Circuit Design and Efficient Simulation of Quantum Inner Product and Empirical Studies of Its Effect on Near-Term Hybrid Quantum-Classic Machine Learning

Hao Xiong[†], **Yehui Tang**[†], **Xinyu Ye**[†], **Junchi Yan**[*]
Department of Computer Science and Engineering & MoE Key Lab of AI, Shanghai Jiao Tong University
{taxuexh,yehuitang,xinyu_ye,yanjunchi}@sjtu.edu.cn
https://github.com/ShawXh/qip_cvpr24

Abstract

For the essential operation, namely inner product (IP) as widely adopted in classic computing e.g. matrix multiplication, its quantum counterpart: quantum inner product (QIP), has also been recently theoretically explored with a verifiable lower complexity on quantum computers. However, it remains unclear for the embodiment of the quantum circuits (QC) for QIP, let alone a (thorough) evaluation of the QIP circuits, especially in a practical context in the NISQ era by applying QIP to ML via hybrid quantum-classic pipelines. In this paper, we carefully design the QIP circuits from scratch, whose complexity is in accordance with the theoretical complexity. To make the simulation tractable on classic computers, especially when it is integrated in the gradient-based hybrid ML pipelines, we further devise a highly-efficient simulation scheme by directly simulates the output state. Experiments show that the scheme accelerates the simulation for more than 68k times compared with the previous circuit simulator. This allows our empirical evaluation on typical machine learning tasks, ranging from supervised and self-supervised learning via neural nets, to K-Means clustering. The results show that the calculation error brought by typical quantum mechanisms would incur in general little influence on the final numerical results given sufficient qubits. However, certain tasks e.g. ranking in K-Means could be more sensitive to quantum noise.

1. Introduction

Quantum algorithms have been actively studied to solve some classical problems with theoretically much lower complexity compared with their classic counterparts, e.g. Quantum SVM [27], HHL algorithm for linear equations [12], quantum Fourier transform [8] etc. The power of quantum computing (QC) comes from the inherent quantum parallelism associated with the superposition principle [7].

[*]Corresponding author. [†] Equal contribution. We acknowledge Zhanpeng Zhou who helped the IBM Quantum access and experiments during his stay in Tokyo, Japan. This is an independent work without any funding.

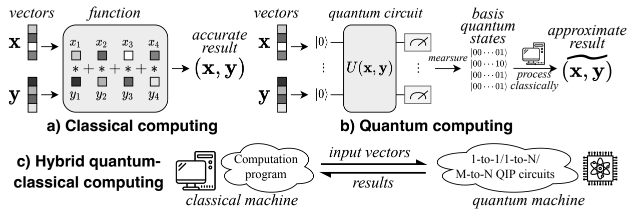

Figure 1. a/b) Inner product computing in classic and quantum ways. c) Hybrid quantum-classical computing scheme with QIP.

Despite the theoretical advantages, the pure quantum methods still face challenges to be put into practical use due to the limited number of available qubits and capability in dealing with deep quantum circuits of the NISQ computers. Compared with the pure quantum methods, hybrid quantum-classical algorithms are regarded as well-suited for execution on NISQ devices by combining quantum computers with classical computers [10]. Researchers design quantum circuits as the substitutions of specific classical parts to gain super high efficiency, e.g. Hardamad transform to replace the classical Conv2D layer [25], or to achieve specific functions that are computationally impractical on classical computers, e.g. feature extraction in quantum Hilbert space [20, 31].

Seeing the essential role of inner product (IP) in classic machine learning (ML) and their quantum counterparts QIP in QML (as will be discussed in detail), this paper studies several basic yet less-studied problems regarding QIP. On the one hand, we develop the detailed embodiment of quantum circuits of QIP with its behavior on quantum computers. On the other hand, we develop highly-efficient simulation scheme for QIP and empirically evaluate the effect of noise brought by quantum mechanisms, by integrating simulated QIP into popular ML methods on classic computers.

As aforementioned, IP is a fundamental operator in modern ML methods. From vector inner product and matrix multiplication, to distance measurement in clustering algorithms, to linear layers of deep neural networks, a large number of IP operations are involved. The acceleration of the IP operation will greatly improve the running speed of classical ML algorithms that contain a lot of IP operations.

As illustrated in Fig. 1 a), on classical machines, the precise result the IP of vectors $\mathbf{x}$ and $\mathbf{y}$ is obtained by adding up the product of vector elements x_i and y_i in a deterministic way without any probabilistic noise. Its complexity is $O(d)$ (d: the dimension of input vectors). In contrast, QC uses a 'quantum circuit' (see Appendix G for preliminaries) to run a quantum algorithm. As shown in Fig. 1 b), the circuit is defined by a unitary $U(\mathbf{x}, \mathbf{y})$, whose input is the quantum basis state $|0 \ldots 0\rangle$ and the output is also a quantum state. By measuring the output quantum state (and processing the results on a classic computer), we can obtain an approximate result of the IP $\widetilde{(\mathbf{x}, \mathbf{y})}$ whose error level is dependent on the number of used qubits and also influenced by the inherent noise of NISQ computers. By QC, the time complexity of IP computation can be much reduced.

Most efforts for QIP circuit design (see Sec. 2) focus on the theoretical complexity side. Yet it is nontrivial to develop a promising hybrid quantum-classical computing scheme (Fig. 1 c) with the *explicit embodiment of QIP circuits* for running on NISQ machines, which is still absent in literature. This paper aims to design quantum circuits $U(\mathbf{x}, \mathbf{y})$ that can run on quantum computers as a complete solution, which can be readily integrated (as will be shown in our paper) into existing ML algorithms via differentiable gradient computing. We further show how our circuits can be efficiently simulated on classic commodity computers which allows to verify the correctness of the numerical QIP results especially as a building block in different ML methods. **The contributions are in three folds as follows.**

Circuit Design of QIP on Quantum Computers. We design the proposed quantum circuits to accommodate quantum machines of different scales. From scratch, we present three QIP circuits for 1-to-1, 1-to-N, and M-to-N IP computation tasks respectively. The required numbers of qubits range from $1 + t + \lceil \log d \rceil$ to $1 + MNt + \lceil \log M \rceil + \lceil \log N \rceil + \lceil \log d \rceil$ (see Table 2). Our methods are based on and enhance the Hadamard test and Quantum Phase Estimation algorithms, with our mathematically proven lower complexity than classical approaches, which is also in line with the previous theoretical studies for QIP (see Sec. 2). We further show its efficacy on quantum computers.

Efficient Simulation of QIP on Classic Computers. Existing quantum simulation platforms (e.g. Qiskit [26], PennyLane [3], etc.) on classic computers cannot simulate our devised circuits due to their high complexity. Hence, we develop an efficient simulation scheme to evaluate our circuits. Our scheme is agnostic to the circuits, and directly simulates the theoretical output quantum states produced by the circuits and employs an efficient sampling strategy to reduce the original probability calculation complexity from $O(2^{2t})$ to close to $O(2^t)$, while avoiding the memory overhead of $O(2^t)$ (t: the number of qubits to store results).

