A Software Architecting for Quantum Machine Learning Platform in Noisy Intermediate-Scale Quantum Era

Wenbin Yu (ywb@nuist.edu.cn)
Nanjing University of Information Science and Technology https://orcid.org/0000-0003-4786-4036
Yuhao Chen
Nanjing University of Information Science and Technology
Chengjun Zhang
Nanjing University of Information Science and Technology https://orcid.org/0000-0002-4458-5843
Yadang Chen
Nanjing University of Information Science and Technology
Hongyu Wei
Nanjing University of Information Science and Technology
Zongyuan Chen
Nanjing University of Information Science and Technology
Yifan Zhang
Nanjing University of Information Science and Technology

Research Article

Keywords: Quantum machine learning, Software architecture, Machine learning platform

Posted Date: November 13th, 2023

DOI: https://doi.org/10.21203/rs.3.rs-3562680/v1

A Software Architecting for Quantum Machine Learning Platform in Noisy Intermediate-Scale Quantum Era

Wenbin Yu[1,3], Yuhao Chen[1], Chengjun Zhang[2,3*], Yadang Chen[2], Hongyu Wei[2], Zongyuan Chen[2], Yifan Zhang[2]

[1]School of Software, Nanjing University of Information Science and Technology, No.219, Ningliu Road, Nanjing, 210044, Jiangsu, China.
[2*]School of Computer Science, Nanjing University of Information Science and Technology, No.219, Ningliu Road, Nanjing, 210044, Jiangsu, China.
[3]Jiangsu Collaborative Innovation Center of Atmospheric Environment and Equipment Technology (CICAEET), Nanjing University of Information Science and Technology, No.219, Ningliu Road, Nanjing, 210044, Jiangsu, China.

*Corresponding author(s). E-mail(s): zhangcj5@gmail.com (C.Z.);
Contributing authors: ywb@nuist.edu.cn(W.Y.); 202212210030@nuist.edu.cn (Y.C.);
adamchen@nuist.edu.cn(Y.C.);

Abstract

In the current Noisy Intermediate-Scale Quantum (NISQ) era, despite quantum computing is challenged by scale limitations and noise, it still holds immense potential to accelerate machine learning for specific problems. However, we have found that existing machine learning platforms do not adequately take into account the properties of quantum computing, resulting in a lack of an integrated quantum machine learning environment. Therefore, this article proposes a software architecture specifically designed for quantum machine learning platform, which aims to provide development tools that offer full life-cycle support for quantum machine learning. We first analyze the software requirements of quantum machine learning platform by using user story method, and then construct the platform architecture blueprint by using multi-view software architecture method. To validate the effectiveness of the architecture we proposed, we used this architecture as a specification to implement quantum reinforcement learning (QRL) and construct a quantum convolutional neural network (QNN). The results indicate that our architecture can effectively support the fusion of quantum computing and machine learning. We believe that this software architecture will propel the development of quantum machine learning in the NISQ era, bringing new opportunities to quantum computing.

Keywords: Quantum machine learning, Software architecture, Machine learning platform

1 Introduction

The physics revolution of the 20th century gave birth to two major scientific theories: relativity theory and quantum mechanics(Nickles 2009). They constitute the two major pillars of modern physics. As the cornerstone of the physical world, quantum mechanics, born in 1900, has brought about numerous significant technological

innovations, such as the invention of semiconductor transistors and laser technology(Schleich et al. 2016), which laid the foundation for many modern technologies. The achievements led to "The First Quantum Revolution"(Dowling and Milburn 2003), which is considered a revolution that is expected to change the future of technology and industry, and quantum computing also began to emerge during this period. In the late 20th century and early 21st century, researchers began to conduct in-depth studies on the theory of quantum computing. Many scientists attempted to explore whether quantum mechanics could be integrated with classical computing. At that time, scientists like David Deutsch, Paul Benioff, and Peter Shor began to discuss whether quantum systems could be effectively simulated on classical computers(Kitagawa 2002), and Paul Benioff wondered if it was possible to construct computer models that could calculate and operate according to quantum mechanics(Lloyd 2013). In Paul Benioff's work, he proposed the concept of the quantum Turing machine, which is similar to the classical Turing machine but incorporates quantum properties. He replaced each component and operation of the classical Turing machine with quantum bits (Qubits) and quantum gate operations from quantum mechanics(Mueller 2007). However, although the quantum Turing machine proposed by Paul can operate according to quantum mechanics, every calculation step it takes does not involve quantum entanglement, quantum interference, or other quantum properties, so it is still a classical calculation in essence(Benioff 1980). David Deutsch proposed his own ideas on the concept of a quantum Turing machine, stating that a quantum Turing machine must possess quantum properties, such as quantum superposition, quantum parallelism, and quantum randomness, among others(Deutsch 1985). At the same time, he pointed out the advantages of quantum algorithms, and foresight pointed out the significance of research on quantum algorithm complexity(Lipton and Regan 2014). These ideas have largely provided crucial inspiration for the development of quantum computing.

What truly sparked the quantum computing craze and piqued the interest of scientists outside the field of physics in quantum computing was the Shor algorithm, proposed by Peter Shor in 1994, for prime factorization(Monz et al. 2016). Before this, the prime factorization problem had always been in an awkward situation because no polynomial complexity algorithm had been found to address this problem. In other words, this problem cannot be solved efficiently on a classical computer. Howerer, Shor's algorithm tells us that through quantum computing, prime factorization problem can be solved using polynomial complexity algorithm. This immediately caused a stir in the academic community, as it was the first time that quantum computing demonstrated its potential in solving problems beyond simulating quantum environments, thus boosting scientists' research in quantum computing field. More and more problems begin to be solved efficiently through quantum computing, such as Bayesian reasoning(Low, Yoder, and Chuang 2014), least square fitting(Wang 2017)and perceptron(Kapoor, Wiebe, and Svore 2016), which are exponentially accelerated compared with classical computing. This potential for quantum computing to perform better than classical computing on specific problems is called quantum speedup(Rønnow et al. 2014).

In the mid-20th century, in the early days of artificial intelligence(AI) research, scientists began using computers to learn from data, model, and make predictions(Zhang and Lu 2021). The development of statistical learning theory provides a theoretical foundation for machine learning, and linear algebra, as an important mathematical foundation of statistics, naturally becomes an important tool for machine learning scientists to understand data, interpret data, and build models. With the rapid development of computing methods and devices, new machine learning methods are constantly emerging, such as deep learning based on neural networks(Schmidhuber 2015). However, classical machine learning faces some challenges when dealing with certain complex problems and large-scale datasets, such as computational complexity problems when dealing with large data(L'heureux et al. 2017) and dimensional disaster problems when handling high-dimensional datasets(Gnana, Balamurugan, and Leavline 2016). These challenges drive researchers to explore new methods to enhance the efficiency and performance of machine learning. As mentioned earlier, some quantum algorithms for machine learning have already demonstrated

the advantages of quantum speedup. Therefore, quantum mechanics is beginning to merge with the field of machine learning. Quantum properties can expedite linear algebra subroutines in machine learning(Ciliberto et al. 2018), implying that machine learning underpinned by mathematical tools from linear algebra can benefit from quantum computing to enhance computational efficiency. Moreover, quantum mechanics can generate non-classical patterns within data, indicating that quantum machine learning can reveal data patterns that classical machine learning cannot explore. All of these factors strongly attest to the tremendous potential of quantum machine learning.

With the advent of "The Second Quantum Revolution" era, the quantum technology has developed rapidly, among which quantum machine learning has also been fully developed. Especially after entering the Noisy Intermediate-Scale Quantum(NISQ) era, although quantum computers are limited by scale and noise, the hardware capability of quantum computers has made a certain breakthrough compared with traditional computers(Ding and Chong 2022). The researchers are working on quantum machine learning models suitable for the NISQ era, such as quantum deep learning. They replace classical neural networks with variational quantum circuits(VQC)(Qi, Yang, and P.-Y. Chen 2021)to serve as function approximators, and use classical optimization methods to avoid some problems encountered in NISQ era, so we also call the quantum machine learning in the NISQ era as quantum-classical hybrid machine learning. Data science engineers have tried to use it to solve some of the original machine learning problems, such as reinforcement learning, computer vision problems, and so on, and have achieved good performance on specific problems(De Luca 2022).

