

GPT on a Quantum Computer

Yidong Liao^{1, 2, *} and Chris Ferrie^{1, †}

¹Centre for Quantum Software and Information,
University of Technology Sydney, Sydney, NSW, Australia

²Sydney Quantum Academy, Sydney, NSW, Australia

(Dated: March 14, 2024)

Large Language Models (LLMs) such as ChatGPT have transformed how we interact with and understand the capabilities of Artificial Intelligence (AI). However, the intersection of LLMs with the burgeoning field of Quantum Machine Learning (QML) is only in its nascent stages. This paper presents an exploration of this niche by detailing a comprehensive framework for implementing the foundational Transformer architecture — integral to ChatGPT — within a quantum computing paradigm. We meticulously design quantum circuits that implement adapted versions of the transformer’s core components and the generative pre-training phase. By integrating quantum computing with LLMs, we aspire to open new avenues for research in QML and contribute to the ongoing evolution of AI technologies.

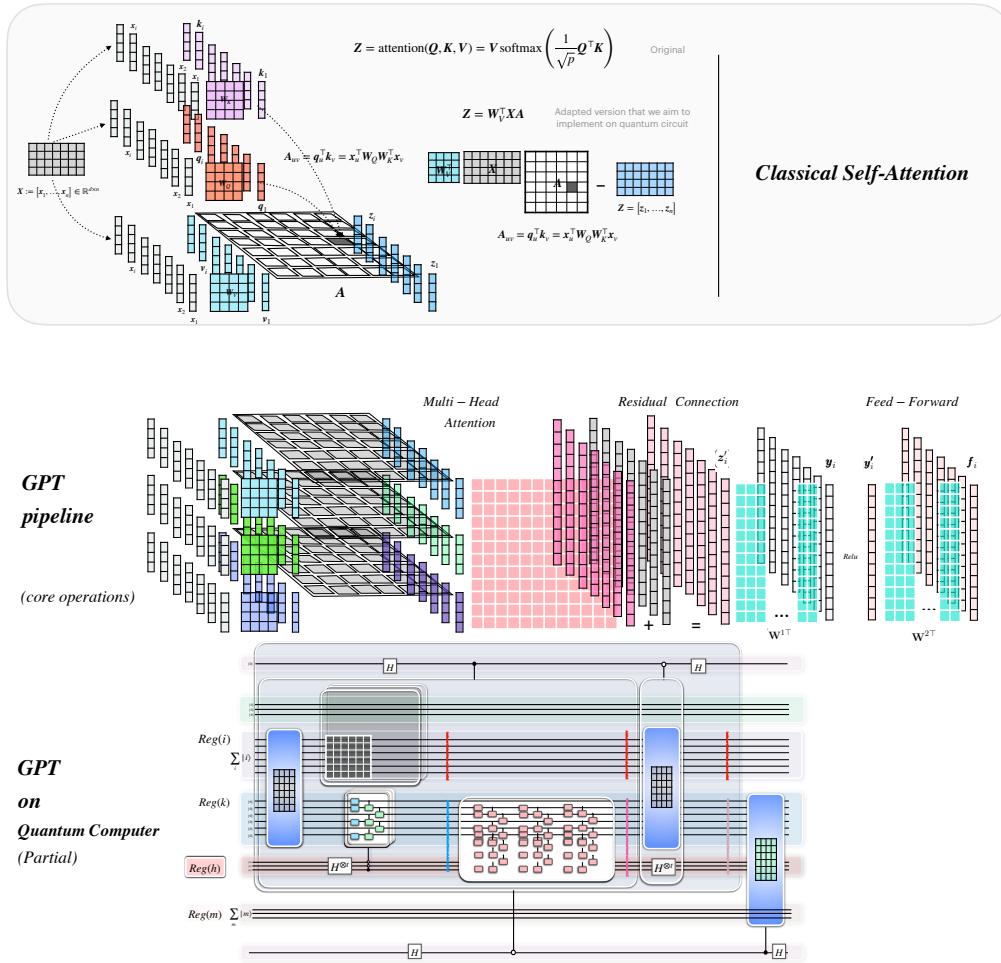


FIG. 1. Preview of some key contents in this paper. Graphical representations of the classical Self-Attention Mechanism, GPT pipeline, and overall quantum implementation of GPT.

* yidong.liao@student.uts.edu.au

† christopher.ferrie@uts.edu.au

Contents

1. Introduction	2
2. Background	3
2.1. Transformer Architecture Used in GPT	3
2.2. Language Modeling basics	5
2.2.1. Tokenization, Word Embedding	6
2.2.2. Next word prediction	6
2.2.3. Generative Pre-training	7
3. GPT on a Quantum Computer	8
3.1. Input encoding	8
3.2. Attentions on Quantum Computer	9
3.2.1. Self-Attention(single-head) on Quantum Computer	9
3.2.2. Multihead-Attention on Quantum computer	13
3.3. Residual-connection on Quantum computer	15
3.4. Feed-Forward Network on Quantum computer	17
3.5. Generative Pre-training on Quantum Computer	23
4. Conclusion	23
References	24
A. Quantum Attention Oracle	25
1. Evaluating Attention score in superposition	25
2. Storing Attention score	28
B. Positional encoding via quantum circuit	29
C. Masked-Attention	32
D. Block-encoding	33
E. Parametrized quantum circuit for implementing trainable linear layer	34

1. Introduction

The emergence and rapid advancement of Large Language Models (LLMs) such as ChatGPT has had a significant global impact, revolutionizing our interactions with artificial intelligence and expanding our understanding of its capabilities. Models like GPT-4 have demonstrated the vast potential of LLMs in a wide range of applications across various domains. In the field of natural language processing (NLP), LLMs have proven to be highly proficient in tasks such as machine translation, sentiment analysis, question answering, and text summarization. They excel in identifying intricate language patterns, comprehending context, and generating text that is coherent and contextually appropriate.