Evaluating QIP in ML on Classic Computers. Equipped with our high-efficient simulation schemes, we evaluate the effectiveness of our methods by investigating the influence (on both numerical calculation error and overall prediction performance) of applying the QIP operators in various existing ML methods on classic computers for simulation. Experiments empirically show how the QC-induced errors affect different ML methods in different ways and in general our techniques can be applicable to these ML models regarding the final overall prediction performance whose decrease is small (see details in our experiments).

2. Related Works

Quantum Inner Product. In QC, the inner product with a specific physical meaning of fidelity, between two quantum states is typically computed by swap test [6] or modified Hadamard test [22]. These two ways involve quantum circuits implemented on quantum devices and have been adopted widely in quantum ML [21, 32]. Quantum phase estimation (QPE) [24] or amplitude estimation [4] can be used to readout the result of the inner product encoded into the amplitude of the superposition state of the quantum circuits. [19, 33] employ the swap test and QPE as a part of the quantum neuron. However, the 1-to-1 QIP computation by the simple combination of the Hadamard test and QPE can hardly meet the ubiquitous needs of batch-wise computing of IP in ML models. Therefore, we propose to extend the 1-to-1 QIP circuit to 1-to-many and many-to-many cases to fully utilize the power of quantum parallelism.

Quantum Matrix Multiplication (QMM). Matrix multiplication is a more general case of IP between a set of row vectors and a set of column vectors. QMM which aims to utilize quantum parallelism has been studied in the quantum community in decades. [29] constructs explicit QC, including row-column multipliers and the modulo N matrix multiplier. But they fail to use superposition and parallelism to enable the quantum advantages. [32] designs a quantum hyper-parallel algorithm based on swap test. [28] devises three quantum algorithms based on swap test, singular value estimation, and HHL (quantum algorithm for linear systems of equations). [18] proposes a binary quantum matrix multiplier that uses the basis state to store data, which means that the algorithm requires no quantum tomography and can obtain results in a few measurements. However, both [29] and [18] only give solutions for matrices of integers, limiting their application in more common float operations; [32] and [28] store the computing results in the amplitude of quantum states, which requires the use of quantum state tomography [9] to convert the result to its classic form, resulting in exponential measurement overheads; a complete, explicit, and implementable quantum circuit for QMM is still absent since previous works only give the theoretical study by quantum linear algebra without a concrete solution. Moreover, all these methods have no numerical experiments nor practical applications in ML/DL methods due to the lack

Table 1. Quantum matrix multiplication methods. Quantum speedup brought by the inherent quantum parallelism and no reliance on tomography are important for practical use of QIP. V denotes the maximum value of the two matrices' entries.

Methods	Explicit Circuit	# Qubits	Quantum Speedup	Time Complexity of Computing $\mathbf{XY}$ ($\mathbf{X} \in \mathbb{R}^{M \times d}$, $\mathbf{Y} \in \mathbb{R}^{d \times N}$)	Data Type	Obtain Results	Experiments: Simulation	Experiments: Devices	Experiments: Applications
[29]	✓	$(MN+M+N+d)d\log(V)$	No	$O(MNd)$	Integer	Measurement	No experiments		
[32]	Partly	N/A	✓	$O(MN\epsilon^{-2}\log\eta^{-1})$ (η: the accuracy)	Float	Tomography			
[28]	Not given		✓	$O(\Vert\mathbf{X}\Vert_F\Vert\mathbf{Y}\Vert_F\Vert\mathbf{XY}\Vert_F^{-1}\epsilon^{-1})$	Float	Tomography			
[18]	✓	N/A	✓	$O(MN\log d)$	Integer	Measurement			
Ours	✓	Referred to Table 2	✓	$O(\epsilon^{-1}\log MNd)$ (M-to-N case in Theorem 3, 1-to-1 and 1-to-N cases referred to Theorem 1, 2)	Float	Measurement	Circuit (slow, Sec. 3), output state (fast, Sec. 4)	Quantum, classical	Numerical computing, ML, DL

Table 2. Designed QIP circuits in three modes, where t is the number of measuring qubits for each pair of the inner product; d is the input dimension; M and N is the number of input.

Mode	Number of required qubits	Use case
1-to-1	$1+t+\lceil\log d\rceil$	vector inner product
1-to-N	$1+Nt+\lceil\log N\rceil+\lceil\log d\rceil$	linear mapping
M-to-N	$1+MNt+\lceil\log N\rceil+\lceil\log M\rceil+\lceil\log d\rceil$	matrix multiplication

of an efficient way for simulation.

A detailed comparison of the quantum matrix multiplication methods is shown in Table 1. Our method manages to **i)** build the concrete quantum circuit, **ii)** employ the parallelism of quantum characteristics to achieve quantum speedup, **iii)** encode results through quantum basis state with small measurement overheads, and we further **iv)** propose an efficient measurement simulation to perform numerous experiments including ML tasks, **v)** lays a foundation for using quantum matrix multiplication to accelerate classical ML tasks.

3. Principles and Circuit Design of QIP

Our QIP computing is conducted by a quantum circuit as a sequence of quantum gates and measurements with initialized quantum bits (qubits) as input. Readers are referred to Appendix G for the preliminaries.

We devise three QIP circuits: 1-to-1, 1-to-N, and M-to-N, to cope with different computing scenarios and different scales of quantum hardware. These circuits approximate the inner product by combining the two quantum algorithms, Hadamard test and Quantum Phase Estimation (QPE): first, load the classical data into a quantum state, and then transform the fidelity of the quantum states (i.e. the inner product of the classical data) into quantum phases by Hadamard test, and finally estimate the phase of the quantum state through QPE. The approximated inner product can be obtained from the measured output quantum states of the circuits.

Notations. We first briefly introduce the notations: $\mathbf{i}$ denotes the imaginary unit $\sqrt{-1}$, $|\cdot\rangle$ denotes a quantum state, $\otimes$ is the tensor product, bold lowercase letters $\mathbf{x}$, $\mathbf{y}$ are vectors, $(\cdot,\cdot)$ denotes the classical inner product of two vectors while $\widetilde{(\cdot,\cdot)}$ denotes the quantum one, and normal capital letters (e.g. U, E, D) represent unitaries.

Step-by-Step Circuit Construction from Special to General Cases. We use the following three theorems to present the step-by-step circuit construction methodology for the three cases. The requirements for the number of qubits for these circuits are listed in Table 2. The detailed proofs are put in Appendix H.