However, when developing quantum machine learning, algorithm engineers often encounter the same problems as in classical machine learning development, such as low model development efficiency, underutilization of computational resources, and difficulties in engineering tasks(Zhou et al. 2017). Although many of the giants in the industry have begun to invest significant resources in quantum computing, such as Google has launched a quantum computing platform called "Quantum AI"(Ho 2019), the quantum open-source framework "Cirq" and the quantum-classical hybrid machine learning framework "TensorFlow-Quantum"(Broughton et al. 2020) underneath it have become one of the preferred quantum computing frameworks for quantum algorithm engineers today. Xanadu, a pioneer in quantum computing and AI, has also launched PennyLane, an open-source quantum machine learning software to seamlessly integrate current quantum machine learning with quantum hardware(Bergholm et al. 2018). These quantum resources allow data engineers to harness the potential of quantum to solve AI problems, but they do not allow quantum machine learning development to form its own set of specifications. Existing machine learning platforms can solve these problems to some extent(Z. Ahmed et al. 2020), but most of them are geared toward classical machine learning, so none of them can support the efficient development of quantum-classical hybrid machine learning models with VQC as the core. Therefore, quantum machine learning engineers need a platform that is truly suitable for quantum machine learning development to achieve efficient quantum machine learning model development. However, the number of existing quantum machine learning platforms is still quite rare, because the development of a quantum machine learning platform not only requires software engineers to have knowledge reserves of software engineering, but also requires a certain understanding of quantum machine learning concepts. We just meet the above two conditions, and can use software engineering theory to solve the architecture design problem of quantum machine learning platform.

Software engineering focuses on building and maintaining software in an engineered way(Sommerville 2011), software architecture design is the creation of the "blueprint" for software, and as a result, it is included within the software engineering discipline. At present, the software engineering theory has a mature system, and the software architecture design has a clear definition. The ISO/IEC 42010 standard defines that an architectural architecture consists of the components of the system, the relationships between the components, and the basic principles(Júnior, Misra, and Soares 2019). Using software architecture design to describe

the platform is designed to get the development team to reach consensus quickly, and this goal requires a specific approach to achieve. Currently, there are many different methodologies for architecture design, such as the object-oriented RUP (4+1 views) method, the process-oriented structured analysis method, the enterprise architecture-oriented TOGAF method, and the decision-oriented ARC42 method(Ferrugento and Rocha 2015). Software architects have successfully applied these theories and architectural methodologies to design, construct, and develop software systems, including machine learning platforms. Although quantum machine learning is a novel interdisciplinary field, our quantum machine learning platform can also be described through architectural methodology. The software architecture design of the quantum machine learning platform can ensure that the platform possesses the capability for quantum machine learning development and provides standards and quality assurance for platform development.

The work of this paper is to use the methods and tools provided by software engineering to design the architecture of quantum machine learning platform to ensure the quality, maintainability and scalability of the system. Firstly, we conduct software requirement analysis for the quantum machine learning platform, which is generally the first step of software architecture design, and contains the business decomposition of the quantum machine learning life-cycle, which forms the basis of the software architecture. Software architecture must be designed to ensure that the system can meet the user's business needs. The next step is to translate requirements into the high-level structure of the system, and we document architectural details in the form of architecture diagrams. After comprehensively referring to a variety of architecture methodologies, we finally chose three software architecture views: logical view (describing the overall composition of the system), physical view (describing the mapping relationship between the software and hardware of the system) and business view (describing the business functions of the system) to describe the system in multiple dimensions, which can meet the needs of different stakeholders. We will emphasize the quantum components in quantum machine learning and redesign modules involving quantum on existing machine learning platforms, including quantum machine learning data services, quantum model development, and quantum software and hardware. We aim to build an all-in-one quantum machine learning development platform to assist users in rapidly creating quantum machine learning models and managing quantum machine learning workflows. The main work of this paper is as follows:

1. We use software engineering theory to complete the software architecture design for quantum machine learning platform.
2. We prove that quantum machine learning can be developed efficiently and perform well under the architecture proposed in this paper.

2 Software requirement analysis for quantum machine learning platform

The first step in the architecture design of quantum machine learning platform is to analyze the software requirements of the platform. Software requirements analysis is the analysis of what the system needs to achieve functionally to meet the business needs of users. This chapter is based on the quantum machine learning life-cycle and focuses on the software requirement analysis for the quantum machine learning platform.

Before conducting software requirements analysis, we first need to determine the desired software development model. From the well-known "Waterfall Model" proposed by Winston W. Royce (Ruparelia 2010)to the later Agile development model(A. Ahmed et al. 2010) and up to the current DevOps model(Ebert et al. 2016), software development models have evolved into various patterns. The traditional "Waterfall Model" is based on three assumptions: i) Users know exactly what they want. ii) Developers can fully understand what users want. iii) Requirements do not change during the development process. However, in fact, these three assumptions do not meet the actual situation, and the products made by the product manager after communicating with the users are often very different from the actual needs. Therefore, we adopt the DevOps model centered on the evolution of user requirements, which is more suitable for innovative projects with unclear requirements. At the same time, in "Waterfall Model",

requirements are usually documented in the form of specifications and use cases, but this approach is tedious, error-prone, and time-consuming to write, which is not adopted in the DevOps development model. Instead, we use stories to describe requirements. User stories are usually described from the user's point of view with a three-stage structure like: As a <Role>,I want to <Activity>, so that <Business Value>(Lucassen et al. 2016). Requirements are usually hierarchically split in the format of "Epic (an overall project, such as the quantum machine learning platform in this article) to Feature (features that are valuable to users under Epic, such as the quantum machine learning data service under this section) to Story (a user scenario breakdown of a feature)". Since the work in this paper is mainly to design the architecture of the quantum machine learning platform, we only adopt the "Epic - Feature" structure and give up the more detailed Story so that we can focus on the advanced structure of the quantum machine learning platform.

The construction process of quantum machine learning is basically the same as its classical counterpart. As shown in Fig. 1, the machine learning process is divided into three steps: data preparation, model development, and model deployment and application. Due to the quantum properties, quantum machine learning is different from classical processes in data processing and model construction, that is, the differences between the two are reflected in the model development part.

This section is divided into three subsections to analyze software requirements for data service, model development, and model deployment and application in quantum machine learning.

2.1 Data service in quantum machine learning

Data is an essential component of machine learning, which can be understood as a branch of statistics, naturally built upon data. In the context of machine learning, data refers to the content that neural networks or other artificial intelligence programs need to learn from. Algorithms extract relationships from the data and comprehend the patterns within it. The better the training data, the better the model's performance. If you need to improve the accuracy of a model, it requires a substantial amount of high-quality data. At this point, the data will also need to be more context-specific, fine-grained, and specialized. For example, different machine learning domains may require different data formats, encompassing various data types and labels. In such cases, a machine learning platform is needed to provide data management services. Similarly, in the emerging field of quantum machine learning, the significance of data is self-evident. The whole process of data service provided by a general machine learning platform is shown in Fig. 2, which is mainly divided into four stages: data creation, data processing, data labeling and data inference.

We write user stories based on process analysis of the data service of the quantum machine learning platform.Firstly, we propose epic on data preparation: As a user, I want to have a one-stop management of data in quantum machine learning, so that I can easily access data, improve the quality of data, and improve efficiency. We will break it down into levels and list the features in user stories about data services in quantum machine learning platform:

1. As a user, I want to have the ability to manage datasets , so that I can create the data needed for quantum machine learning.
2. As a user, I want to have the ability to process data , so that I can extract useful features from the data.
3. As a user, I want to have the ability to add labels to datasets , so that I can start my machine learning tasks.
4. As a user, I want to be able to manage the data after the inference of the model, so that I can evaluate whether the inference results are as expected.

