Omole, G. S. Ravi, K. N. Smith, J. Viszlai, X. C. Wu, N. Hardavellas, M. R. Martonosi and F. T. Chong, "Supermarq: A scalable quantum benchmark suite", Proceedings - International Symposium on High-Performance Computer Architecture **2022-April**, 587–603, URL https://arxiv.org/abs/2202.11045v3 (2022).

Wack, A., H. Paik, A. Javadi-Abhari, P. Jurcevic, I. Faro, J. M. Gambetta and B. R. Johnson, "Quality, speed, and scale: three key attributes to measure the performance of near-term quantum computers", URL https://arxiv.org/abs/2110.14108v2 (2021).

Zhu, C., H. Mao, S. Han and W. J. Dally, "Trained ternary quantization", 5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings URL https://arxiv.org/abs/1612.01064v3 (2016).

APPENDIX A

QUANTUM COMPUTATION

Quantum computation is a field of study in computer science and physics that aims to use quantum-mechanical phenomena to perform computations. Unlike classical computers, which operate on classical bits that can only be in two states (0 or 1), quantum computers use quantum bits or qubits that can exist in multiple states simultaneously.

Definition A.0.1 *The state of a qubit can be represented using a mathematical concept called a quantum state vector, which is a complex vector in a vector space called a Hilbert space. Mathematically, the state of a single qubit can be written as:*

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{A.1}$$

where α and β are complex numbers called probability amplitudes, and $|0\rangle$ and $|1\rangle$ are basis states representing the classical states 0 and 1, respectively. The probability of measuring the qubit in the state $|0\rangle$ is given by $|\alpha|^2$, and the probability of measuring it in the state $|1\rangle$ is given by $|\beta|^2$. The sum of the squares of the probability amplitudes must be equal to 1, i.e., $|\alpha|^2 + |\beta|^2 = 1$.

In quantum computation, operations are performed on qubits using quantum gates, which are unitary transformations that preserve the probability amplitudes of the quantum state vector. For example, the Pauli-X gate is a quantum gate that flips the state of a qubit from $|0\rangle$ to $|1\rangle$ and vice versa. Mathematically, the Pauli-X gate can be represented as:

$$X = [0 \quad 1\ 1 \quad 0] \tag{A.2}$$

where the matrix elements represent the amplitudes for the transformation of the basis states $|0\rangle$ and $|1\rangle$.

Quantum algorithms, such as Shor's algorithm for factoring large numbers and Grover's algorithm for searching unsorted databases, use quantum gates to manipulate qubits in a way that takes advantage of quantum-mechanical phenomena such as superposition and entanglement to solve problems more efficiently than classical algorithms. In quantum computation, one major challenge is the presence of noise, which can arise from various sources such as environmental factors, imperfect control of the quantum system, and errors in the hardware. Quantum noise can cause errors in the computation, and therefore, it is important to understand and mitigate its effects. There are different types of quantum noise that can affect the performance of a quantum computer. One common type is decoherence, which occurs when a quantum system interacts with its environment and loses its quantum coherence over time. Other types of noise include thermal noise, which arises from the temperature of the system, and readout noise, which arises from errors in the measurement process. Understanding and mitigating the effects of these different types of quantum noise is crucial for the development of practical quantum computers.

APPENDIX B

QUANTUM COMPUTING SIMULATORS

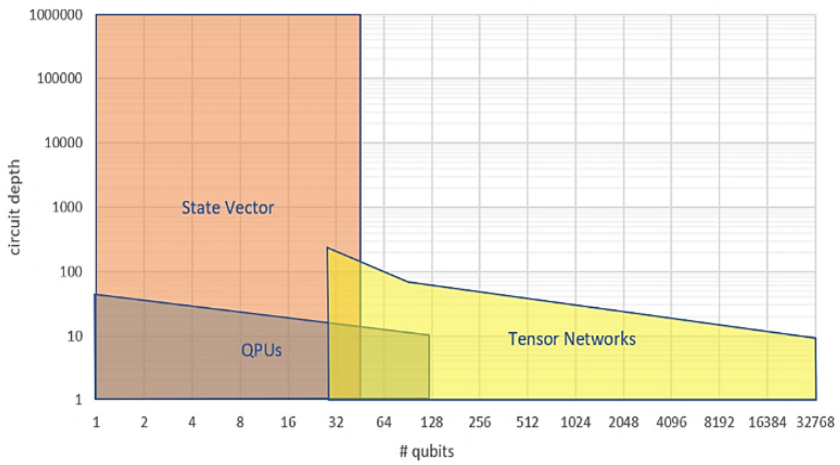

Figure B.1: Simulators: Computational Limits

In this section, we will discuss the different types of quantum simulators based on their mathematical representations. Fig. B.1 shows the current computing limits of the state vector, tensor network simulators, and the latest Quantum Processing Unit (QPU), in terms of the number of qubits and circuit depth.