Theorem 1 (**1-to-1 QIP Circuit** (for vector inner product))**.** *There exists a quantum circuit $U(\mathbf{x}, \mathbf{y})$ computing the inner product of two normalized vector $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ with complexity $O(\frac{\log d}{\epsilon})$ where ϵ is a given precision parameter.*

Proof (A Sketch). We use different colors to mark qubits with different roles. Blue is for loading data and Hadamard test, and green is for the output phase by QPE.

$$\begin{aligned} \text{Input} &= |0\rangle^{\otimes t}|0\rangle|0\rangle^{\otimes\lceil\log d\rceil} \\ &\xrightarrow[\textit{Hardamard Test}]{E(\mathbf{x},\mathbf{y})} e^{\mathbf{i}\theta}|0\rangle^{\otimes t}|w_+\rangle - e^{-\mathbf{i}\theta}|0\rangle^{\otimes t}|w_-\rangle \\ &\xrightarrow[\textit{QPE}]{D(\mathbf{x},\mathbf{y})} e^{\mathbf{i}\theta}|\boldsymbol{R}_+\rangle|w_+\rangle - e^{-\mathbf{i}\theta}|\boldsymbol{R}_-\rangle|w_-\rangle, \end{aligned} \tag{1}$$

where $E(\mathbf{x}, \mathbf{y})$ is a unitary as the 'encoder' transforming classical data $\mathbf{x}$ and $\mathbf{y}$ to two orthogonal quantum states $|w_+\rangle$ and $|w_-\rangle$, and storing the IP in the phase $\theta = \arccos\big(-(\mathbf{x}, \mathbf{y})\big)/2 \in [0, \frac{\pi}{2}]$ by Hadamard test, $D(\mathbf{x}, \mathbf{y})$ is a unitary as the 'decoder' that decodes phase θ as quantum states $|\boldsymbol{R}_\pm\rangle$. The IP in the phase θ can be recovered with high probability by measurement in the computational basis of the state $|\boldsymbol{R}_+\rangle = \frac{1}{2^t}\sum_{x=0}^{2^t-1}\sum_{k=0}^{2^t-1} e^{-\frac{2\pi\mathbf{i}k}{2^t}(x-2^t\theta/\pi)}|x\rangle$. We define $2^t\theta/\pi = a + 2^t\delta/\pi$ where a is an integer in $[0, 2^t - 1]$. Then the probability of measuring $|\boldsymbol{R}_+\rangle$ as $|a\rangle$ is $\left|\frac{1}{2^t}\sum_{k=0}^{2^t-1} e^{2\mathbf{i}k\delta}\right|^2$. The two cases $a = \lceil 2^t\theta/\pi\rceil$ and $a = \lfloor 2^t\theta/\pi\rfloor$ have the highest probabilities. □

Theorem 2 (**1-to-N ($N \geq 2$) QIP Circuit** (for linear mapping))**.** *There exists a quantum circuit computing inner products between a normalized vector $\mathbf{x} \in \mathbb{R}^d$ and a set of N normalized vectors $\{\mathbf{y}_i\}_{i=0}^{N-1}$, $\mathbf{y}_i \in \mathbb{R}^d$ with the time complexity $O(\frac{\log Nd}{\epsilon})$ (ϵ is a given precision parameter).*

Proof (A Sketch). We add an index register to the 1-to-1 QIP circuit to encode multiple vectors into quantum states. It runs as follows with the index register marked in red:

$$\begin{aligned} &\text{Input} = |0\rangle^{\otimes\lceil\log N\rceil}|0\rangle^{\otimes Nt}|0\rangle|0\rangle^{\otimes\lceil\log d\rceil} \\ &\xrightarrow[\textit{Hardamard Test}]{E(\mathbf{x},\{\mathbf{y}_i\})} \frac{1}{\sqrt{N}}\sum_{i=0}^{N-1}|i\rangle(e^{\mathbf{i}\theta_i}|0\rangle^{\otimes Nt}|w_{i+}\rangle - e^{-\mathbf{i}\theta_i}|0\rangle^{\otimes Nt}|w_{i-}\rangle) \\ &\xrightarrow[\textit{QPE}]{D(\mathbf{x},\{\mathbf{y}_i\})} \frac{1}{\sqrt{N}}\sum_{i=0}^{N-1}|i\rangle(e^{\mathbf{i}\theta_i}|\boldsymbol{R}_{i+}\rangle|w_{i+}\rangle - e^{-\mathbf{i}\theta_i}|\boldsymbol{R}_{i-}\rangle|w_{i-}\rangle), \end{aligned} \tag{2}$$

where $|\boldsymbol{R}_{i+}\rangle$ is defined as:

$$
\begin{aligned}
|\boldsymbol{R}_{i+}\rangle =& \big(\underbrace{|+\rangle \otimes \cdots \otimes |+\rangle}_{it \text{ terms}}\big)|\boldsymbol{r}_{i+}\rangle\big(\underbrace{|+\rangle \otimes \cdots \otimes |+\rangle}_{(N-i-1)t \text{ terms}}\big),\ |+\rangle \\
=& \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle.
\end{aligned} \tag{3}
$$

We define $|\boldsymbol{r}_{i+}\rangle = \frac{1}{2^t}\sum_{x=0}^{2^t-1}\sum_{k=0}^{2^t-1} e^{-\frac{2\pi \mathbf{i} k}{2^t}(x-2^t\theta_i/\pi)}|x\rangle$. From $|\boldsymbol{r}_{i+}\rangle$, the approximated inner product $\widetilde{(\mathbf{x}, \mathbf{y}_i)}$ can be recovered. The procedure is the same as Theorem 1. □

Theorem 3 (M**-to-**N ($M, N \geq 2$) **QIP Circuit** (for matrix multiplication))**.** *There exists a quantum circuit computing inner products between a set of M vectors $\{\mathbf{x}_i\}_{i=0}^{M-1} \in \mathbb{R}^d$ and another set of N vectors $\{\mathbf{y}_j\}_{j=0}^{N-1} \in \mathbb{R}^d$ with time complexity $O(\frac{\log MNd}{\epsilon})$ (ϵ is the precision parameter).*

The core idea of Theorem 3 is to stack the 1-to-N QIP circuits in a similar way as Theorem 2 by adding an index register for vectors in $\{\mathbf{x}_i\}_{i=0}^{M-1}$. For page limit, we put the detailed proof in Appendix H.4.

Remarks. In practical ML, the 1-to-1, 1-to-N, and M-to-N QIP circuits can be used for vector inner product, linear mapping, and matrix multiplication, respectively.

4. Efficient Simulation on Classic Computers

The simulation of the output states of our QIP circuits is essential for verifying the performance of our method for ML applications. However, simulating the circuits directly on classical machines is infeasible due to the exponential memory w.r.t. the number of qubits by existing quantum simulation platforms (e.g. Qiskit [26], PennyLane [3]). This limits the applicability of our QIP operators to ML models in real-world scenarios. Based on the theorem proofs and mathematical derivation in the previous section, we can obtain the analytical expression of the output quantum states. Therefore, we can simulate the output quantum states of the circuits directly, bypassing the step of simulating the circuit execution. In this way, we successfully integrate our QIP simulation module into existing ML methods. We develop a package `torch_qip` for efficient simulation of QIP. With `torch_qip`, one can easily define a differentiable operator in Pytorch by using the simulated QIP operator, such as quantum inner product, quantum matrix multiplication, etc.