2.2 Model development in quantum machine learning

The development of machine learning models is the most important part of the machine learning life-cycle, so the development of quantum machine learning models in the quantum machine learning platform has also become the focus of platform construction. At the same time, since this part is mostly combined with quantum properties and shows different properties from classical machine learning, we focus on the requirement analysis and corresponding architecture design of

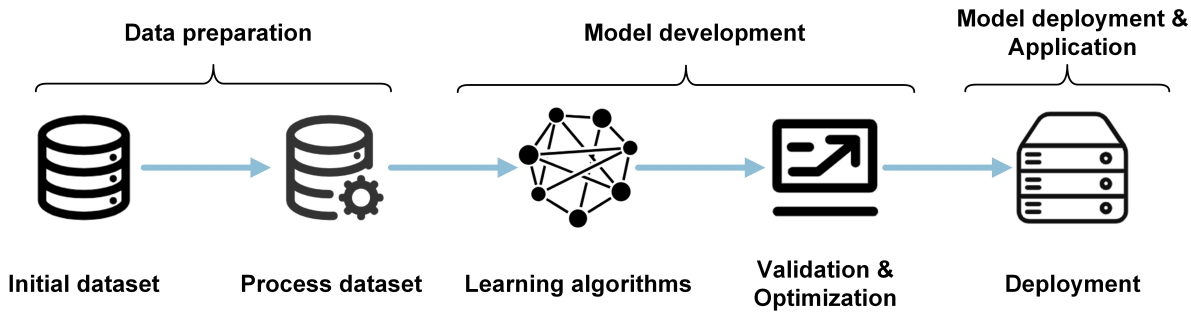


Fig. 1: The life-cycle of quantum machine learning

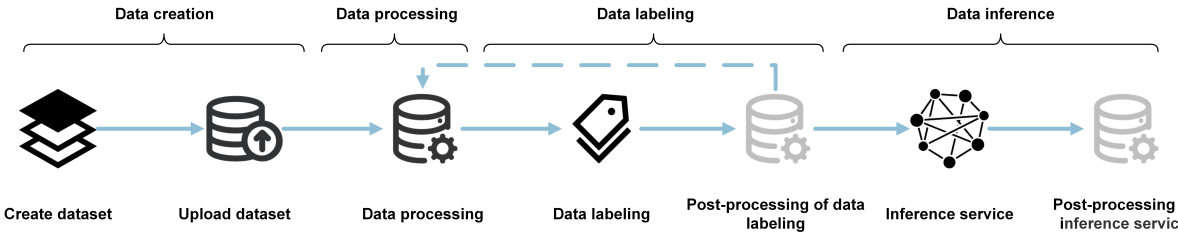


Fig. 2: The whole process of data service in quantum machine learning platform

this part. In contrast to the classical counterparts, the development of quantum machine learning models involves two stages: the development stage and the experimental stage. The development stage includes preparing and configuring the environment, choosing input datasets, and preparing the model. The experimental stage involves creating training tasks, monitoring training tasks, and evaluating and reviewing the training results. These two stages constitute the life-cycle of model development in a quantum machine learning platform. As shown in Fig. 3, in this whole stage, the quantum properties are mainly reflected in the model preparation part and the experimental stage, the former is related to the quantum variational circuit(VQC), the latter is combined with the quantum hardware. We'll look at these two parts separately.

The model presented in the development stage above refers to the quantum-classical hybrid machine learning model proposed in the NISQ era, which is a combination of quantum and classical models(Liu et al. 2021), and we mainly discuss quantum models here. Quantum model generally refers to Variational Quantum Circuit(VQC), that is, quantum circuit composed of quantum gates with differentiable parameters, which is one of the approaches to quantum machine learning. In many cases, quantum machine learning scientists also refer to VQC as quantum neural networks(Jeswal and Chakraverty 2019) in order to draw an analogy with neural networks in classical machine learning. VQC accept quantum data sets as inputs or quantum encoded classical data as inputs, transform the inputs using quantum gates with differentiable parameters, and then convert them into classical data by measurement. The measured data is fed into the classical model and the loss value of the model is calculated using the conventional loss function. Finally, the gradient of the model is calculated based on the value of the loss function and the model parameters are updated, which includes the parameters of the quantum model and the parameters of the classical model. The detailed process is shown in Fig. 4. The quantum neural network depicted in the figure mainly consists of three components: the encoding layer, the variational layer, and the measurement layer. The encoding layer is used to encode classical data into quantum data, and this operation is called "Embedding." Embedding layer involves taking input samples as parameters and optionally using parameter weighting to encode input features into

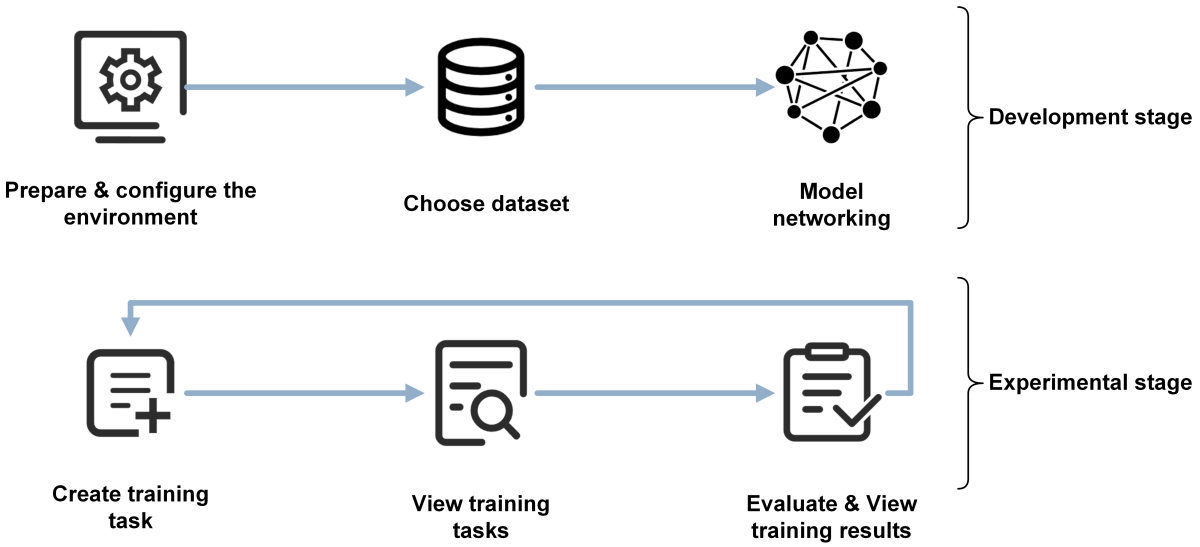


Fig. 3: The whole process of model development in quantum machine learning platform

the quantum state. The variational layer is a trainable circuit, typically referred to as an Ansatz. An Ansatz consists of a series of quantum gates acting on specific wires, with these gates containing freely differentiable parameters. Its structure is similar to that of classical neural networks, allowing us to define the gate types and free parameters as needed. The measurement layer consists of one or more observables, which act on specified circuits to obtain measurement expectations.

What is mentioned is that the concept of the quantum circuit layer mentioned above has similarities with the layers in classical neural networks, so we can also customize the quantum circuit structure according to the experimental needs, so the modular idea also provides convenience for the construction of quantum machine learning models. Users need to be free to model networking, that is, the construction of VQC. For example, there are many commonly used Embedding methods: AmplitudeEmbedding using amplitude coding, AngleEmbedding using Angle coding, and so on(Sun and Chan 2016). There are also different types of Ansatz, such as the StronglyEntanglingLayer for creating highly entangled quantum states, the RandomLayer for randomly generating quantum gates(McArdle et al. 2019), and the measurement layer can also choose different measurement bases. Different combinations of these layers give the VQC the ability to solve different problems. Therefore, users need the platform to have the ability of visual modeling, and realize the networking of quantum-hybrid machine learning models by dragging and dropping network components. In addition, interactive modeling of model networking through programming of multiple quantum AI frameworks is also necessary for the platform.

The quantum-classical hybrid machine learning models described above rely on quantum hardware to operate in a true quantum environment, and quantum data storage also requires support from quantum hardware. In the NISQ era, most hybrid machine learning models simulate quantum properties using quantum simulators and then execute them on classical computing hardware. However, in a quantum environment, this intermediate step is not necessary. Similar to the processors used in classical environments, such as CPUs and GPUs, quantum environments also have their dedicated processors called QPUs(Streif and Leib 2020). QPU is used to process quantum instructions on quantum computers, which are generally composed of low-level qubits and quantum gate sequences compiled from high-level quantum operations performed by users on the platform. We can understand that the working process of quantum computing system and classical computing system is roughly the same, but the quantum-classical hybrid model mentioned in the NISQ era requires the collaborative operation of quantum and classical devices, which requires