Concurrently, the field of quantum computing has seen remarkable progress, offering unprecedented computational power that promises to solve complex problems beyond the reach of classical

computing [1]. Quantum Machine Learning (QML), an intersection of quantum computing and machine learning, has emerged as a fertile ground for research, aiming to leverage quantum algorithms to enhance machine learning tasks [2]. Despite the significant achievements in QML, integrating quantum computing with state-of-the-art machine learning models, especially LLMs, is only at its nascent stages. Recent explorations such as Ref. [3, 4] indicate a burgeoning interest in leveraging quantum computing to elevate the capabilities of LLMs.

This paper delves into the quantum implementation of Generative Pre-trained Transformer (GPT) [5] (also referred to as GPT-1¹) — the original version of ChatGPT, focusing on adapting its architecture and pre-training processes to leverage the computational paradigms of quantum computing. We explore how quantum algorithms can be applied to the foundational components of GPT’s architecture and the generative pre-training, with the potential to enhance its efficiency and capabilities. By integrating quantum computing with LLMs, we aspire to open new avenues for research in QML and contribute to the ongoing evolution of AI technologies.

The rest of this paper is organized as follows: Section 2 provides a background on the transformer architecture used in GPT and the basics of language modeling. Section 3 details our approach to implement GPT’s architecture on quantum computers, including the quantum circuit designs for key model components and the generative pre-training. Section 4 concludes the paper with a summary of our findings and directions for future research.

2. Background

2.1. Transformer Architecture Used in GPT

The Generative Pre-trained Transformer (GPT) [5] is the inaugural version in the series of groundbreaking language models developed by OpenAI, marking the beginning of a new era in NLP. GPT’s architecture is predicated on the transformer model [6], a type of neural network that relies on self-attention mechanisms to process sequences of data, such as text. With 117 million parameters, GPT was a large model for its time, capable of capturing complex language patterns and generating coherent and contextually relevant text. GPT’s introduction was a pivotal moment in NLP; it paved the way for the development of more advanced models, such as GPT-2 and GPT-3, setting the stage for the rapid advancement of AI technologies in the years that followed. The remainder of this section gives an overview of GPT’s architecture and its training, and the next section 2.2 presents its application in language modeling.

GPT’s architecture is a multi-layer decoder-only Transformer [7] (a variant of the Transformer[6]). The primary part of the architecture is a stack of transformer blocks [5], each of which is composed of two main components: a (masked) multi-head self-attention mechanism followed by a position-wise fully connected feed-forward network. Layer Normalization and Residual Connections are placed around these two main components. The transformer blocks are stacked on top of each other, with each layer processing the output of the previous one. Prior to the input embedding entering the transformer blocks, positional encoding is added. Fig.2 illustrates these components in GPT’s architecture, the following paragraphs briefly² introduce each component.

¹ In this paper, we use "GPT" instead of "GPT-1."

² Here in this section, for each component in GPT’s architecture, we only give an overview, the detailed mathematical descriptions of each component are presented in Section 3.

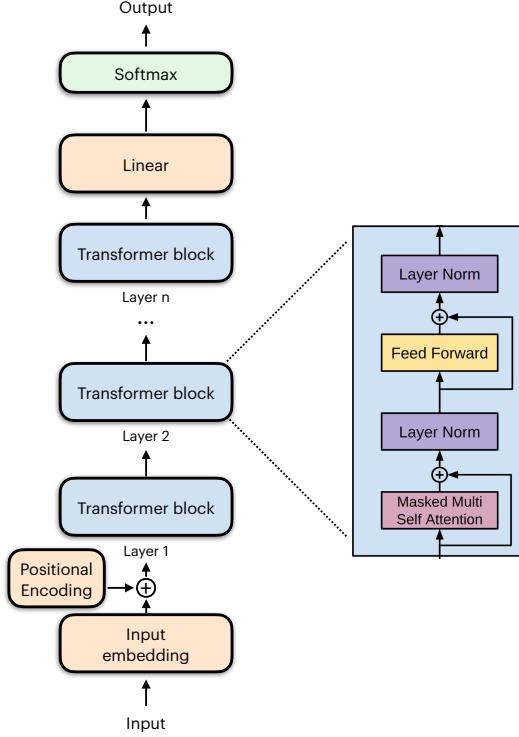


FIG. 2. *GPT’s Architecture, adapted from [5]*. GPT’s architecture is a multi-layer decoder-only Transformer [7] (a variant of the Transformer[6]). The primary part of the architecture is a stack of transformer blocks [5], each of which is composed of two main components: a (masked) multi-head self-attention mechanism followed by a position-wise fully connected feed-forward network. Layer Normalization and Residual Connections are placed around these two main components. The transformer blocks are stacked on top of each other, with each layer processing the output of the previous one. Prior to the input embedding entering the transformer blocks, positional encoding is added.

a. Multi-Head Masked Self-Attention Mechanism . The multi-head self-attention mechanism, with the addition of a masking operation, is a core component of the transformer block in GPT. This attention mechanism allows a model to weigh the importance of different words in a sentence. Unlike previous models (such as Recurrent Neural Networks(RNNs) [8]) that process words in a sequential manner, self-attention enables the model to look at all parts of the sentence simultaneously. This allows for a more nuanced understanding of context and relationships between words, regardless of their position in the sentence. The masking operation is a critical aspect of this mechanism, especially in the context of language modeling(a brief introduction is given in Section 2.2): it ensures that the prediction of a current word does not get influenced by future words. [9]

b. Layer Normalization and Residual Connections . Each transformer block in GPT includes layer normalization and residual connections. Layer Normalization is applied after the self-attention mechanism and after the feed-forward network within each transformer block. It normalizes the inputs across the features, improving the stability of the model. Residual Connections allow the input of each sub-layer (i.e., the self-attention and feed-forward networks) to be added to its output.

c. Position-Wise Fully Connected Feed-Forward Network . In each transformer block in GPT, after the attention mechanism together with corresponding Layer Normalization and Residual Connection, the output is passed through a feed-forward network that applies the same transformation to

each position separately and identically.

d. Positional Encoding . Since GPT (and transformer models in general) does not inherently process sequential data in order, it uses positional encodings to incorporate information about the order of the sequence into its inputs. These positional encodings are added to the input embeddings at the bottom of the model stack, providing the model with information about the position of each word in the sequence.