4.1. Simulation of the Output Quantum States

Sec. 3 presents the output quantum states of different circuits by three theorems. As illustrated in Fig. 2 a), the workflow of the three circuits consists of two steps: first, 'encoding' the input vectors and their inner products into an entangled quantum state through an entangling layer; and second, 'decoding' the phase of the quantum state as a measurable output quantum state through the QPE layer. In the simulation process as depicted in Fig. 2 b), we can directly compute the output quantum state distribution after QPE and sample the measured basis quantum state from the distribution to

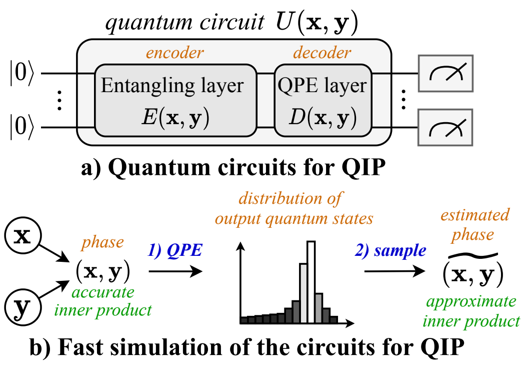
quantum circuit $U(\mathbf{x}, \mathbf{y})$
encoder
decoder
$|0\rangle$
$|0\rangle$
Entangling layer
$E(\mathbf{x}, \mathbf{y})$
QPE layer
$D(\mathbf{x}, \mathbf{y})$
a) Quantum circuits for QIP
x
y
phase
$(\mathbf{x}, \mathbf{y})$
accurate inner product
1) QPE
distribution of output quantum states
2) sample
estimated phase
$\widetilde{(\mathbf{x}, \mathbf{y})}$
approximate inner product
b) Fast simulation of the circuits for QIP

Figure 2. In **a)** we summarize the proposed three quantum circuits in a common framework, which is similar to the encoder-decoder architecture in neural networks. In **b)** we show a fast simulation scheme for the proposed quantum circuits.

Algorithm 1 Simulation of QIP on a classic computer.

```
1:  Input: vectors $\mathbf{x}$, $\mathbf{y}$, # of measuring qubits $t$, sampling times $r$.
2:  Compute the accurate inner product $(\mathbf{x}, \mathbf{y})$, real quantum phase
    $\theta = \arccos\big(-(\mathbf{x}, \mathbf{y})\big)/2$;
3:  $i_{left} = \texttt{floor}(2^t\theta/\pi)$, $i_{right} = \texttt{ceil}(2^t\theta/\pi)\%2^t$,
    $prob\_trace = \texttt{dict}()$, $i_{hits} = \texttt{list}()$;
4:  for $r\_ = 1, 2, \cdots, r$ do
5:      Generate a random number $p$ from $\texttt{Uniform}(0, 1)$;
6:      for $k = 0, 1, \cdots, 2^t - 1$ do
7:          if $i_{left}$ not in $prob\_trace$ do $p_{left} =$
            $\texttt{compute\_prob}(i_{left})$, $prob\_trace[i_{left}] = p_{left}$;
8:          else do $p_{left} = prob\_trace[i_{left}]$;
9:          $p = p - p_{left}$; if $p < 0$, $i_{hit} = i_{left}$, break;   ▶ The
            measured basis state has been hit
10:         For $i_{right}$ and $p_{right}$, do the same as the above line 7-9;
11:         $i_{left} = (i_{left} + 2^t - 1)\%2^t$, $i_{right} = (i_{right} + 1)\%2^t$;
            ▶ Search for other basis states
12:     end for
13:     $i_{hits}.\texttt{append}(i_{hit})$;
14: end for
15: (I) OUTPUT-AVG: $\widetilde{(\mathbf{x}, \mathbf{y})} = \texttt{avg}\Big(-\cos(2\pi i/2^t)$ for all
    $i$ in $i_{hits}\Big)$
16: (II) OUTPUT-MODE: select the mode number from $i_{hits}$ as
    $i_{out}$, $\widetilde{(\mathbf{x}, \mathbf{y})} = -\cos(2\pi i_{out}/2^t)$;
17: return $\widetilde{(\mathbf{x}, \mathbf{y})}$; ▶ The approximated inner product by quantum
function $\texttt{compute\_prob}(i)$:
1:  Complex number $x = 0 + 0\mathbf{i}$;
2:  for $k = 0, 1, \cdots, 2^t - 1$ do $x+ = \exp\big(2\pi\mathbf{i}k(i/2^t + \theta/\pi)\big)$
    end for
3:  return $x\overline{x}/2^{t+1}$                ▶ $\overline{x}$ is the conjugate of $x$
```

simulate the output of the entire quantum circuit. Thus, we simulate the output of the quantum circuit directly without simulating the operation of the quantum circuit.

The simulation method shown in Fig. 2 b) still faces the challenge of complexity. We denote the number of measurement qubits as t (referred to in the quantum algorithm in

Sec. 3). The total number of computational basis states is 2^t, and for each state calculating the probability consumes $O(2^t)$ time complexity. Therefore, $O(2^{2t})$ complexity in total is required to obtain the probabilities of all basis states. In Algorithm 1, we introduce two techniques to reduce the complexity of obtaining the basis state as the result of measurement from $O(2^{2t})$ to close to $O(2^t)$:

- **Quantum state search (QSS).** We start from the basis states with the highest probabilities, and stop once hit. In Algorithm 1, we initialize i_{left} and i_{right} as the two basis states with the highest probability (line 3), and then calculate the states with lower probability step by step (line 11). We stop calculating the probabilities of the basis states once the basis state is hit (line 9). As the initial probabilities p_{left} and p_{right} when $k = 0$ is usually very high, the entire process of calculating probabilities of basis states is very likely to stop at $k = 0$, saving much time from calculating the basis states with lower probabilities.
- **Reuse of calculation results.** In quantum computing, the output basis state of a circuit is often measured multiple times to obtain precise results. In the simulation, to avoid recalculating the probability of the ground state that has already been calculated, we reuse the results in previous sampling iterations that are stored in the variable $prob\_trace$. In this way, multiple sampling iterations won't linearly increase the running time of the simulation.

Parallelization. Note that the simulation of QIP between different vector pairs are independent. We parallelize Algorithm 1 for a batch of QIP simulation tasks (e.g. in M-to-N QIP there are totally MN simulation tasks) for speedup.

4.2. Output Inner Products From Quantum States

After we obtain the output basis quantum state of the circuits, the next step is to convert the quantum states into classical data, that is, the calculated inner product. As shown in Algorithm 1, we store the quantum states obtained by multiple times of sampling in i_{hits}, where the quantum state corresponding to each element $i \in i_{hits}$ is i's binary representation $|i_0 i_1 \cdots i_{t-1}\rangle$ s.t. $i = i_0 + 2i_1 + \cdots + 2^{t-1} i_{t-1}$. For each measured basis state $|i\rangle$, the corresponding computed inner product can be obtained by $\widetilde{(\mathbf{x}, \mathbf{y})} = -\cos(2\pi i/2^t)$. Given a list of quantum states i_{hits}, we define two strategies for the final computed inner product, i.e. i) OUTPUT-AVG which outputs the average inner products, and ii) OUTPUT-MODE which outputs the mode number of the inner products, as shown by line 15-16 in Algorithm 1. OUTPUT-AVG yields more accurate results for lower t and OUTPUT-MODE the opposite, as will be shown in the experiments.

4.3. Enabling Automatic Differentiation

The measurement process of the quantum state appears to be non-differentiable, which hinders the application of quantum applicability to most ML methods that rely on gradient optimization. To make the QIP operator differentiable, we adopt a simple solution: During the forward propagation, we substitute the result of the classical computing with the result of the quantum computing, and this result is used for the subsequent differentiable calculation; during the backward propagation, we obtain the gradient by applying the chain rule as in classical approaches.