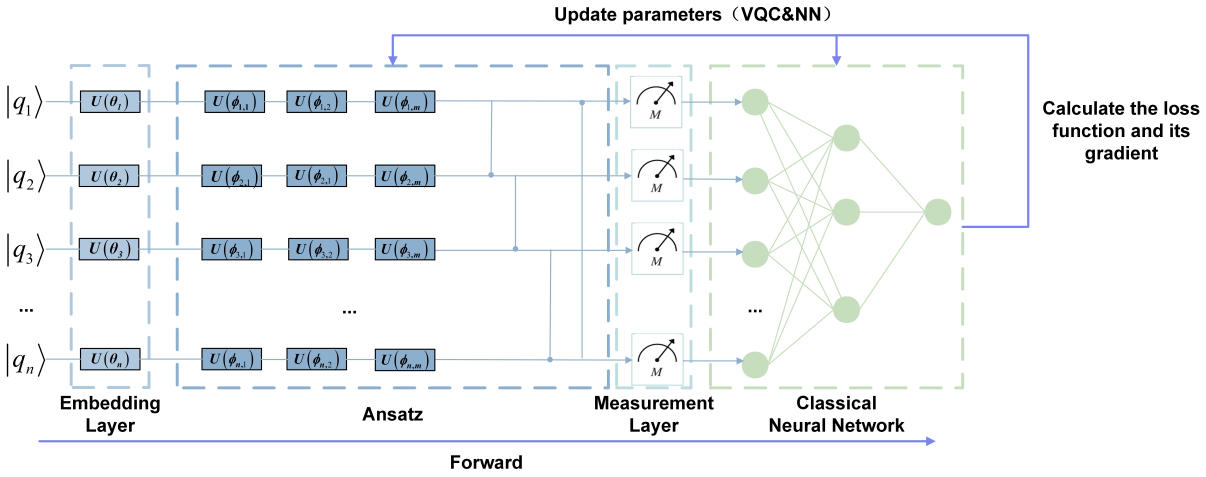


Fig. 4: The process of quantum-classical hybrid machine learning model

certain support from our platform. The corresponding architecture design for this part will be explained in the physical architecture view in Section 3.

With the foundation of VQC, we do the same as the previous subsection to analyze the user story requirements for the model development part of the quantum machine learning platform. With the foundation of VQC, we do the same as the previous section, the user story requirements for the model development part of the quantum machine learning platform first. We propose Epic in three stages:As a user, I want to have the ability to develop quantum machine learning models, so that I can conduct the development and experimental stages of the models. Next we break Epic down and list Stories:

1. As a user, I want to do some pre-development work (such as configuring the environment and selecting a dataset) , so that I can get started on the machine learning training task.
2. As a user, I want to be able to network the quantum machine learning model so that I can implement the model.
3. As a user, I want to be able to train the model , so that I can experiment with various combinations of algorithms, data, and hyperparameters.

2.3 Model deployment in quantum machine learning

After the completion of training a quantum machine learning model, users need to utilize the trained results in order to rapidly apply the acquired knowledge to unseen data; this is known as model inference. Following the development of a quantum machine learning model, within the quantum machine learning platform, users can create quantum AI applications and select a previously trained quantum AI model for swift deployment as a quantum AI application. Generally, there are two primary scenarios for deploying AI application services: online services and batch services. Online services involve deploying the model as a web service, enabling users to access the inference results through an API or console. Batch services perform inference on batch data until the batch data processing is completed. The process of deploying and applying quantum machine learning models to applications is illustrated in Fig. 5, showing the three main steps of AI model inference: creating AI applications, deploying AI applications, and inference services.

We propose a three-stage Epic for model deployment: As a user, I want to have the ability to deploy and apply a quantum machine learning model , so that I can use the model for inference. Then, we listed the Story:

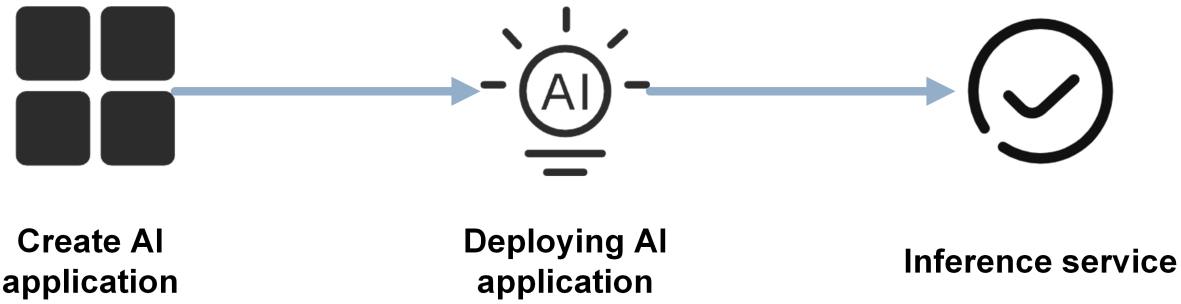


Fig. 5: The whole process of model deployment in quantum machine learning

1. As a user, I want to be able to import my model into the model repository of the quantum machine learning platform, so that I can build workable AI applications.
2. As a user, I want to deploy the AI application as a container instance in a resource pool, so that I can register it as an available service.
3. As a user, I want to have access to my deployed AI application, so that I can integrate AI reasoning capabilities into my business processes.

3 Architecture design for quantum machine learning platform

After conducting a requirements analysis for the quantum machine learning platform, the next step is to translate those requirements into the design. As artificial systems grow increasingly complex, the design, development, and maintenance of these systems pose certain challenges. The emergence of software architecture has greatly assisted software engineers in coping with the complexity of systems. By conceptualizing the architecture and describing the various components within it, software engineers can better understand the essential and critical attributes of the system in terms of behavior, composition, and evolution. This, in turn, allows them to focus on concerns such as the feasibility, practicality, and maintainability of the software.

The ISO/IEC 42010 standard is dedicated to the application of architecture descriptions to analyze, create, and maintain the architecture of systems. It provides a process for iterative construction of software architecture that spans the software development life-cycle. Within this standard, an architectural framework offers a structured approach to architectural design, serving as an organizational and methodological guide for designing software architecture(Lo and N. Chen 2017). Nowadays, there are several software architectural methodologies that propose architecture frameworks that meet the above-mentioned standards, such as Kruchten's "4+1" view model and the ppen group architecture framework (TOGAF model), among others. These architectural frameworks all encompass multiple architecture viewpoints and their corresponding architecture views, with each viewpoint focusing on specific aspects of the system. For example, the "4+1" architectural view model focuses on the system's logic, development, process, physical, and scene views, each of which has corresponding specific representations. These views help ensure that software architecture design can meet the needs of various stakeholders. Therefore, in order to design a blueprint for a quantum machine learning platform, we have integrated several existing software architectural methodologies, incorporating logical views, physical views, and business views, to ensure a comprehensive consideration of the business requirements, functionality, and deployment of the quantum machine learning platform. This contributes to the overall architecture of the quantum machine learning platform.

A logical view is a view used to describe the functional components within a system, focusing on the high-level functions and modules of the system without delving into specific implementation details. It is typically used to illustrate the composition of a system's functions and modules at a high level to show the system's structure. Fig. 6 is a software architecture design of the quantum

machine learning platform from the perspective of logical architecture. It decomposes the quantum machine learning platform into manageable components in a bottom-up sequence, going from low-level hardware to high-level interfaces, making development, maintenance, and testing easier. At the bottom is the heterogeneous resource layer of the platform, which contains the various hardware facilities of the platform, including the CPUs and the GPUs for classical computing, the QPU for quantum computing, and the memory used for quantum/classical(Q/C) storage. Above the resource layer is the basic platform, which manages various heterogeneous resources and provides various operation and maintenance services for the platform. Among them, the quantum simulator is deployed on the basic platform as a container service, which can simulate classical operations into quantum operations and then run on classical hardware, which lays the foundation for quantum-classical hybrid machine learning model to run on classical environment. The basic platform provides interfaces for AI services, including training and inference services, to enable task scheduling and resource allocation while being compatible with mainstream quantum AI frameworks. This helps users enhance resource utilization and job execution efficiency. At a higher level, it encompasses quantum machine learning life-cycle management and quantum AI asset management, integrating the key stages of data preparation, model development, model training, and model inference in quantum machine learning, abstracting away the underlying resource and environment management, meeting users' needs for rapidly building quantum machine learning models. The top layer of the platform is the tool layer, whose command line tools and console can quickly realize platform management and visual operations, and the SDK provided by the platform can also facilitate users to create custom quantum machine learning solutions.