Definition B.0.1 ***Pure state or state vector simulators*** *are used to simulate the evolution of an ideal quantum system. In the case of gate-model quantum computers, they simulate the initialization of perfect qubit states, circuits (operations), and measurement. The ideal state of the quantum state is represented using normalized($\langle\psi|\psi\rangle = 1$, for any state ψ) state vectors that contain the statistical information of that state. The quantum gates or operators are represented using unitary matrices($U^\dagger U = UU^\dagger = I$, for any operator U).*

Definition B.0.2 ***Mixed state simulators or density matrix simulators*** *are used to simulate and model noise and decoherence in a quantum system. They use the density matrix representation of a quantum state to model all possible noisy states. A density matrix (ρ) is given by,*

$$\rho = \sum_i p_i \ \ |\psi_i\rangle \langle\psi_i|$$

where ψ_i is one of the possible quantum states with probability p_i. While mixed states are very good at simulating noisy qubits, gate, measurement, and reset errors, they are slower than pure-state simulators.

Definition B.0.3 *In* ***Tensor Network simulators****, quantum circuits are represented using tensor networks. When large tensors are factorized into smaller ones, they give rise to specific patterns of networks. These networks are called tensor networks. Some popular tensor networks are matrix product states (MPS) and tree tensor networks (TTN). The representation of the tensor contractions in MPS and TTN. Quantum circuits with tensor representations are capable of simulating circuits of a large number of qubits and smaller depths.*

APPENDIX C

QUANTUM ALGORITHMS

In this section, we will discuss the quantum algorithms that were used in the first part of the experiments for the benchmarking suite.

Quantum Fourier Transform

Introduction to QFT

In classical computing literature, Discrete Fourier Transform (DFT) and its faster implementation of the Fast Fourier Transform (FFT) algorithms have ubiquitous applications in Signal Processing, Information Theory, Matrix Multiplication, and much more. QFT is analogous to DFT, such that it is acted upon the Amplitudes of the state of a Wavefunction.

Definition of QFT

The Quantum Fourier Transform on a state $|\Psi\rangle$ is given by,

$$QFT_n : |\Psi\rangle = \sum_{j=0}^{N-1} a_j \, |j\rangle_n \rightarrow \sum_{j=0}^{N-1} b_j \, |j\rangle_n$$

where

$$b_j = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} a_k e^{\frac{2\pi jki}{N}}.$$

Having $N = 2^n$ and $\omega_n = e^{\frac{2\pi i}{N}}$, The General form of the QFT function becomes,

$$QFT_n(|\psi\rangle) = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} \sum_{k=0}^{N-1} a_k \omega^{jk} \, |j\rangle_n$$

1-Qubit QFT

Let U_{QFT} be the Unitary Matrix corresponding to the QFT operation. We first take a 1-Qubit system with state $|\psi\rangle = \alpha \, |0\rangle + \beta \, |1\rangle$ and N = 2, based on the definition of the QFT, we have,

$$U_{QFT} \, |\psi\rangle = \frac{1}{\sqrt{2}}(\alpha + \beta) \, |0\rangle + \frac{1}{\sqrt{2}}(\alpha - \beta) \, |1\rangle$$

This is the same result that we get when we apply a Hadamard Gate H to a Qubit. Hence, we can see that the Hadamard operator is the QFT operation for N = 2.

N-Qubit QFT

Extending the definition of the QFT, we have the below Unitary U_{QFT} for an N-Qubit System,

$$U_{QFT} = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} \sum_{k=0}^{N-1} \omega_N^{jk} \left|k\right\rangle \left\langle j\right|$$

Circuit Implementation of QFT

Let ROT_k represent a rotation of the state $|\psi\rangle$ by an angle $\frac{2\pi}{2_k}$ Then, we can implement QFT using the below circuit for N = 8 using Hadamard, Rotation and Swap Gates.