The training process of GPT consists of two main stages [5]: unsupervised pre-training and supervised fine-tuning. During pre-training, GPT is exposed to a large corpus of text data, learning the underlying structure of the language without any task-specific instructions. This stage allowed the model to develop a broad understanding of language, including grammar, semantics, and common phrases. The fine-tuning stage then adapted the pre-trained model to specific tasks, such as translation, question-answering, and summarization, by training it on smaller, task-specific datasets.

2.2. Language Modeling basics

This section briefly introduces one of GPT’s applications in language modeling — text generation, which is the foundation of the services provided by ChatGPT. We start by presenting an overview of language modelling:

Language modeling is a fundamental aspect of natural language processing (NLP) that focuses on the development of probabilistic models capable of understanding, generating, and interpreting human language. At its core, a language model predicts the likelihood of upcoming sequences of words occurring in a text [10]. This predictive capability enables a wide range of applications, from autocomplete systems in smartphones and email platforms to sophisticated chatbots, machine translation, speech recognition, and even content generation tools.

The essence of language modeling lies in capturing the statistical structure of language—learning how words and phrases tend to come together to convey meaning. Early language models were relatively simple n -gram models[11], where the prediction of the next word in a sequence depended on the previous $n - 1$ words. However, these models had limitations, particularly in dealing with long-term dependencies and the vast diversity of linguistic contexts. The advent of neural networks brought a significant leap forward in language modeling. Recurrent Neural Networks (RNNs)[8], and later, Long Short-Term Memory (LSTM) networks[12], provided mechanisms to remember information over longer sequences, improving the model’s ability to handle context in language. Despite these advances, RNNs and LSTMs also faced challenges, such as difficulty in training over very long sequences and capturing complex dependencies.

The breakthrough came with the introduction of the Transformer architecture[6], which led to the development of models like OpenAI’s GPT series that demonstrated unprecedented capabilities in generating coherent and contextually relevant text over extended passages. The development of language models continues to be a vibrant area of research in AI, with ongoing work aimed at improving their accuracy, efficiency, and ability to understand and generate human language in all its complexity.

Next, we delve into fundamental concepts of language modeling, such as tokenization, explaining how a language model (here, GPT) operates for text generation.

2.2.1. Tokenization, Word Embedding

Tokenization is the process of converting text into tokens which are the basic units of language models. There are three levels of tokenization:

- Character-level: Processes text one letter at a time.
- Word-level: Segments text into individual words.
- Subword-level: Breaks down words into subunits. For example, the subword tokenization of the phrase "Language Model" may look like ["Lan", "gu", "age ", "Mod", "el"].

Upon establishing the tokens for a language model, we arrange them into a structured vocabulary, assigning a distinct index to each token. These indices are then transformed into input features through various methodologies. Directly inputting these indices into the model is inadvisable, as the sequential order within the vocabulary does not inherently reflect semantic relationships. An alternative is the utilization of one-hot encoding. For a vocabulary encompassing 10,000 words, each word is symbolized by a 10,000-element vector, predominantly composed of zeros, save for the element corresponding to the word's indexed position. The primary benefit of one-hot encoding is its ability to preclude presumptions about word importance, facilitating the model's learning of word relationships during its training phase.

However, one-hot encoding presents scalability issues in the context of extensive vocabularies. Considering the English language, with its repertoire exceeding 100,000 words, representing a single word would necessitate a vector comprising 100,000 elements. In scenarios involving lengthy sequences, this approach demands substantial storage space and computational resources. To mitigate this issue of dimensionality, the use of word embeddings is proposed[13–16]. In this framework, an embedding layer projects the one-hot encoded tokens into a more condensed vector space. These embeddings, essentially denser token representations, are generated through a linear layer equipped with a weight matrix, which the model optimizes during its training process. Fig. 3 summarizes this section.

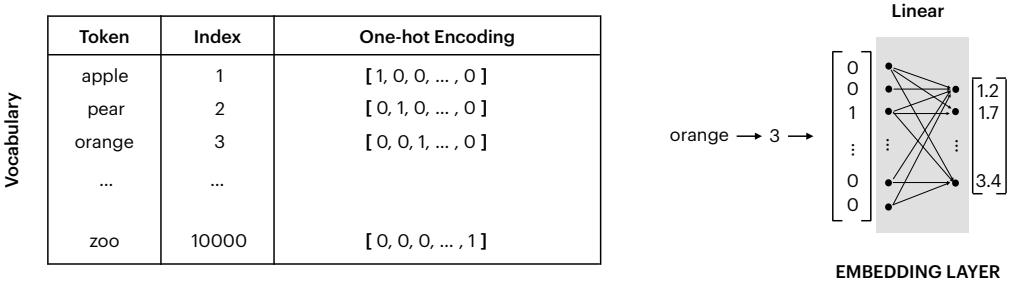


FIG. 3. Tokenization, one-hot encoding and word embedding.

2.2.2. Next word prediction

In the inference stage, a language model (here, GPT) takes in a sequence of one-hot encoded tokens, and generates predictions for the next word in a sequence.³ The sequence of one-hot encoded tokens

³ Here, we only consider autoregressive language models.

is first transformed through word embedding, as described in the above section. These embeddings, after the positional encodings are added, are then input into the transformer blocks. Then, a final linear layer is applied to map the outputs from the transformer blocks back into the vocabulary space, generating a sequence of transformed vectors. The last transformed vector in the sequence is referred as "logits". The logits are passed through a softmax activation function, yielding a probability distribution across the vocabulary, indicating the likelihood of each word as the next sequence component. Fig.4 summarizes this section.