Physical view refers to the description of how a system is deployed and executed in physical hardware and software. It focuses on the system's physical components and hardware devices to ensure the effective operation of the system. Fig. 7 represents the software architecture design of the quantum machine learning platform from a physical architecture perspective. The first part is the storage of Q/C data services. We already know that in classical computers, bits are the smallest unit of data storage and processing, and bit gates are used to perform data arithmetic and data processing operations. Similarly, in quantum computers, the equivalent of bits and logic gates are qubits and quantum gates. We will not go into details about their quantum properties here. The platform uses heterogeneous computing resources, mainly including CPU, GPU, and QPU. CPU and GPU are widely used accelerators in the field of machine learning, capable of significantly improving computational performance, and many machine learning platforms have already integrated them. In contrast, in quantum machine learning, the accelerator used is the QPU. QPU is used to manipulate quantum bits and perform quantum operations composed of quantum gates. It can create superposition states and entangled states of quantum bits and perform measurement operations on quantum bits after computation execution. The second and third parts of Fig. 7 provide an introduction to the compilation of advanced operations in the quantum machine learning platform into machine instructions. Since the construction of quantum-classical hybrid machine learning models involves both quantum models and classical models, along with preprocessing and post-processing of the models, these correspond to classical operations and quantum operations. Classical operations are naturally executed on classical hardware, while quantum operations can be divided into being executed in a simulated quantum environment or on real quantum hardware. The former involves simulating the evolution of quantum systems through quantum simulators and is ultimately processed using classical hardware devices, while the latter directly utilizes quantum hardware for quantum operations. The coexistence of classical and quantum operations requires the collaborative processing of classical hardware and quantum hardware, aligning with the needs of quantum machine learning in the NISQ era.

Business view is used to describe the business functions of a system, focusing on how the system meets business requirements and how business processes operate. Fig. 8 provides a business architecture design of the quantum machine

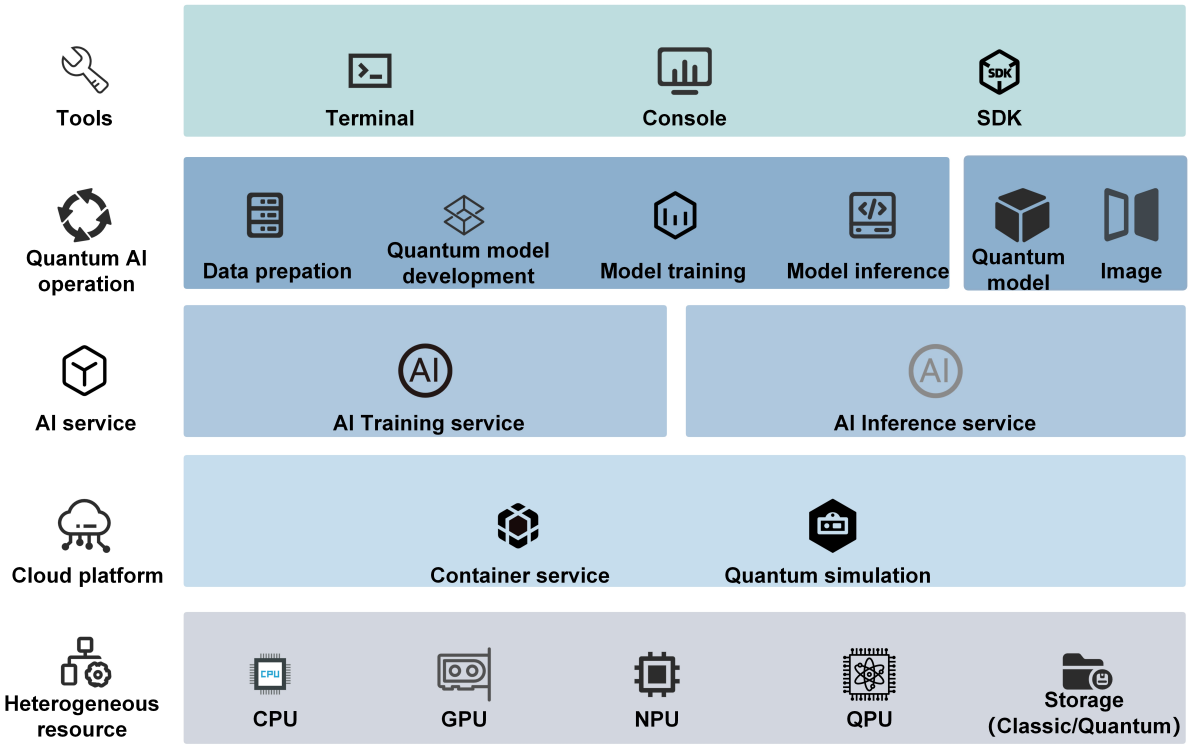


Fig. 6: Logic view of quantum machine learning platform

learning platform from a business function perspective. The business functions of the quantum machine learning platform have been thoroughly explained in Section 2. Therefore, based on our software requirements analysis, we conducted business design for the platform, mainly including modules such as Q/C data services, quantum machine learning model development, training tasks, deployment tasks, and AI asset management. Q/C data services mainly provide protection for data in the quantum machine learning life-cycle, including data set, feature engineering and data annotation services, to meet the needs of the quantum machine learning life-cycle from data preparation to post-inference data management. The model development module of quantum machine learning meets the needs of model development, including a variety of model modeling methods. Q/C visual Modeling services support drag-and-drop component modeling, and users can choose classic neural network or VQC components for model networking work on demand, without relying on code modeling. At the same time, users can also use the online IDE or upload custom images for code modeling to complement the visual modeling capabilities. The module provides preset machine learning templates and a variety of quantum AI computing frameworks, which is easy for users to quickly start developing. The training and deployment task module provides a pipeline service from machine learning training to deployment, so that users can quickly get through the quantum AI process. The AI asset management module provides managed services for data sets, models, images, tasks and code, along with the life-cycle of quantum machine learning, building the workflow of quantum AI. At the same time, the above major functional modules rely on the platform infrastructure and computing framework, which provide support for the implementation of the business.

4 The application of quantum machine learning platform

In the previous chapters, we have completed the software architecture design for the quantum-classical hybrid machine learning platform. To