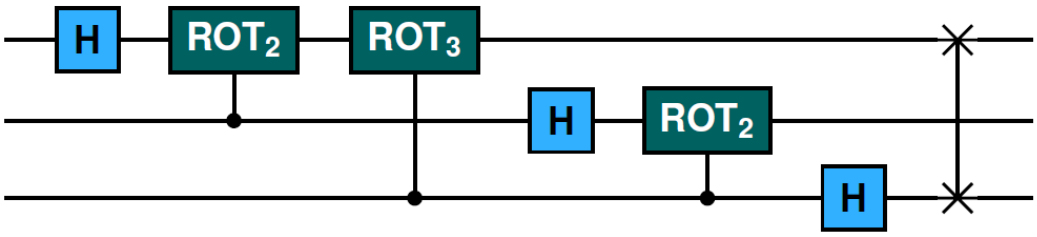

Figure C.1: Gate-Model Circuit Diagram of a 3-Qubit QFT.

Inverse QFT

We can get the Inverse QFT, by applying the same operation in reverse. This is due to the fact that the Unitary operations we consider are invertible. Inverse QFT is given by,

$$IQFT_n(|\psi\rangle) = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} \sum_{k=0}^{N-1} a_k \omega^{-jk} \left|j\right\rangle_n$$

where $\omega = e^{\frac{2\pi i}{N}}$

Quantum Adder

A variational quantum adder is a quantum circuit that performs addition of two numbers encoded as quantum states. It is an important application of quantum computing that can potentially speed up certain types of calculations. The circuit consists of a set of quantum gates that act on the input states, producing an output state that encodes the sum of the input numbers. The circuit is "variational" because it is optimized to minimize the difference between the output state and the desired state using a classical optimization algorithm.

The circuit can be represented mathematically using the following equations:

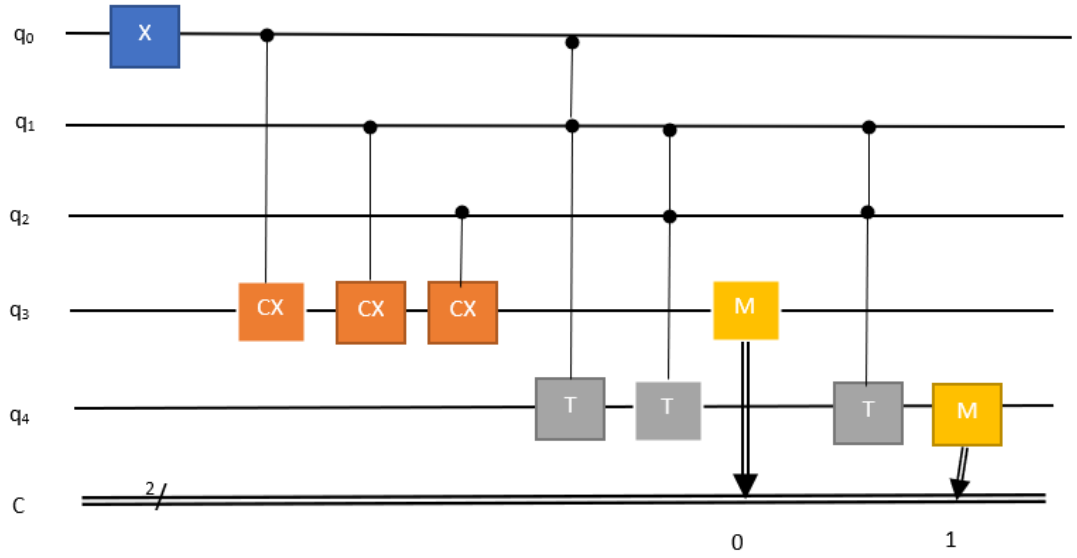

Figure C.2: Circuit Diagram of Quantum Adder

Let the two input numbers be encoded as quantum states $|\psi_1\rangle$ and $|\psi_2\rangle$, where $|\psi_i\rangle$ represents the i-th qubit in the quantum state.

The goal of the circuit is to produce an output state $|\psi_{out}\rangle$ that represents the sum of the input numbers. This can be written as:

$$|\psi_{out}\rangle = U_{adder}|\psi_1\rangle|\psi_2\rangle$$

where U_{adder} is the quantum circuit that performs the addition.

The variational nature of the circuit comes from the fact that the circuit is optimized to minimize the difference between the output state and the desired state $|\psi_{target}\rangle$, which represents the correct sum of the input numbers. This can be written as:

$$E = |||\psi_{target}\rangle - U_{adder}|\psi_1\rangle|\psi_2\rangle||^2$$

where E is the cost function that the classical optimization algorithm seeks to minimize, and $||\cdot||$ denotes the Euclidean norm.

The optimization algorithm finds the values of the parameters in the circuit that minimize the cost function E, which correspond to the optimal values of the quantum gates that implement the circuit. This is done using techniques such as gradient descent (Stokes *et al.* (2020)) or other classical optimization methods. The resulting circuit is then used to perform the addition operation on the input numbers, producing an output state that represents their sum.

