Architectural patterns in NISQ-era quantum-enhanced AI systems

Mykhailo Klymenko and Thong Hoang

Quantum software engineering, as an emergent branch of software engineering, focuses on adapting established software design and development principles to enhance the quality of systems designed for quantum computing. The quality of a system is determined by its adherence to various quality attributes defined by the software quality standard IEEE: ISO/IEC 25002:2024 [1], which includes functional suitability, reliability, performance efficiency, usability, security, compatibility, maintainability, and portability. In this work, we present the findings from a systematic literature review (SLR) that investigates architectural patterns in current quantum AI systems. Architectural patterns, being general and reusable solutions to recurrent software architecture issues within a specific context [2, 3], are critical for optimizing these quality attributes. By identifying these patterns through evidence-based research, we aim to significantly broaden the knowledge base in software engineering.

The research data were gathered from databases including IEEEXplore, Nature Portfolio, ACM Digital Library, Springer, and GoogleScholar. Our initial search returned 6,822 research articles. After applying specified selection criteria and filtering the sources based on quality requirements, we collected 133 papers, which were then used to synthesize the presented research findings. We identifie four primary classes of architectural patterns in quantum AI systems, categorized by their role in the system architecture: the topology of the inference engine (quantum-classic split), the user-component interface, the model training process, and the execution environment.

The first category of patterns addresses the challenge of leveraging the quantum advantage of quantum machine learning models while ensuring they are deployable on current quantum computers. Contemporary Noisy Intermediate-Scale Quantum (NISQ) era hardware is limited by having fewer than a thousand qubits and short coherence times, which restrict the depth of quantum circuits. These limitations mean quantum components can only process small data volumes. To mitigate this, some systems use hybrid inference engines that combine quantum and classical computing elements. An example is the hybrid system depicted in the component model in Figure 1, where quantum inference is enhanced by classical neural networks.

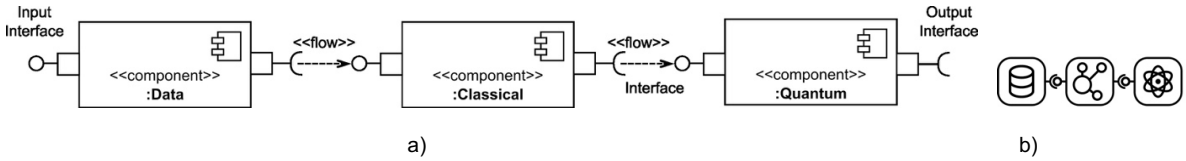

Fig. 1 a) Component diagram of a hybrid inference engine and b) its simplified representation.

The results of SLR reveals several quantum-classic split patterns illustrated in Fig. 2. The most prevalent pattern, termed quantum neural networks (QNNs) and depicted in Figure 2b, is predominantly featured in theoretical research. These studies typically propose a new model and serve as proof of concept, where data is directly input into the encoder of the quantum circuit. In contrast, more practical research, which is generally aimed at real-world applications, often utilizes what is known as the "quantum head" pattern, illustrated in Figure 2e. This approach involves processing high-dimensional data through classical, often pretrained, inference systems to distill essential features from raw data. These features are then used by a quantum parametrized circuit, which is trained to perform inference tasks based on these distilled features.

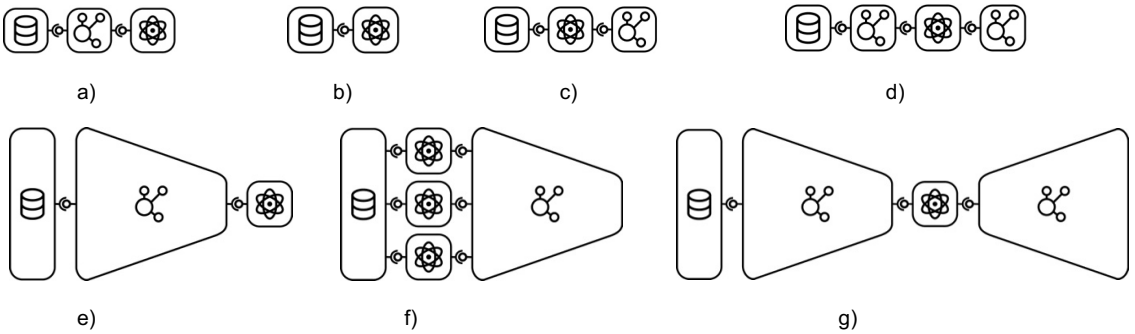

Fig. 2 Classic-quantum split architectural patterns: a) quantum head, b) QNN, c) quantum feature extraction, d) intermediate quantum layer, e) quantum head for large data, f) quanvolution, and d) quantum generative models.

References:

1. IEEE: ISO/IEC 25002:2024 https://www.iso.org/standard/78175.html
2. R.N. Taylor, N. Medvidović, E.M. Dashofy, Software architecture. Wiley, 2009.
3. L. Bass, P. Clements, and R. Kazman, Software architecture in practice. Addison-Wesley, 2003.