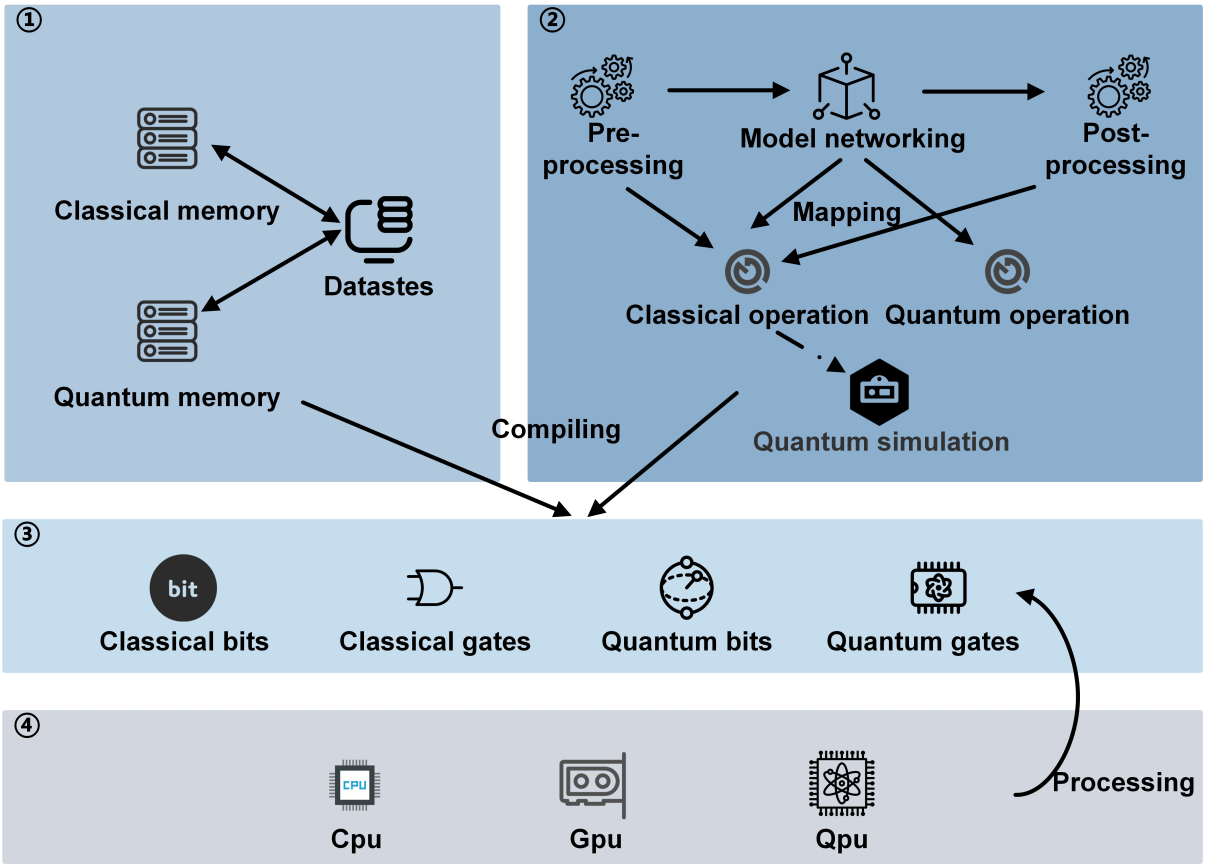


Fig. 7: Physical view of quantum machine learning platform

demonstrate the soundness of the architecture design, we have chosen to develop quantum-classical hybrid machine learning using the specifications outlined in the architecture design and validate its experimental results to prove the robustness of the architecture. We have selected quantum reinforcement learning (QRL)(Hamann and Wölk 2022) and quantum convolutional neural networks (QCNN)(Henderson et al. 2020) as example tasks for this purpose. Reinforcement learning is a type of machine learning that can naturally be improved by quantum algorithms. In reinforcement learning, agents are rewarded by constantly interacting with the environment for continuous iterative optimization. Various experiments have shown that the combination of quantum properties and reinforcement learning can optimize the agent's performance in the exploration environment. The quantum convolutional neural network uses VQC as the quantum convolutional kernel to extract features from images, while the other network layers maintain the structure of classical convolutional neural networks. Furthermore, these two mixed machine learning tasks employ a feedback loop between quantum exploration and classical optimization methods, which aligns with the definition of quantum-classical hybrid machine learning in the NISQ era.

Next, we are developing QRL according to the platform's specifications and the tools provided. We start with data preparation. QRL learns through interaction with the environment, so it does not rely on static datasets, and thus, users do not need to prepare datasets. However, the interactive environment it requires must be deployed within the platform's container service. This environment can be either a quantum or classical environment, and the data obtained can be stored on classical or quantum storage devices. We have

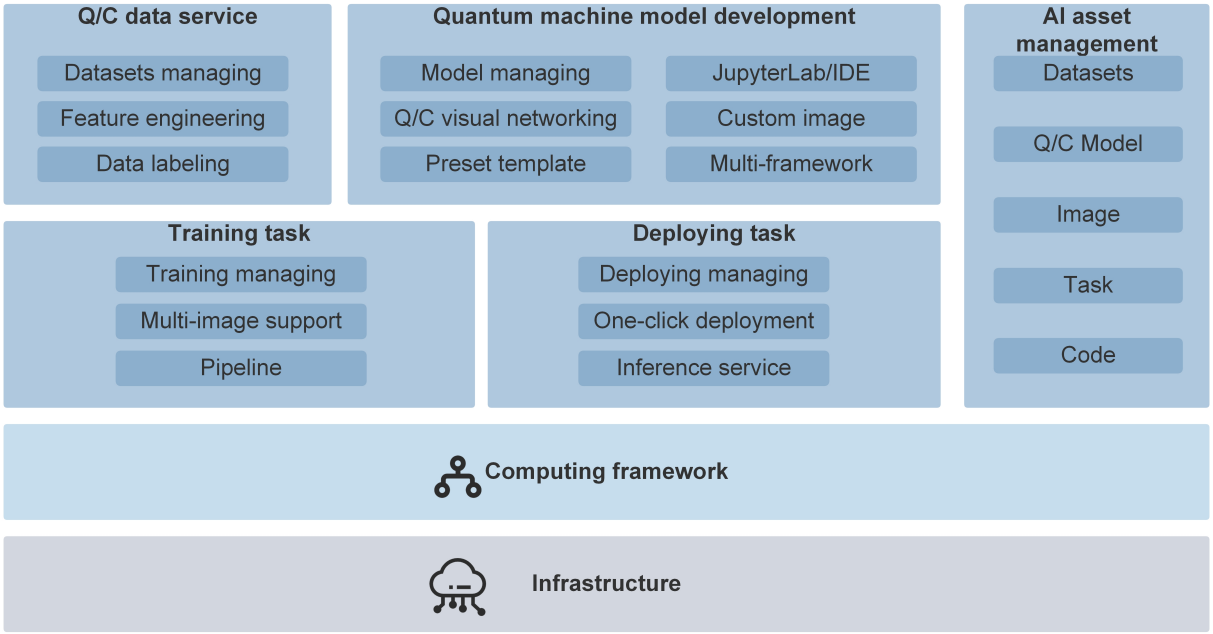


Fig. 8: Business view of quantum machine learning platform

chosen the classical environment CartPole as the interactive environment for reinforcement learning. This environment is an inverted pendulum, and the agent needs to continuously adjust the pendulum's angle to prevent it from falling. The longer the pendulum remains vertical, the higher the score the agent receives. Therefore, the task of Q-agent is to strive for high scores. After completing the data preparation, we will proceed with the model construction for the QRL task, which includes the networking of VQC and the preparation of classical optimization algorithms. We modularly constructed the Embedding Layer, Ansatz, and Measurement Layer in VQC following a visual modeling approach. We directly used templates for the preset circuits, employing AngleEmbedding, StronglyEntanglingLayer, and PauliMeasurement Layer as the components of VQC, as shown in Fig. 9. We applied the concept of model modularization from architecture design to modularize and componentize the networking of VQC in QRL. It can be observed that from quantum bits to quantum gates and the various layers within the quantum circuit, modular networking can be performed, facilitating efficient development of quantum models by algorithm engineers. Finally, we chose to implement VQC using the Pennylane quantum AI framework and leveraged its strong integration with Pytorch to implement classical RL algorithms. Simultaneously, we opted to train QRL task on a quantum simulator on a CPU, which only requires specifying the device during training tasks.

Similarly, we approach the construction of QCNN in a similar way to the completion of QRL. The structure of QCNN is shown in Fig. 10. We still choose to construct quantum circuit and classical neural network by presetting template. VQC plays the role of convolution kernel in QCNN. We choose AngleEmbedding as EmbeddingLayer and RamdomLayer as Ansatz. We measure each qubit separately on the Pauli z matrix and input the measured results to neural network. CNN is composed of pooled layer and fully connected layer, which are used to reduce feature dimension and improve the learning ability of the model.We used the MNIST dataset to verify the capabilities of QCNN.

Through Fig. 11, we can see that through the quantum-classical hybrid machine learning software architecture we designed, our quantum-classical hybrid machine learning task can be built and trained normally. After the training task is complete, we have the run result, as shown in Fig. 11. The experimental results show that Q-agent can get the highest reward in CartPole,