We give an example of a differentiable quantum matrix multiplication operator in Pytorch with `torch_qip` in Fig. 12 (Appendix I), where `torch_qip.qip` is an operator developed according to Algorithm 1. `torch_qip.qip` takes exact classical inner products as input and returns the approximated quantum inner products. We also provide other differentiable quantum operators within the `torch_qip` package, such as quantum inner product, batch quantum matrix multiplication, quantum linear layer, and so on, which can replace their classical counterparts.

5. Experiments

Experiments include: i) (Sec. 5.1) Direct studies of the simulated QIP operator, including its visualization, accuracy, and efficiency. ii) (Sec. 5.2) Indirect studies in various ML scenarios to examine the impact of errors caused by the quantum scheme. All experiments of simulations run on a machine with Intel(R) Core(TM) i9 CPU @ 2.90GHz and 128GB memory. The used quantum computer is IBM *Brisbane* with 127 available qubits. Detailed settings of the ML/DL methods, tasks, and metrics are in Appendix J.

5.1. Evaluations on the QIP Core

5.1.1 Evaluation on the Circuit Simulator (with qiskit) and Quantum Machine

We evaluate our QCs with both classic simulators implemented with qiskit and IBM quantum cloud (Brisbane). Due to memory limitation, we can only run the 1-to-1 QIP circuit on the cloud with input of small dimensions $d = 2, 4, 8$ and a small number of qubits ($t = 4$). We run each experiment with 5 pairs of normalized random vectors as the data. We compare the error level of the simulator and Brisbane in Table 3. Results show that though the quantum machine yields good results sometimes, the error and variance are still relatively high to the results by our classic simulator. It also suggests that the current quantum computers (IBM Bristane) may not be fully ready for QIP applications.

5.1.2 Evaluation on the Proposed State Simulator

Our major experiments are performed by simulation on CPU. **Visualization.** We first study the output of QIP and the discrepancies between classic and quantum inner products. We show the output values with different numbers of qubits t for QPE output registers in Fig. 3. In the top, QIP-Avg is computed by the weighted average of all possible output values according to their probabilities, QIP-Mode is obtained by selecting the output value with the highest probability, and the classic computation results are identical to the ground-truth. QIP-Avg and QIP-Mode can also be viewed as the

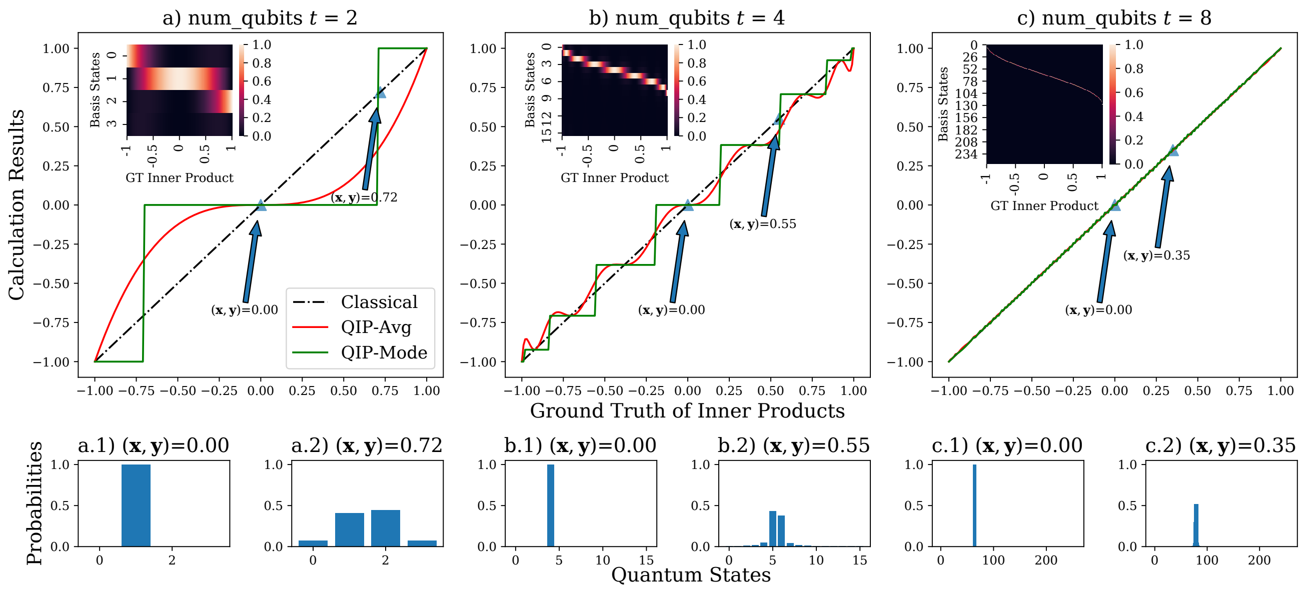

Figure 3. **Top figures:** x-axis represents the ground-truth of inner products, and the y-axis is the calculation results by sufficiently large times of measurements. Each figure shows the probability distribution of the output quantum basis states for each ground-truth inner product value. **The bottom six figures:** x-axis denotes the basis quantum states, and y-axis is the probabilities of quantum states being measured.

Table 3. The error level comparison between the QIP results of the classical circuit simulator and quantum machine.

	$d = 2$		$d = 4$		$d = 8$	
	Circuit Simulator	IBM Brisban	Circuit Simulator	IBM Brisban	Circuit Simulator	IBM Brisban
MSE ↓	0.0066 ± 0.0098	0.0990 ± 0.1835	0.0060 ± 0.0068	0.1183 ± 0.1739	0.0048 ± 0.0044	0.0387 ± 0.0399
MAE ↓	0.0560 ± 0.0586	0.1800 ± 0.2580	0.0680 ± 0.0366	0.2705 ± 0.2126	0.0584 ± 0.0369	0.1633 ± 0.1095

output of OUTPUT-AVG and OUTPUT-MODE in Algorithm 1 with a sufficiently large number of samples. We highlight some points that produce either the most accurate or most ambiguous results in Fig. 3 a) - c), and plot the probability distribution of the quantum states of these points below the corresponding figures, as shown by the bottom six figures.

Analysis. i) As compared between Fig. 3 a) - c), generally, the increase in the number of qubits t makes the QIP results approach the classical solution (ground truth – GT) thus improving the accuracy. With about $t = 8$ qubits, the results by QIP are almost as accurate as GT. **ii)** QIP-Avg and QIP-Mode show different output behaviors: QIP-Avg produces continuous values, while QIP-Mode produces discrete ones. This implies that QIP-Mode is not suitable for tasks that require precise output e.g., when sorting the inner products in K-means. **iii)** The bottom six plots show that the probabilities of the output basis states are often concentrated in one (Fig. 1 a/b/c.1)) or two quantum states (Fig. 1 a/b/c.2)) that yield the closest results to the ground-truth value. This implies a low probability of deviation from the ground truth and allows the solution accuracy to be enhanced by multiple measurements. **iv)** By comparing Fig. 1 a/b/c.1) with a/b/c.2), we find that the accuracy of QIP computation depends on the inner product values. For a value that matches a certain quantum state, the output of QIP is almost deterministic and very accurate. For example, in Fig. 1 a.1), $(\mathbf{x}, \mathbf{y}) = 0$ corresponds to the state $|1\rangle$ and the probability of measuring $|1\rangle$ as the output basis state is almost 1.