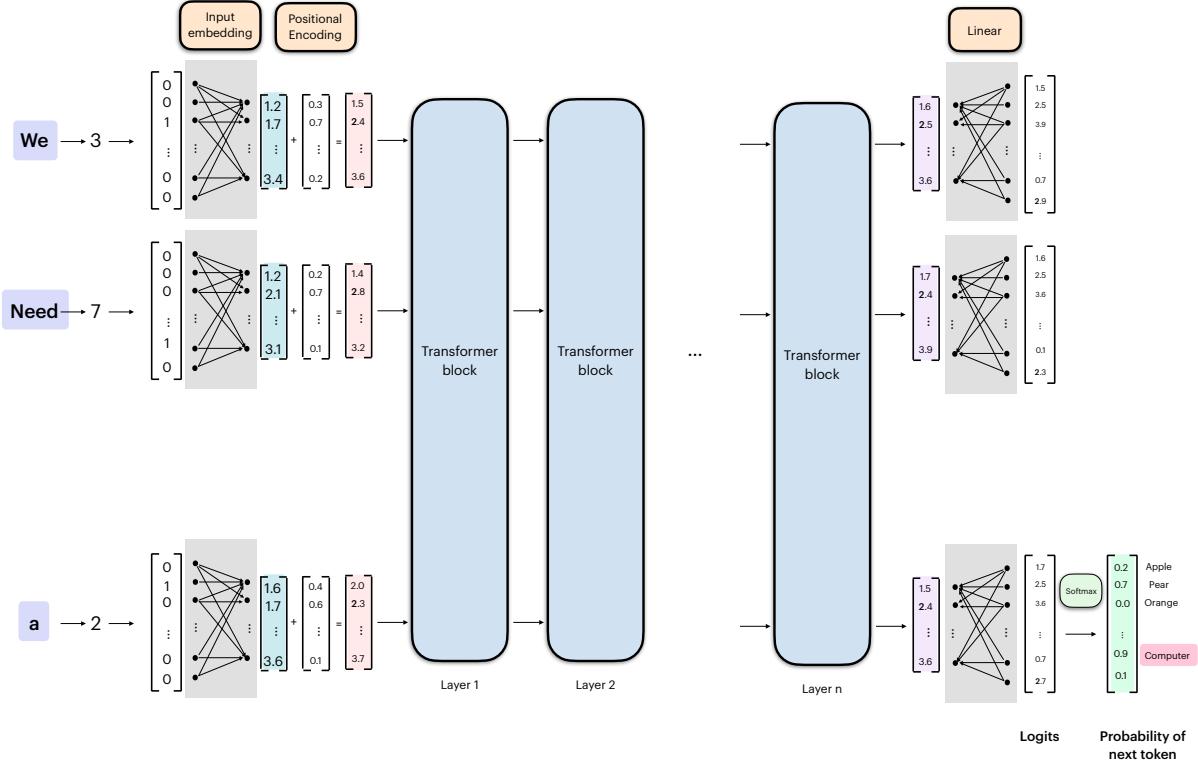


FIG. 4. *Next word prediction process of GPT*. In the inference stage, a language model (here, GPT) takes in a sequence of one-hot encoded tokens, and generates predictions for the next word in a sequence. The sequence of one-hot encoded tokens is first transformed through word embedding, as described in the above section. These embeddings, after the positional encodings are added, are then input into the transformer blocks. Then, a final linear layer is applied to map the outputs from the transformer blocks back into the vocabulary space, generating a sequence of transformed vectors. The last transformed vector in the sequence is referred as "logits". The logits are passed through a softmax activation function, yielding a probability distribution across the vocabulary, indicating the likelihood of each word as the next sequence component.

2.2.3. Generative Pre-training

GPT model undergo extensive pre-training on large text corpora using the following loss function:

$$L(\theta) = - \sum_t \log P(w_t | w_{1:t-1}; \theta) \quad (1)$$

where w_t is the t -th word, and θ represents the model parameters. This pre-training endows GPT with a broad understanding of language, which is then refined for specific tasks through fine-tuning.

Given the unlabeled nature of these sentences, this process is classified as unsupervised learning. It involves the pairing of a text segment as input with its subsequent segment as the target. The training process encompasses processing these input-target pairs in batches. The loss is computed by evaluating the next token in the target output, and this process is repeated for each subsequent token in the sequence. The cumulative loss is calculated across all batches, followed by the execution of backpropagation to adjust the model’s parameters.

The culmination of this process is a pre-trained language model, which we can then employ for text generation. This begins with the input of an initial word or phrase, serving as the genesis for text generation. The model assesses this input to predict the next token, which is subsequently reintroduced into the model as the new input. This iterative process engenders a feedback loop, enabling the model to generate continuous text sequences.

3. GPT on a Quantum Computer

As outlined in Section 2.1 and depicted in Figure 2, the architecture of GPT encompasses several key elements: input embedding, positional encoding, a series of transformer blocks, and a concluding linear layer followed by a softmax function. Within the context of this paper, it is assumed that both the input embedding and positional encoding are executed on classical computers. Our focus, however, shifts to detailing the implementation of the transformer blocks’ core components on a quantum computer. Additionally, we delve into the methodologies employed for executing Generative Pre-training of the model on a quantum computer. To ensure a holistic presentation, a detailed exposition on the quantum implementation of positional encoding is provided in Appendix B.

3.1. Input encoding

In this section, we focus on the process of input encoding for the quantum implementation of a Transformer block. As illustrated in Figure 4, the input to the Transformer block is a sequence of vectors $\{\mathbf{x}_i \in \mathbb{R}^d\}_{i=1}^n$ stacked as a matrix $\mathbf{X} := [\mathbf{x}_1, \dots, \mathbf{x}_n] \in \mathbb{R}^{d \times n}$. It can be encoded in a quantum state (after normalization⁴) $|\psi_{\mathbf{X}}\rangle$ as,

$$|\psi_{\mathbf{X}}\rangle := \sum_{i=1}^n |i\rangle |\mathbf{x}_i\rangle, \quad (2)$$

where $|\mathbf{x}_i\rangle := \sum_{k=1}^d \mathbf{x}_i^{(k)}|k\rangle$ is the amplitude encoding of the vector \mathbf{x}_i whose k -th elements are denoted as $\mathbf{x}_i^{(k)}$.