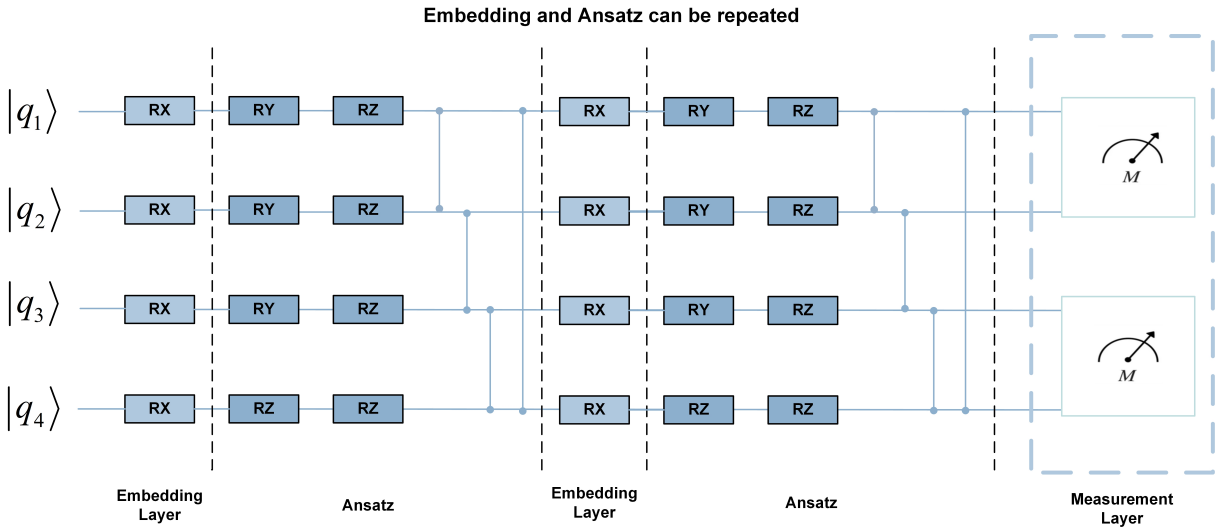


Fig. 9: Composition of RL-VQC in quantum machine learning

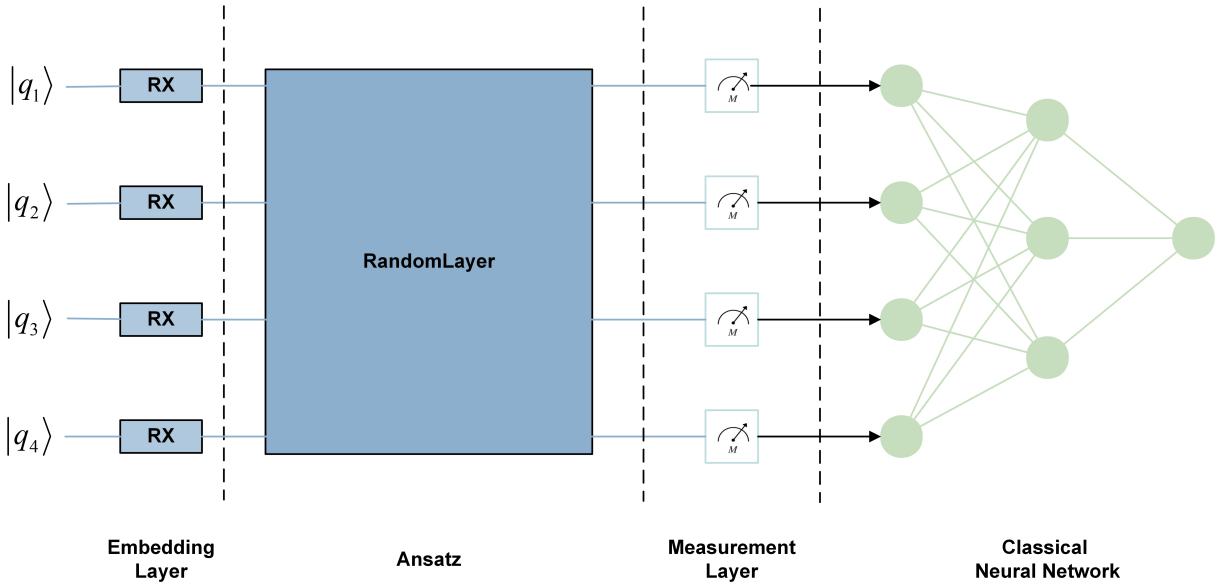
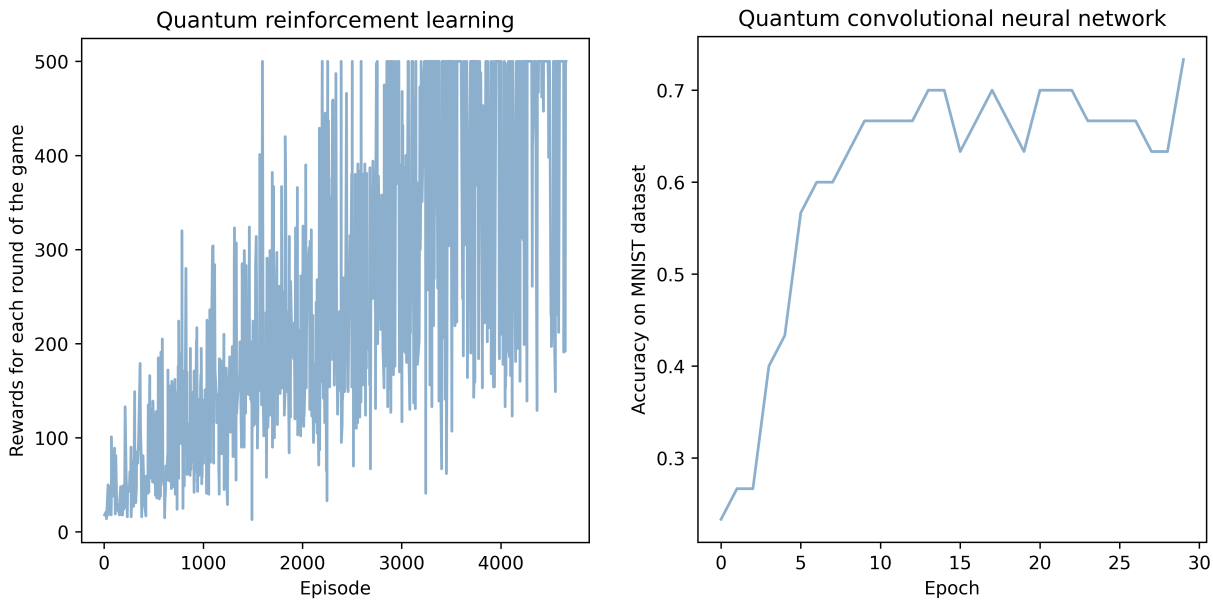


Fig. 10: Composition of QCNN in quantum machine learning

and QCNN also gets a good performance on the MNIST dataset. In other words, the software architecture of our quantum-classical hybrid machine learning platform can meet the test of practical tasks.

5 Conclusions

The aim of this paper is to design the software architecture of the AI platform suitable for quantum machine learning development in the NISQ era, so as to meet the needs of the quantum computing field. During the architecture design process, we completed a series of important tasks, including software requirements analysis of the



(a) Quantum reinforcement learning

(b) Quantum convolutional neural network

Fig. 11: Performance of quantum machine learning guided by software architecture

quantum machine learning platform using software engineering theory, multi-view architecture design, and the justification of the architecture using quantum machine learning tasks.

Firstly, we conducted a software requirements analysis for the quantum machine learning platform. We employed the user story method from the DevOps agile development model to assess the platform's user business requirements. Building upon a comprehensive analysis of the quantum machine learning process, we hierarchically decomposed the platform's key functional requirements, including data services, model development, and model deployment. Furthermore, we also took into consideration hardware requirements integrated with business functions to ensure the implementation of the quantum machine learning pipeline. Secondly, we integrated the "4+1" view model from the software architecture design framework and different architecture viewpoints from the TOGAF model to guide the architectural design of the quantum machine learning platform. We selected the logical view, which describes the core components of the system, the physical view, which details the deployment hardware of the platform, and the business view, which outlines the platform's business functions. The logical view abstracts the software and hardware modules of the quantum machine learning platform at a high granularity, reducing system complexity and enhancing the overall comprehensibility of the system. The logical view not only provides the foundation for architectural decisions, allowing architects and development teams to make rational divisions of labor between different logical components of the system, select appropriate components and technologies, but also provides a basis for code division within the development team, enabling parallel development and expediting project progress. The physical view offers insight into the physical deployment and operational environment of the quantum machine learning platform. It permits architects to plan hardware resources for the platform, including Q/C computing devices, Q/C storage devices, and other hardware components. Understanding the physical deployment of the system helps in comprehending the physical connections between various components, ensuring seamless communication between them. Moreover, the reliability of hardware devices contributes to the successful execution of quantum machine learning tasks.

The business view focuses on how the quantum machine learning platform meets business requirements and objectives to ensure its effectiveness at the business level. It integrates the content of software requirements analysis for the quantum machine learning platform, summarizing the business architecture of the platform, including data services, model development, training and deployment tasks, and AI asset management. This helps architects gain a deep understanding of the platform's business requirements, facilitating alignment between the system and the business to ensure competitiveness in a continuously evolving business environment. Finally, we validated the rationality of our quantum machine learning platform architecture design by running quantum machine learning example tasks.

In summary, this paper has proposed and implemented a software architecture for a quantum machine learning platform. Through in-depth software requirements analysis and a multi-view architecture design, the platform has effectively met the current needs in the field of quantum machine learning while providing a robust foundation for the future development of quantum machine learning applications. These efforts hold the potential to make significant contributions in the field of quantum computing and drive the advancement of quantum machine learning.

Declarations

Author Contributions All authors contributed to the study conception and design.

Funding This work was supported by the Natural Science Foundation of China (Grant number 62071240); the Natural Science Foundation of Jiangsu Province (Grant number BK20231142); Innovation Program for Quantum Science and Technology (Grant number 2021ZD0302900).