How sampling times r and number of qubits t influence the computing error. We generate 100K instances of inner products from the uniform distribution $(\mathbf{x}, \mathbf{y}) \sim \texttt{Uniform}(-1, 1)$ as the dataset $\mathcal{D}$, and simulate QIP by Algorithm 1. For a ground-truth $(\mathbf{x}, \mathbf{y})$ and the computed QIP $\widetilde{(\mathbf{x}, \mathbf{y})}$, we use the mean squared error (MSE) and the mean absolute error (MAE) as the metrics (see Appendix J.1 for definitions). We range the initial random seed from 0 to 9, and report the mean and standard deviation of the error of QIP in Table 4. **Analysis. i)** Both increasing number of qubits t and the sampling times r can improve the calculation accuracy and narrow the gap with the ground truth. **ii)** When the number of qubits t is low, the result of the OUTPUT-AVG strategy is more accurate. Due to the small number of basis quantum states, the calculated inner products by basis quantum states can be very sparse, therefore, the OUTPUT-MODE strategy is at a disadvantage. **iii)** When the number of qubits t is relatively high, the OUTPUT-MODE strategy has a greater advantage. Because there are a big number of basis quantum states, for each ground-truth inner product there are always one or two quantum states that can yield an inner product very close to the ground truth. The OUTPUT-MODE strategy is very likely to select this 'right' quantum state. However, the OUTPUT-AVG strategy may

Table 4. Error of QIP on a randomly generated dataset of 100k instances. **Recall that all the experiments in this paper are performed on a classic computer for simulation hence our experiments mainly check the correctness of our approach instead of the actual speedup.**

		number of qubits t=2		t=4		t=6		t=8	
	r	MSE ↓	MAE ↓	MSE ↓	MAE ↓	MSE ↓	MAE ↓	MSE ↓	MAE ↓
Avg	1	0.2665±0.0027	0.3830±0.0014	0.0624±0.0005	0.1356±0.0003	0.0156±0.0003	0.0448±0.0003	0.0038±0.0002	0.0138±0.0002
	3	0.1396±0.0003	0.2858±0.0003	0.0232±0.0001	0.0949±0.0002	0.0053±0.0001	0.0449±0.0003	0.0013±0.0000	0.0110±0.0001
	5	0.1141±0.0006	0.2661±0.0007	0.0152±0.0001	0.0816±0.0003	0.0033±0.0000	0.0291±0.0001	0.0008±0.0000	0.0099±0.0001
	7	**0.1034**±0.0005	**0.2588**±0.0006	0.0118±0.0001	**0.0743**±0.0002	0.0024±0.0000	0.0268±0.0001	0.0006±0.0000	0.0092±0.0001
Mode	1	0.2665±0.0014	0.3830±0.0010	0.0626±0.0010	0.1359±0.0008	0.0157±0.0004	0.0449±0.0003	0.0038±0.0002	0.0138±0.0001
	3	0.2032±0.0017	0.3419±0.0011	0.0364±0.0005	0.1074±0.0004	0.0080±0.0003	0.0322±0.0002	0.0019±0.0001	0.0094±0.0001
	5	0.1616±0.0010	0.3174±0.0006	0.0141±0.0002	0.0856±0.0002	0.0015±0.0001	0.0220±0.0001	0.0002±0.0000	0.0056±0.0000
	7	0.1462±0.0004	0.3076±0.0004	**0.0108**±0.0001	0.0814±0.0001	**0.0007**±0.0000	**0.0205**±0.0000	**0.0000**±0.0000	**0.0051**±0.0000

Table 5. Running time for 10k rounds of matrix multiplication of random matrices $\mathbf{A}$, $\mathbf{B}$ and the gradient backward propagation. Numbers in parentheses indicate speedup over the one w/o QSS.

	with QSS				without QSS			
	t=2	t=4	t=6	t=8	t=2	t=4	t=6	t=8
r=1	3.46 s (0.99x)	3.56 s (1.57x)	4.74 s (9.45x)	16.20 s (39.71x)	3.44 s	5.60 s	44.86 s	643.43 s
r=3	3.58 s (1.02x)	3.86 s (1.51x)	6.52 s (7.00x)	27.25 s (23.16x)	3.64 s	5.79 s	45.65 s	631.02 s
r=5	3.72 s (1.01x)	4.04 s (1.52x)	7.71 s (5.92x)	34.63 s (17.93x)	3.74 s	6.15 s	45.61 s	620.88 s
r=7	3.82 s (1.00x)	4.27 s (1.41x)	8.68 s (5.22x)	41.03 s (14.75x)	3.81 s	6.01 s	45.33 s	605.24 s

introduce the inner products obtained by 'wrong' quantum states, thus bringing more noise and losing some accuracy.

Efficiency study to demonstrate the significance of the proposed QSS algorithm. In Sec. 4.1, we have proposed to simulate output basis quantum states by introducing the quantum state search (QSS) algorithm, reusing the calculation results, and parallelizing the process. Among them, the QSS algorithm substantially reduces the computational complexity. Here we investigate the speedup brought by the QSS algorithm by comparing the running time of different scenarios with and without QSS. Concretely, we conduct 10k iterations of forward and backward propagation for the multiplication of two matrices $\mathbf{A} \in \mathbb{R}^{16\times128}$ and $\mathbf{B} \in \mathbb{R}^{128\times16}$ of random values, by the code implemented in Fig. 12. We use the Frobenius norm of $\mathbf{AB}$ as the loss, and OUTPUT-MODE as the default output strategy. The running time is recorded in Table 5. **Analysis. i)** By comparing the running time of different qubit numbers t, the running time of the cases without QSS increases with t much faster than that with QSS. This allows us to simulate the results of higher qubit number t in a reasonable time. **ii)** By comparing the running time of different sampling numbers r, it shows that whether to use QSS or not, increasing the number of sampling times will not incur a notable increase in the algorithm running time. This benefit is attributed to the reuse of the calculation results of the quantum state probabilities in simulation.

Comparison with the circuit simulator. We run 10 pairs of normalized random vectors of variant dimension d with the 1-to-1 QIP circuit, and compare both the accuracy and efficiency in Table 7. As d increases, the running time of circuit simulator increases exponentially while the running time of our state simulator keeps almost as a constant without sacrificing the computing accuracy. It demonstrates the necessity of our state simulator to realize the simulation of QIP in practical applications in ML models.

Table 7. Accuracy and efficiency comparison of the circuit simulator implemented in qiskit and our proposed state simulator (Sec. 4), with number of qubits $t = 6$, 32000 times of measurements for each run of QIP, and the OUTPUT-MODE strategy.