The entire state is prepared on two quantum registers hosting the index k and index i , which are denoted as $Reg(k)$ and $Reg(i)$, respectively. The unitary that realizes the data encoding as,

$$U_{\mathbf{X}} : |i\rangle |0\rangle \rightarrow |i\rangle |\mathbf{x}_i\rangle, \forall i = 1, \dots, n, \quad (3)$$

is represented as the blue box in Fig.6. $U_{\mathbf{X}}$ can be achieved by “Controlled Quantum State Preparation(CQSP)” process [17].

⁴ Note here and throughout the paper, we omit the normalization factors.

3.2. Attentions on Quantum Computer

As mentioned in Section 2.1, the multi-head self-attention, with the addition of a masking operation, is a core component of the transformer block in GPT. In this section, we present the quantum implementation of an adapted version of this component: Subsection 3.2.1 illustrates the adaptation of single-head self-attention and its implementation on quantum computers, while Subsection 3.2.2 discusses multi-head self-attention and its implementation on quantum computers. The masking operation and its quantum implementation are given in Appendix C.

In classical transformer architecture, the attention function can be described as mapping queries, keys, and values to an output, where the queries, keys, values, and output are all vectors [6]. The query \mathbf{q}_i , key \mathbf{k}_i , and value \mathbf{v}_i are p -dimensional, p -dimensional, and r -dimensional vectors defined as [9]:

$$\mathbb{R}^p \ni \mathbf{q}_i = \mathbf{W}_Q^\top \mathbf{x}_i, \quad (4)$$

$$\mathbb{R}^p \ni \mathbf{k}_i = \mathbf{W}_K^\top \mathbf{x}_i, \quad (5)$$

$$\mathbb{R}^r \ni \mathbf{v}_i = \mathbf{W}_V^\top \mathbf{x}_i, \quad (6)$$

where $\mathbf{W}_Q \in \mathbb{R}^{d \times p}$, $\mathbf{W}_K \in \mathbb{R}^{d \times p}$, and $\mathbf{W}_V \in \mathbb{R}^{d \times r}$ are trainable matrices.⁵ Similar to vectors $\{\mathbf{x}_i \in \mathbb{R}^d\}_{i=1}^n$ being stacked as a matrix $\mathbf{X} := [\mathbf{x}_1, \dots, \mathbf{x}_n] \in \mathbb{R}^{d \times n}$, we define $\mathbf{Q} := [\mathbf{q}_1, \dots, \mathbf{q}_n] \in \mathbb{R}^{p \times n}$, $\mathbf{K} := [\mathbf{k}_1, \dots, \mathbf{k}_n] \in \mathbb{R}^{p \times n}$, and $\mathbf{V} := [\mathbf{v}_1, \dots, \mathbf{v}_n] \in \mathbb{R}^{r \times n}$. From these definitions, we have

$$\mathbf{Q} = \mathbf{W}_Q^\top \mathbf{X}, \quad (7)$$

$$\mathbf{K} = \mathbf{W}_K^\top \mathbf{X}, \quad (8)$$

$$\mathbf{V} = \mathbf{W}_V^\top \mathbf{X}, \quad (9)$$

The "Scaled Dot-Product Attention" defined in [6] can be written in matrix form as:

$$\mathbf{Z}_0 := \text{attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathbf{V} \text{softmax} \left(\frac{1}{\sqrt{p}} \mathbf{Q}^\top \mathbf{K} \right), \quad (10)$$

where $\mathbf{Z}_0 = [\mathbf{z}_{01}, \dots, \mathbf{z}_{0n}] \in \mathbb{R}^{r \times n}$.

Note that for each i , the queries, keys, and values are all from the same vector \mathbf{x}_i in the sequence; this type of attention is referred to as the "self-attention" [9].

3.2.1. Self-Attention(single-head) on Quantum Computer

The "Scaled Dot-Product Attention" defined in Eqn. 10 can also be written as follows by plugging in Eqn. 9 and denoting $\mathbf{A}_0 \equiv \text{softmax} \left(\frac{1}{\sqrt{p}} \mathbf{Q}^\top \mathbf{K} \right)$:

$$\mathbf{Z}_0 = \mathbf{W}_V^\top \mathbf{X} \mathbf{A}_0 \quad (11)$$

⁵ The "transpose" in 6,5,4 are in the definition for some reason which will be clear in Section 3.4.