Data Availability The datasets analysed during the current study are available in http://yann.lecun.com/exdb/mnist/.

Conflict of interest The authors declare that there is no conflict of interest regarding the publication of this paper.

Human Participants and/or Animals This work does not contain any studies with human participants or animals performed by any of the authors.

Ethical approval Ethical approval is not applicable for this article.

Acknowledgment The authors would like to thank all of the people who helped with this work but did not qualify for authorship.

References

Ahmed, Adeel et al. (2010). "Agile software development: Impact on productivity and quality". In: *2010 IEEE International Conference on Management of Innovation Technology*. IEEE, pp. 287–291. ISBN: 1424465672.

Ahmed, Zeeshan et al. (2020). "Artificial intelligence with multi-functional machine learning platform development for better healthcare and precision medicine". In: *Database* 2020, baaa010. ISSN: 1758-0463.

Benioff, Paul (1980). "The computer as a physical system: A microscopic quantum mechanical Hamiltonian model of computers as represented by Turing machines". In: *Journal of statistical physics* 22, pp. 563–591. ISSN: 0022-4715.

Bergholm, Ville et al. (2018). "Pennylane: Automatic differentiation of hybrid quantum-classical computations". In: *arXiv preprint arXiv:1811.04968*.

Broughton, Michael et al. (2020). "Tensorflow quantum: A software framework for quantum machine learning". In: *arXiv preprint arXiv:2003.02989*.

Ciliberto, Carlo et al. (2018). "Quantum machine learning: a classical perspective". In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 474.2209, p. 20170551. ISSN: 1364-5021.

De Luca, Gennaro (2022). "A survey of NISQ era hybrid quantum-classical machine learning research". In: *Journal of Artificial Intelligence and Technology* 2.1, pp. 9–15. ISSN: 2766-8649.

Deutsch, David (1985). "Quantum theory, the Church–Turing principle and the universal quantum computer". In: *Proceedings of the Royal Society of London. A. Mathematical and*

Physical Sciences 400.1818, pp. 97–117. ISSN: 0080-4630.

Ding, Yongshan and Frederic T Chong (2022). *Quantum computer systems: Research for noisy intermediate-scale quantum computers*. Springer Nature. ISBN: 303101765X.

Dowling, Jonathan P and Gerard J Milburn (2003). "Quantum technology: the second quantum revolution". In: *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* 361.1809, pp. 1655–1674. ISSN: 1364-503X.

Ebert, Christof et al. (2016). "DevOps". In: *Ieee Software* 33.3, pp. 94–100. ISSN: 0740-7459.

Ferrugento, Adriana and Álvaro Rocha (2015). "Evolution of methodological proposals for the development of enterprise architecture". In: *New Contributions in Information Systems and Technologies: Volume 1*. Springer, pp. 351–359. ISBN: 3319164856.

Gnana, D Asir Antony, S Appavu Alias Balamurugan, and E Jebamalar Leavline (2016). "Literature review on feature selection methods for high-dimensional data". In: *International Journal of Computer Applications* 136.1, pp. 9–17. ISSN: 0975-8887.

Hamann, Arne and Sabine Wölk (2022). "Performance analysis of a hybrid agent for quantum-accessible reinforcement learning". In: *New Journal of Physics* 24.3, p. 033044. ISSN: 1367-2630.

Henderson, Maxwell et al. (2020). "Quanvolutional neural networks: powering image recognition with quantum circuits". In: *Quantum Machine Intelligence* 2.1, p. 2. ISSN: 2524-4906.

Ho, Alan (2019). "Next Generation Technology from Google AI Quantum". In: *APS March Meeting Abstracts*. Vol. 2019, p. V34. 002.

Jeswal, SK and S Chakraverty (2019). "Recent developments and applications in quantum neural network: A review". In: *Archives of Computational Methods in Engineering* 26, pp. 793–807. ISSN: 1134-3060.

Júnior, Ademir AC, Sanjay Misra, and Michel S Soares (2019). "A systematic mapping study on software architectures description based on ISO/IEC/IEEE 42010: 2011". In: *Computational Science and Its Applications–ICCSA 2019: 19th International Conference, Saint Petersburg, Russia, July 1–4, 2019, Proceedings, Part V 19*. Springer, pp. 17–30. ISBN: 3030243079.

Kapoor, Ashish, Nathan Wiebe, and Krysta Svore (2016). "Quantum perceptron models". In: *Advances in neural information processing systems* 29.

Kitagawa, Masahiro (2002). "Experimental quantum computation with molecules". In: *International Conference on Developments in Language Theory*. Springer, pp. 21–27.

L'heureux, Alexandra et al. (2017). "Machine learning with big data: Challenges and approaches". In: *Ieee Access* 5, pp. 7776–7797. ISSN: 2169-3536.

Lipton, Richard J and Kenneth W Regan (2014). *Quantum algorithms via linear algebra: a primer*. MIT Press. ISBN: 0262323575.

Liu, Junhua et al. (2021). "Hybrid quantum-classical convolutional neural networks". In: *Science China Physics, Mechanics Astronomy* 64.9, p. 290311. ISSN: 1674-7348.

Lloyd, Seth (2013). "The universe as quantum computer". In: *A Computable Universe: Understanding and exploring Nature as computation*, pp. 567–581.

Lo, Shun Chi and Ning Chen (2017). "IEEE 42010 and Agile Process-Create Architecture Description through Agile Architecture Framework". In: *Proceedings of the International Conference on Software Engineering Research and Practice (SERP)*. The Steering Committee of The World Congress in Computer Science, Computer ..., pp. 149–155.

Low, Guang Hao, Theodore J Yoder, and Isaac L Chuang (2014). "Quantum inference on Bayesian networks". In: *Physical Review A* 89.6, p. 062315.

Lucassen, Garm et al. (2016). "Improving agile requirements: the quality user story framework and tool". In: *Requirements engineering* 21, pp. 383–403. ISSN: 0947-3602.

McArdle, Sam et al. (2019). "Variational ansatz-based quantum simulation of imaginary time evolution". In: *npj Quantum Information* 5.1, p. 75. ISSN: 2056-6387.

Monz, Thomas et al. (2016). "Realization of a scalable Shor algorithm". In: *Science* 351.6277, pp. 1068–1070. ISSN: 0036-8075.

Mueller, Markus (2007). "Quantum Kolmogorov complexity and the quantum Turing machine". In: *arXiv preprint arXiv:0712.4377*.

Nickles, Thomas (2009). "Scientific revolutions". In.

Qi, Jun, Chao-Han Huck Yang, and Pin-Yu Chen (2021). "Qtn-vqc: An end-to-end learning framework for quantum neural networks". In: *arXiv preprint arXiv:2110.03861*.

Rønnow, Troels F et al. (2014). "Defining and detecting quantum speedup". In: *science* 345.6195, pp. 420–424. ISSN: 0036-8075.

Ruparelia, Nayan B (2010). "Software development lifecycle models". In: *ACM SIGSOFT Software Engineering Notes* 35.3, pp. 8–13. ISSN: 0163-5948.

Schleich, Wolfgang P et al. (2016). "Quantum technology: from research to application". In: *Applied Physics B* 122, pp. 1–31. ISSN: 0946-2171.

Schmidhuber, Jürgen (2015). "Deep learning in neural networks: An overview". In: *Neural networks* 61, pp. 85–117. ISSN: 0893-6080.

Sommerville, Ian (2011). "Software engineering (ed.)" In: *America: Pearson Education Inc.*

Streif, Michael and Martin Leib (2020). "Training the quantum approximate optimization algorithm without access to a quantum processing unit". In: *Quantum Science and Technology* 5.3, p. 034008. ISSN: 2058-9565.

Sun, Qiming and Garnet Kin-Lic Chan (2016). "Quantum embedding theories". In: *Accounts of chemical research* 49.12, pp. 2705–2712. ISSN: 0001-4842.

Wang, Guoming (2017). "Quantum algorithm for linear regression". In: *Physical review A* 96.1, p. 012335.

Zhang, Caiming and Yang Lu (2021). "Study on artificial intelligence: The state of the art and future prospects". In: *Journal of Industrial Information Integration* 23, p. 100224. ISSN: 2452-414X.

Zhou, Lina et al. (2017). "Machine learning on big data: Opportunities and challenges". In: *Neurocomputing* 237, pp. 350–361. ISSN: 0925-2312.

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- Highlights.pdf