	Simulator	MSE ↓	MAE ↓	Running Time (s) ↓
$d = 4$	Circuit (qiskit)	4.31e-4	1.63e-2	7.072
	State (ours)	numerically the same		0.036, **196x faster**
$d = 16$	Circuit (qiskit)	7.64e-4	2.32e-2	114.502
	State (ours)	numerically the same		0.037, **3095x faster**
$d = 64$	Circuit (qiskit)	9.05e-4	2.53e-2	2388.785
	State (ours)	numerically the same		0.035, **68251x faster**

5.2. Applications in ML with Classic Simulation

Due to the prohibitively long running time and waiting time in using the currently available IBM's quantum computer and also the high complexity of the circuit simulator by qiskit, it is impractical to demonstrate the effectiveness of our circuits in computationally intensive machine learning applications. Therefore, we used the proposed efficient simulator we developed in ML applications.

Training Unitary Neural Networks for Image Classification. In training, the errors of one layer might be propagated through layers in forward/backward propagation. ProjUNN [17] defines a unitary layer in neural networks to maintain long-range stability, which could be highly related to quantum since all the quantum gates are unitaries. With our QIP, ProjUNN becomes a quantum model by implementing the unitary matrices as our 1-to-many or many-to-many QIP circuits. In the experiments, we embed the unitary layer in a three-layer MLP (see Appendix J.2), and conduct image classification on the MNIST dataset. We use ProjUNN-D as the projector of the unitary layer (experiments of ProjUNN-T are in Appendix K.1) and adopt RMSprop as the optimizer. In the simulation, we set the sampling times as $r = 5$. The convergence curve is plotted in Fig. 4, and the test accuracy after 20 epochs of training is given in Table 4.

We make the following analysis. **i)** In line with the conclusions in the accuracy study in the last section, when t is small ($t = 4$), the calculation error of OUTPUT-AVG is smaller and hence Q-4-Avg converges faster and has a higher test accuracy than Q-4-Mode. When t gets larger ($t = 6, 8$),

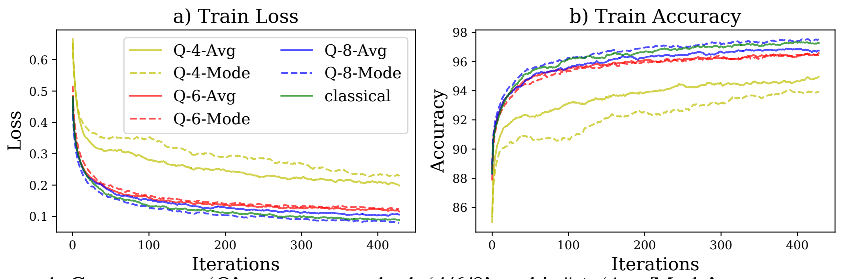

Figure 4. Convergence. ‘Q’: quantum method; ‘4/6/8’: qubit # t; ‘Avg/Mode’: output strategy.

Table 6. Test accuracy after 20 epochs on MNIST.

	Accuracy (%)
Q-4-Avg	93.70
Q-4-Mode	92.96
Q-6-Avg	94.65
Q-6-Mode	95.37
Q-8-Avg	95.54
Q-8-Mode	**96.13**
classical	95.97

Table 8. Results of unsupervised clustering by K-Means on MNIST. The higher the better.

Metrics	classical	Q-4-Mode	Q-4-Avg	Q-6-Mode	Q-6-Avg	Q-8-Mode	Q-8-Avg
RI $\uparrow$	**0.9318** $\pm$ **0.0096**	0.7866 $\pm$ 0.0079	0.8902 $\pm$ 0.0015	0.9244 $\pm$ 0.0084	0.9175 $\pm$ 0.0060	0.9295 $\pm$ 0.0111	0.9271 $\pm$ 0.0091
NMI $\uparrow$	**0.7314** $\pm$ **0.0156**	0.3473 $\pm$ 0.0262	0.4930 $\pm$ 0.0076	0.6962 $\pm$ 0.0233	0.6430 $\pm$ 0.0137	0.7300 $\pm$ 0.0229	0.7069 $\pm$ 0.0122
AMI $\uparrow$	**0.7211** $\pm$ **0.0191**	0.3159 $\pm$ 0.0263	0.4848 $\pm$ 0.0077	0.6858 $\pm$ 0.0253	0.6343 $\pm$ 0.0155	0.7185 $\pm$ 0.0268	0.6967 $\pm$ 0.0154

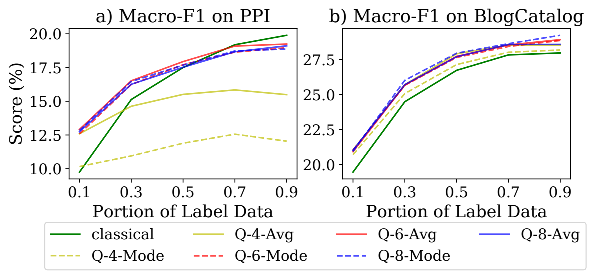

Figure 5. Node classification on PPI and BlogCatalog.

OUTPUT-MODE outperforms OUTPUT-AVG since less noise is introduced. **ii)** By comparing the results of different t, we see that generally higher qubit number t will yield a faster convergence speed and a better accuracy. **iii)** Surprisingly, we observe that the Q-8-Mode outperforms the classical implementation on both convergence speed and test accuracy. We speculate that this may be because the quantum method introduces some randomness that has a similar effect to the dropout strategy. **iv)** In our cases, the training of neural networks is robust to errors by quantum with sufficient qubits (e.g. $t = 8$), as shown by the comparison of the classical method and quantum counterparts.

Embedding Learning by Node2vec. Embedding learning is essential in ML, to obtain representations in continuous vector space for different raw data including discrete ones, e.g. natural language and graphs. We consider the network embedding algorithm node2vec [11] which is derived from word2vec [23]. We evaluate the node embeddings on the task of node classification. We set the sampling time $r = 5$. Experiments on node classification are conducted on a protein network: PPI [5], and BlogCatalog [30] which is a social network. We use Macro-F1 and Micro-F1 as the metrics. Results are given in Fig. 5 (full version in Fig. 14). **Analysis. i)** Cases of a high qubit number outperform cases of a low qubit number, and OUTPUT-AVG has better performance in low-qubit cases compared with OUTPUT-MODE. The observations are consistent with former experiments. **ii)** Surprisingly, on BlogCatalog, Q-4-Avg has better performance and comparable performance compared with the classical model. It indicates that the node2vec model can be highly robust to errors in some cases (BlogCatalog) even with a small number of qubits.

Unsupervised K-Means Clustering. K-means partitions data points into clusters. Yet K-means would intuitively be sensitive to calculation errors as the computed distances would be sorted and small errors can cause dis-ordering. We test on MNIST and use the rand index (RI), normalized mutual information (NMI), and adjusted mutual information (AMI) for evaluation. We set the sampling times $r = 7$. The scores are given in Table 8. **Analysis.** Unlike the previous two ML applications, K-Means with QIP fails to achieve comparable performance to the classical model even with sufficient qubits ($t = 8$). This suggests the limitations of QIP for error-sensitive tasks like K-Means.