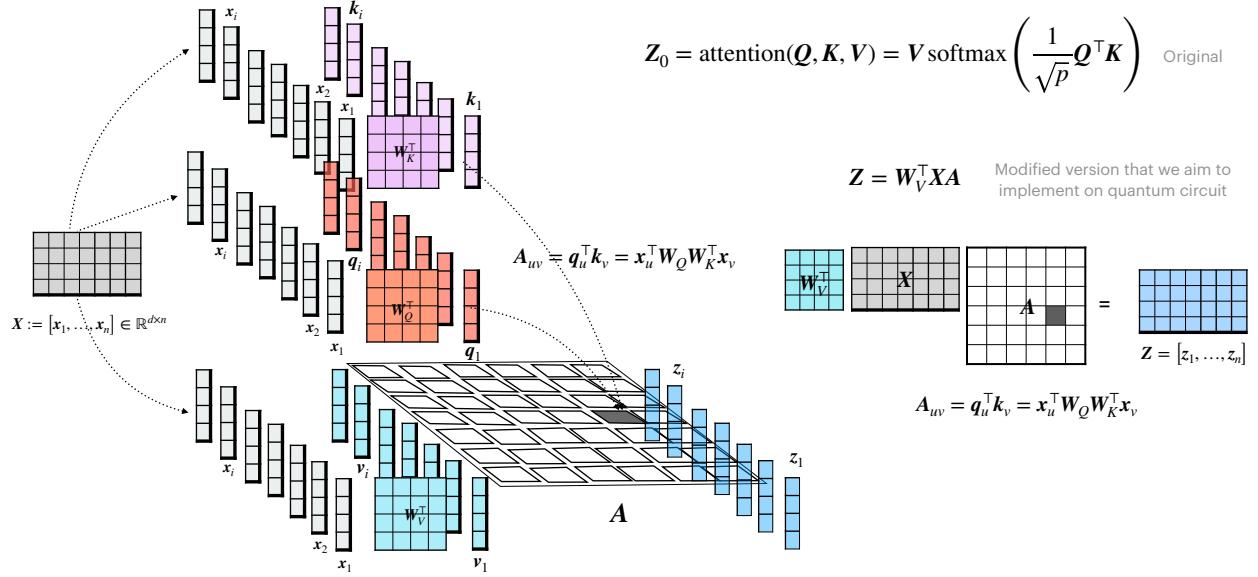


FIG. 5. *The Classical Self-Attention(modified version that we aim to implement on the quantum circuit)* As described in Ref. [6, 9], an attention function maps queries, keys, and values to an output, with all being vectors. The queries \mathbf{q}_i , keys \mathbf{k}_i , and values \mathbf{v}_i are p -dimensional, p -dimensional, and r -dimensional vectors, respectively, defined as: $\mathbb{R}^p \ni \mathbf{q}_i = \mathbf{W}_Q^\top \mathbf{x}_i, \mathbb{R}^p \ni \mathbf{k}_i = \mathbf{W}_K^\top \mathbf{x}_i, \mathbb{R}^r \ni \mathbf{v}_i = \mathbf{W}_V^\top \mathbf{x}_i$, where $\mathbf{W}_Q \in \mathbb{R}^{d \times p}, \mathbf{W}_K \in \mathbb{R}^{d \times p}$, and $\mathbf{W}_V \in \mathbb{R}^{d \times r}$ are the projection matrices. Similar to vectors $\{\mathbf{x}_i \in \mathbb{R}^d\}_{i=1}^n$ being stacked as a matrix $\mathbf{X} := [\mathbf{x}_1, \dots, \mathbf{x}_n] \in \mathbb{R}^{d \times n}$, we define $\mathbf{Q} := [\mathbf{q}_1, \dots, \mathbf{q}_n] \in \mathbb{R}^{p \times n}, \mathbf{K} := [\mathbf{k}_1, \dots, \mathbf{k}_n] \in \mathbb{R}^{p \times n}$, and $\mathbf{V} := [\mathbf{v}_1, \dots, \mathbf{v}_n] \in \mathbb{R}^{r \times n}$, respectively. The "Scaled Dot-Product Attention" defined in [6] can be written in matrix form as: $\mathbf{Z}_0 := \text{attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathbf{V} \text{softmax}\left(\frac{1}{\sqrt{p}}\mathbf{Q}^\top \mathbf{K}\right)$. Notably, for each i , the queries, keys, and values are derived from the same vector \mathbf{x}_i , a process referred to as "self-attention" [9]. Denote $\text{softmax}\left(\frac{1}{\sqrt{p}}\mathbf{Q}^\top \mathbf{K}\right)$ as \mathbf{A}_0 and substituting $\mathbf{V} = \mathbf{W}_V^\top \mathbf{X}$, we obtain $\mathbf{Z}_0 = \mathbf{W}_V^\top \mathbf{X} \mathbf{A}_0$. Considering the implementation of the softmax function via a quantum circuit is not straightforward and that scaling is managed in the block-encoding process, we aim to implement an alternative version of \mathbf{A}_0 , denoted as $\mathbf{A} = \mathbf{Q}^\top \mathbf{K}$, meaning we aim to design a quantum circuit to execute the computation: $\mathbf{Z} = \mathbf{W}_V^\top \mathbf{X} \mathbf{A}$. Here, $\mathbf{Z} = [z_1, \dots, z_n] \in \mathbb{R}^{r \times n}$. The matrix elements of \mathbf{A} are $A_{uv} = \mathbf{q}_u^\top \mathbf{k}_v = \mathbf{x}_u^\top \mathbf{W}_Q \mathbf{W}_K^\top \mathbf{x}_v$.

Considering it is not straightforward to implement the softmax function⁶ using quantum circuit and the scaling will be taken care of in the block-encoding⁷ procedure described later in this section, we aim to implement an alternative version of \mathbf{A}_0 denoted as $\mathbf{A} \equiv \mathbf{Q}^\top \mathbf{K}$, that is, we aim to design quantum circuit implementing the following computation:

$$\mathbf{Z} = \mathbf{W}_V^\top \mathbf{X} \mathbf{A}, \quad (12)$$

where $\mathbf{Z} = [z_1, \dots, z_n] \in \mathbb{R}^{r \times n}$.

⁶ Exploring alternatives to the softmax function in attention mechanisms has garnered interest due to the potential for efficiency gains and improved model performance. Research has demonstrated that it's possible to achieve high performance without the need for softmax normalization [18, 19].

⁷ Appendix.D provides a brief introduction of block-encoding.

Note that the matrix elements of \mathbf{A} are

$$\mathbf{A}_{uv} = \mathbf{q}_u^\top \mathbf{k}_v = \mathbf{x}_u^\top \mathbf{W}_Q \mathbf{W}_K^\top \mathbf{x}_v \quad (13)$$

The above description of classical attention (the modified version that we aim to implement on the quantum circuit) can be illustrated in Fig. 5. Next, we present its quantum implementation.

On a quantum circuit, after the input encoding described in 3.1, the attention function can be implemented by applying the block-encoding of \mathbf{A}^\top and a parameterized quantum circuit⁸ for \mathbf{W}_V^\top on the two quantum registers $Reg(i)$ and $Reg(k)$ respectively, as depicted in Fig. 6.