Variational Quantum Eigensolver

A Variational Quantum Eigensolver (VQE) (Cerezo *et al.* (2020), Schuld *et al.* (2018)) is an algorithm that uses a quantum computer to find the ground state energy of a given Hamiltonian. The VQE algorithm is a hybrid algorithm that combines classical and quantum computations. The VQE algorithm can be summarized in figure C.3.

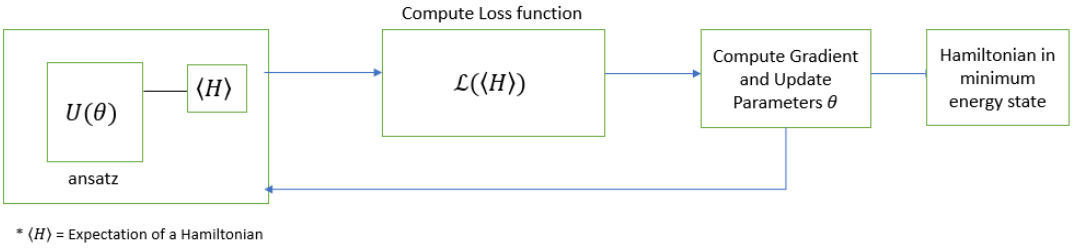

Figure C.3: Block Diagram of VQE

$$H = \sum_{i,j} h_{i,j} \sigma_i \sigma_j$$

where σ_i and σ_j are the Pauli matrices and $h_{i,j}$ are the coefficients of the Hamiltonian. The quantum circuit used in the VQE algorithm is typically composed of layers of gates, which are parametrized by a set of variables, $\vec{\theta}$. The circuit is designed to prepare a quantum state, $|\psi(\vec{\theta})\rangle$, which represents the ground state of the Hamiltonian. The energy of the quantum state, $|\psi(\vec{\theta})\rangle$, can be calculated by measuring the expectation value of the Hamiltonian, H, in the quantum state. This can be written as:

$$E(\vec{\theta}) = \langle \psi(\vec{\theta}) | H | \psi(\vec{\theta}) \rangle$$

The goal of the VQE algorithm is to find the set of parameters, $\vec{\theta}^*$, which minimize the energy, $E(\vec{\theta})$. This can be achieved by using a classical optimization algorithm, such as gradient descent or the Nelder-Mead algorithm, to update the parameters iteratively. The algorithm stops when the energy converges to a minimum value.

Quantum Neural Network

A Quantum Neural Network (QNN) as described by Farhi and Neven (2018) is a type of quantum machine learning model that combines the principles of neural networks and quantum computing. It is used to solve optimization problems, pattern recognition, and data classification tasks. In a QNN, quantum circuits are used to represent the weights and biases of a classical neural network. The quantum circuits are parameterized, meaning that they can be tuned to optimize the performance of the QNN. The QNN is trained using a classical optimization algorithm that adjusts the quantum circuit parameters to minimize the cost function.