6. Conclusions and Further Discussion

This paper focuses on the quantum inner product (QIP) problem, with the ultimate goal to accelerate classic ML on a quantum computer in the NISQ era. Covering the 1-to-1, 1-to-many, and many-to-many cases, we devise quantum circuits for QIP. Meanwhile, we devise a scheme that can efficiently simulate QIP, as done on a commodity classic computer with low memory. APIs have been encapsulated in the `torch_qip` package, with automatic differentiation enabled. In experiments, we show that the mean squared error of the approximated results by QIP can be smaller than 1e-4 with an 8-qubit output register and only 7 times of measurement. The performance of models that incorporate QIP improves in training neural networks, remains comparable in embedding, and declines slightly in K-Means.

Limitations. 1) The QIP operation is limited to normalized vectors as input data, which is also a common constraint in many quantum algorithms e.g. [15, 27]. 2) The impact of noise and decoherence process is often non-negligible in real-world quantum computer operation process when the number of quantum gates in the circuit becomes large.

References

[1] Hamed Ahmadi and Chen-Fu Chiang. Quantum phase estimation with arbitrary constant-precision phase shift operators. *arXiv preprint arXiv:1012.4727*, 2010. 4

[2] Esma Aïmeur, Gilles Brassard, and Sébastien Gambs. Machine learning in a quantum world. In *Conference of the Canadian Society for Computational Studies of Intelligence*, pages 431–442. Springer, 2006. 3

[3] Ville Bergholm, Josh Izaac, Maria Schuld, Christian Gogolin, Shahnawaz Ahmed, Vishnu Ajith, M Sohaib Alam, Guillermo Alonso-Linaje, B AkashNarayanan, Ali Asadi, et al. Pennylane: Automatic differentiation of hybrid quantum-classical computations. *arXiv preprint arXiv:1811.04968*, 2018. 2, 4

[4] Gilles Brassard, Peter Hoyer, Michele Mosca, and Alain Tapp. Quantum amplitude amplification and estimation. *Contemporary Mathematics*, 305:53–74, 2002. 2, 7

[5] Bobby-Joe Breitkreutz, Chris Stark, Teresa Reguly, Lorrie Boucher, Ashton Breitkreutz, Michael Livstone, Rose Oughtred, Daniel H Lackner, Jürg Bähler, Valerie Wood, et al. The biogrid interaction database: 2008 update. *Nucleic acids research*, 36(suppl_1):D637–D640, 2007. 8, 10

[6] Harry Buhrman, Richard Cleve, John Watrous, and Ronald De Wolf. Quantum fingerprinting. *Physical Review Letters*, 87(16):167902, 2001. 2

[7] G. Casati and G. Benenti. Quantum computation and chaos. In *Encyclopedia of Condensed Matter Physics*, pages 9–17. Elsevier, Oxford, 2005. 1

[8] Don Coppersmith. An approximate fourier transform useful in quantum factoring. *arXiv preprint quant-ph/0201067*, 2002. 1

[9] G Mauro D'Ariano, Matteo GA Paris, and Massimiliano F Sacchi. Quantum tomography. *Advances in imaging and electron physics*, 128:206–309, 2003. 2, 5

[10] Suguru Endo, Zhenyu Cai, Simon C Benjamin, and Xiao Yuan. Hybrid quantum-classical algorithms and quantum error mitigation. *Journal of the Physical Society of Japan*, 90 (3):032001, 2021. 1

[11] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *SIGKDD*, 2016. 8, 9

[12] Aram W Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. *Physical review letters*, 2009. 1, 3

[13] Sonika Johri, Shantanu Debnath, Avinash Mocherla, Alexandros Singk, Anupam Prakash, Jungsang Kim, and Iordanis Kerenidis. Nearest centroid classification on a trapped ion quantum computer. *npj Quantum Information*, 7(1):122, 2021. 3, 4

[14] Iordanis Kerenidis and Anupam Prakash. Quantum gradient descent for linear systems and least squares. *Physical Review A*, 101(2):022316, 2020. 4

[15] Iordanis Kerenidis, Jonas Landman, Alessandro Luongo, and Anupam Prakash. *Q-Means: A Quantum Algorithm for Unsupervised Machine Learning*. Curran Associates Inc., Red Hook, NY, USA, 2019. 8

[16] Iordanis Kerenidis, Jonas Landman, and Anupam Prakash. Quantum algorithms for deep convolutional neural networks. In *International Conference on Learning Representations*, 2020. 3, 4, 5

[17] Bobak Kiani, Randall Balestriero, Yann LeCun, and Seth Lloyd. projUNN: efficient method for training deep networks with unitary matrices. In *Advances in Neural Information Processing Systems*, 2022. 7, 9, 10, 11

[18] Hong Li, Nan Jiang, Zichen Wang, Jian Wang, and Rigui Zhou. Quantum matrix multiplier. *International Journal of Theoretical Physics*, 60:2037–2048, 2021. 2, 3

[19] Panchi Li and Bing Wang. Quantum neural networks model based on swap test and phase estimation. *Neural Networks*, 130:152–164, 2020. 2, 3

[20] Junhua Liu, Kwan Hui Lim, Kristin L Wood, Wei Huang, Chu Guo, and He-Liang Huang. Hybrid quantum-classical convolutional neural networks. *Science China Physics, Mechanics & Astronomy*, 64(9):290311, 2021. 1

[21] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. Quantum algorithms for supervised and unsupervised machine learning. *arXiv preprint arXiv:1307.0411*, 2013. 2

[22] Alessandro Luongo. quantumalgorithms.org - modified hadamard test. `https://quantumalgorithms.org/chapter-intro.html#modified-hadamard-test`, 2022. 2

[23] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26, 2013. 8

[24] Michael A Nielsen and Isaac Chuang. Quantum computation and quantum information, 2002. 2

[25] Hongyi Pan, Xin Zhu, Salih Furkan Atici, and Ahmet Cetin. A hybrid quantum-classical approach based on the hadamard transform for the convolutional layer. In *International Conference on Machine Learning*, pages 26891–26903. PMLR, 2023. 1

[26] Qiskit contributors. Qiskit: An open-source framework for quantum computing, 2023. 2, 4

[27] Patrick Rebentrost, Masoud Mohseni, and Seth Lloyd. Quantum support vector machine for big data classification. *Physical review letters*, 2014. 1, 8, 3

[28] Changpeng Shao. Quantum algorithms to matrix multiplication. *arXiv preprint arXiv:1803.01601*, 2018. 2, 3

[29] Mario Sracic. Quantum circuits for matrix multiplication, 2011. 2, 3

[30] Lei Tang and Huan Liu. Relational learning via latent social dimensions. In *SIGKDD*, 2009. 8, 10

[31] Yuan-Fu Yang and Min Sun. Semiconductor defect detection by hybrid classical-quantum deep learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2323–2332, 2022. 1

[32] Xin-Ding Zhang, Xiao-Ming Zhang, and Zheng-Yuan Xue. Quantum hyperparallel algorithm for matrix multiplication. *Scientific reports*, 6(1):1–7, 2016. 2, 3

[33] Jian Zhao, Yuan-Hang Zhang, Chang-Peng Shao, Yu-Chun Wu, Guang-Can Guo, and Guo-Ping Guo. Building quantum neural networks based on a swap test. *Physical Review A*, 100 (1):012334, 2019. 2, 3