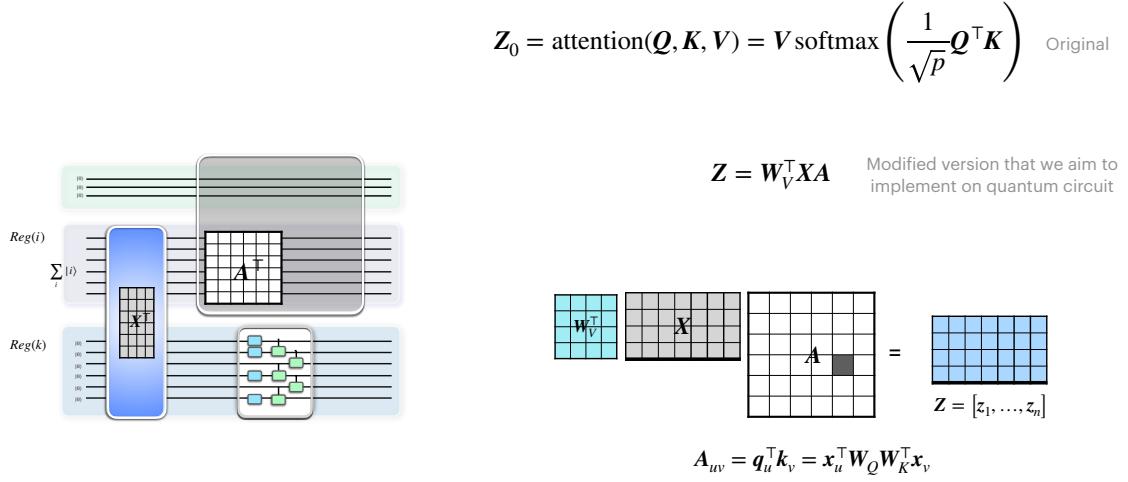


FIG. 6. *Self-Attention on Quantum Computer* The right side of the figure describes classical attention (the modified version that we aim to implement on a quantum circuit), and the left side of the figure depicts its quantum implementation. On the quantum circuit, the input encoding is represented by the blue box, as described in Section 3.1. The attention function can be implemented by applying the block-encoding of \mathbf{A}^\top and a parameterized quantum circuit for \mathbf{W}_V^\top on the two quantum registers $Reg(i)$ and $Reg(k)$ respectively.

This can be proven as follows:

Starting from Eq. 12, we have

$$\mathbf{Z} = \mathbf{W}_V^\top \mathbf{X} \mathbf{A},$$

Utilizing the formula $\text{vec}(ABC) = (C^\top \otimes A)\text{vec}(B)$, we obtain

$$\text{vec}(\mathbf{Z}) = \text{vec}(\mathbf{W}_V^\top \mathbf{X} \mathbf{A}) = (\mathbf{A}^\top \otimes \mathbf{W}_V^\top)\text{vec}(\mathbf{X}). \quad (14)$$

For a matrix M , defining vectors $\psi_M = \text{vec}(M)$ transforms Eq. 14 into

⁸ Appendix.E provides a brief introduction of using parametrized quantum circuit for implementing trainable linear transformations.

$$\psi_Z = (\mathbf{A}^\top \otimes \mathbf{W}_V^\top) \psi_X. \quad (15)$$

Recall Eq. 2:

$$|\psi_X\rangle = \sum_{i=1}^n |i\rangle |\mathbf{x}_i\rangle,$$

where writing the quantum states in Eq. 2 as vectors yields

$$\psi_X = |\psi_X\rangle. \quad (16)$$

Correspondingly, we have

$$\psi_Z = |\psi_Z\rangle, \quad (17)$$

by defining $|\psi_Z\rangle := \sum_{i=1}^n |i\rangle |\mathbf{z}_i\rangle$, where $|\mathbf{z}_i\rangle := \sum_{k=1}^d \mathbf{z}_i^{(k)} |k\rangle$ is the amplitude encoding of the vector \mathbf{z}_i with its k -th elements denoted by $\mathbf{z}_i^{(k)}$.

From Eq. 17 to Eq. 15, we derive

$$|\psi_Z\rangle = (\mathbf{A}^\top \otimes \mathbf{W}_V^\top) |\psi_X\rangle. \quad (18)$$

This corresponds to the action of the quantum circuit for self-attention, which can be represented as

$$|\psi_Z\rangle \otimes |0\rangle + \dots = (U_{\mathbf{A}^\top} \otimes U_{\mathbf{W}_V^\top})(|\psi_X\rangle \otimes |0\rangle), \quad (19)$$

where $U_{\mathbf{A}^\top}$ corresponds to applying the block-encoding of \mathbf{A}^\top , and $U_{\mathbf{W}_V^\top}$ is a parameterized quantum circuit implementing \mathbf{W}_V^\top on the two quantum registers $Reg(i)$ and $Reg(k)$ respectively. The term " $+\dots$ "⁹ indicates that upon post-selecting, we can obtain the desired state $|\psi_Z\rangle = \text{vec}(\mathbf{Z})$.

The block-encoding of \mathbf{A}^\top can be constructed using the following lemma from Ref. [20].

Lemma 3.2 from Ref. [20] (*Naive block-encoding of dense matrices with oracle access*). Let $A \in \mathbb{C}^{N \times N}$ (where $N = 2^s$) with a_{ij} being its elements and let $\hat{a} \geq \max_{i,j} |a_{ij}|$. Suppose the following oracle is provided

$$O_A : |i\rangle |j\rangle |0\rangle^{\otimes b} \rightarrow |i\rangle |j\rangle |\tilde{a}_{ij}\rangle,$$

where $0 \leq i, j < N$ and \tilde{a}_{ij} is the (exact) b-qubit description of a_{ij}/\hat{a} . Then one can implement a $(N\hat{a}, s+1, \epsilon)$ -block-encoding of A with two uses of O_A , $O(\text{polylog}(\hat{a}N/\epsilon))$ one- and two-qubit gates and $O(b, \text{polylog}(\hat{a}N/\epsilon))$ extra qubits (which are discarded before the post-selection step).

Below we explain how the block-encoding of $\mathbf{A}^\top \in \mathbb{R}^{n \times n}$ can be constructed using the above lemma:

⁹ Throughout this paper, the terms " $+\dots$ " in the quantum states are consistently used as defined here: " \dots " represents a quantum state that is orthogonal to the state before the "+" sign.