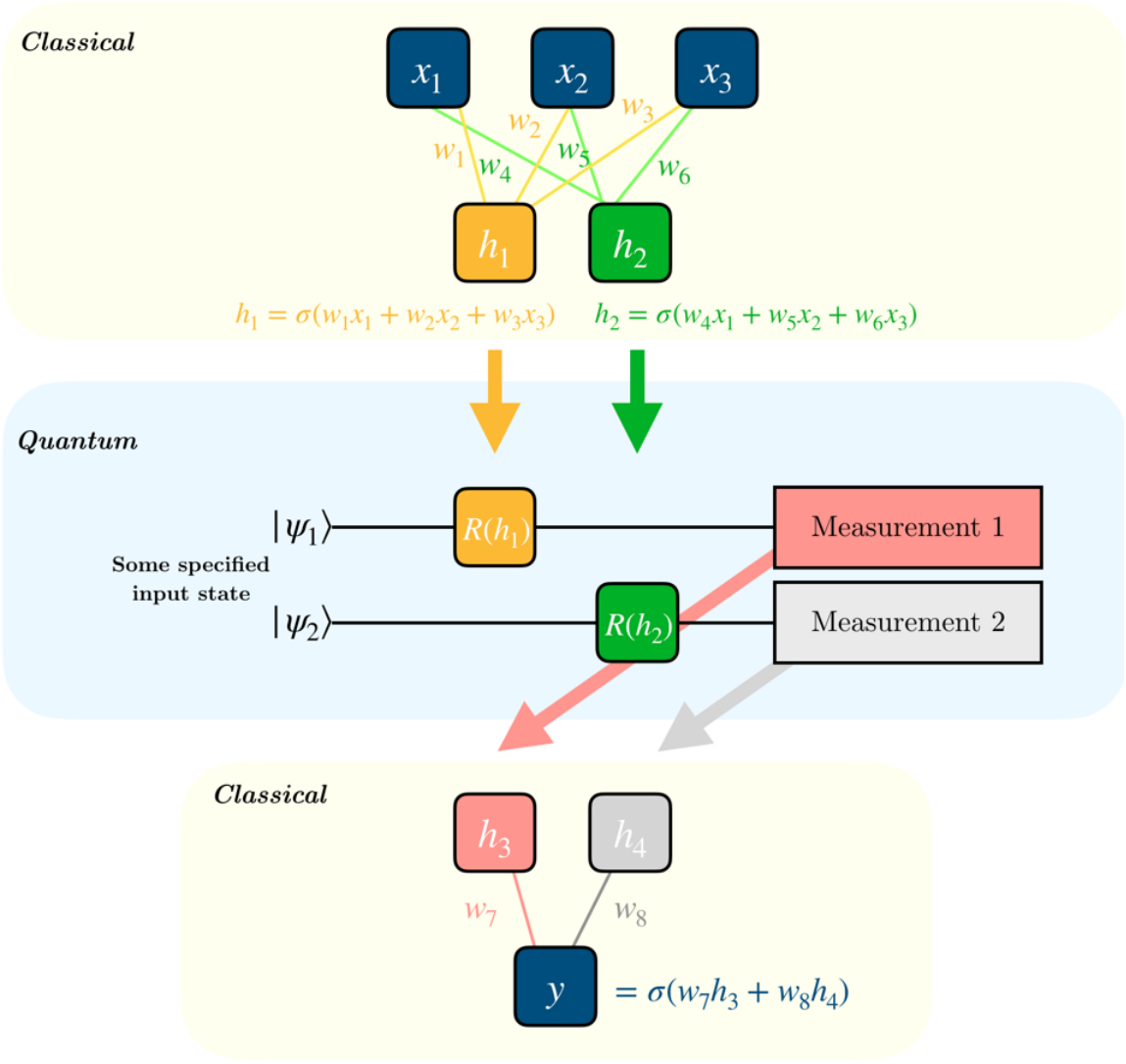

Figure C.4: Block Diagram of QNN

APPENDIX D

QML BENCHMARKING DATASETS

Below are the datasets that were used in the QML Benchmarking suite.

1. **MNIST**: The MNIST dataset is a collection of handwritten digits used in machine learning research. The dataset contains 60,000 training images and 10,000 testing images of grayscale, 28x28 pixels in size. Each image represents a single digit from 0 to 9, with labels indicating the correct digit represented in the image. The MNIST dataset has been widely used as a benchmark for image classification tasks and is often used as a starting point for beginners in the field of machine learning

2. **ImageNet**: It is a large-scale visual recognition challenge dataset, designed to enable the development and benchmarking of state-of-the-art algorithms in image classification. The dataset contains over 14 million images that cover more than 20,000 object categories. Each image in the dataset is annotated with multiple labels, providing additional information about the object or objects depicted in the image. The dataset is split into training and validation sets, with the former containing around 1.2 million images and the latter around 50,000 images. The ImageNet dataset (Deng *et al.* (2009)) is widely used in computer vision research and has been instrumental in advancing the state of the art in image classification and other related tasks.

3. **Wikipedia**: It is a vast collection of articles, discussions, and other textual content from the online encyclopedia Wikipedia. The dataset contains more than 6 million articles in different languages, including English, German, French, and Spanish, among others. Each article includes structured metadata, such as the title, category, and timestamp, as well as the unstructured content in the form of plain text. The dataset also includes discussion pages, edit histories, and user pages, which provide additional contextual information about the articles and the editors who contribute to the platform. Overall, the Wikipedia dataset is a valuable resource for natural language processing, machine learning, and other data-driven applications that require large amounts of diverse, high-quality textual data.

4. **MovieLens**: MovieLens-20M is a popular movie recommendation dataset made available by Mov (2016), that contains a vast amount of data on movie ratings. The dataset comprises over 20 million ratings given by users to approximately 27,000 movies, collected from the online movie recommendation website MovieLens. The ratings range from 0.5 to 5 on a 5-point scale, and each movie has at least one rating. In addition to the ratings, the dataset includes information about the movies such as their titles, genres, and release years, as well as demographic information about the users such as their age, gender, and occupation. This information makes it a rich dataset for developing and testing movie recommendation systems.