From Eqn.13, the matrix elements of \mathbf{A}^\top (which we denote as Λ_{ij}) are

$$\Lambda_{ij} := \mathbf{A}^\top_{ij} = \mathbf{x}_j^\top \mathbf{W}_Q \mathbf{W}_K^\top \mathbf{x}_i \quad (20)$$

let $\hat{\Lambda} \geq \max_{i,j} |\Lambda_{ij}|$ and $\tilde{\Lambda}_{ij}$ is defined to be the (exact) b-qubit description of $\Lambda_{ij}/\hat{\Lambda}$.

According to the above lemma, we need the following oracle to the block-encoding of \mathbf{A}^\top

$$O_{\mathbf{A}^\top} : |i\rangle|j\rangle|0\rangle^{\otimes b} \rightarrow |i\rangle|j\rangle|\tilde{\Lambda}_{ij}\rangle, \quad (21)$$

where $0 \leq i, j < n$.

This oracle $O_{\mathbf{A}^\top}$ can be constructed in the same way as constructing $O_{\text{attention}}$ (described in Appendix A) defined in Eqn.A1 with the attention score as Eqn.A2. Therefore, substituting A in the above lemma with \mathbf{A}^\top , we can implement the block-encoding of \mathbf{A}^\top with two uses of $O_{\mathbf{A}^\top}$, $O(\text{polylog}(\hat{\Lambda}n/\epsilon))$ one- and two-qubit gates.

3.2.2. Multihead-Attention on Quantum computer

In the multihead attention module, We have H set of queries, values, and keys as [9]:

$$\begin{aligned} \mathbb{R}^{p \times n} &\ni \mathbf{Q}_h = \mathbf{W}_{Q,h}^\top \mathbf{X}, \quad \forall h \in \{1, \dots, H\}, \\ \mathbb{R}^{p \times n} &\ni \mathbf{V}_h = \mathbf{W}_{V,h}^\top \mathbf{X}, \quad \forall h \in \{1, \dots, H\}, \\ \mathbb{R}^{r \times n} &\ni \mathbf{K}_h = \mathbf{W}_{K,h}^\top \mathbf{X}, \quad \forall h \in \{1, \dots, H\}. \end{aligned}$$

Then, the scaled dot product attention is applied to generate the H output $\{\mathbf{Z}_h\}_{h=1}^H$, in accordance with Eqn.12:

$$\mathbf{Z}_h = \mathbf{W}_{V,h}^\top \mathbf{X} \mathbf{A}_h, \quad (22)$$

where $\mathbf{A}_h \equiv \mathbf{Q}_h^\top \mathbf{K}_h$, and $\mathbf{Z}_h = [\mathbf{z}_{1,h}, \dots, \mathbf{z}_{n,h}] \in \mathbb{R}^{r \times n}$.

The outputs are concatenated over different heads as

$$\mathbf{Z}_{\text{Multi-heads}} = \left[\|_{h=1}^H \mathbf{z}_{1,h}, \|_{h=1}^H \mathbf{z}_{2,h}, \dots, \|_{h=1}^H \mathbf{z}_{n,h} \right] \in \mathbb{R}^{rH \times n} \quad (23)$$

where \parallel represents concatenation [21]. Then, by a linear projection \mathbf{W}_O^\top , the total attention value is obtained:

$$\mathbf{z}_i^{\text{Total}} := \mathbf{W}_O^\top \|_{h=1}^H \mathbf{z}_{i,h} \quad (24)$$

$$\mathbf{Z}^{\text{Total}} := \mathbf{W}_O^\top \mathbf{Z}_{\text{Multi-heads}}$$

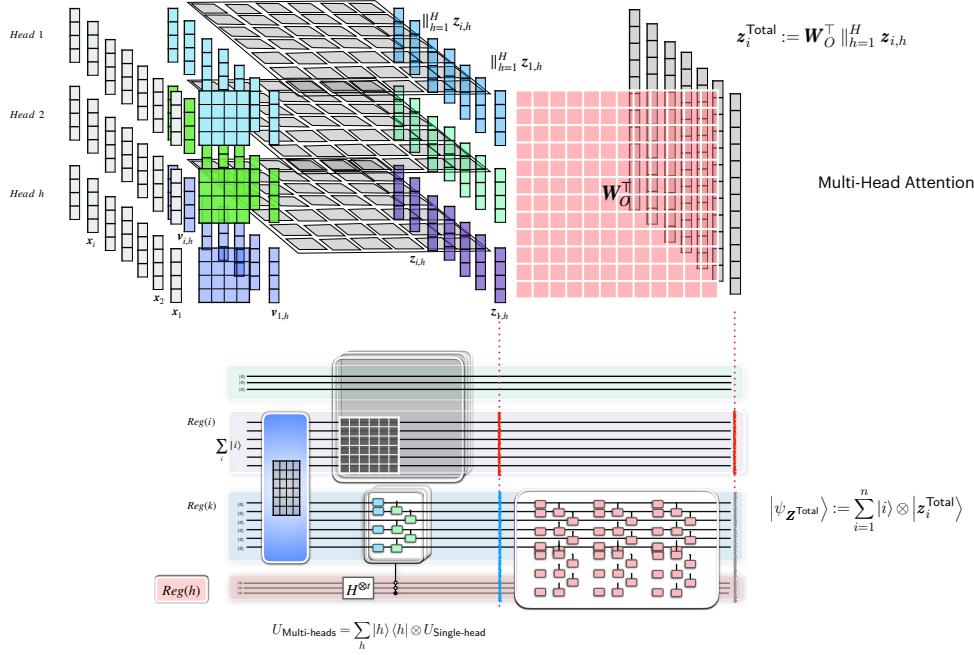


FIG. 7. *Multihead-Attention on Quantum Computer.* The upper part of the figure provides the illustration of classical Multihead-Attention, and the lower part of the figure depicts its quantum implementation. In the multihead attention module, We have H set of queries, values, and keys as: $\mathbb{R}^{p \times n} \ni \mathbf{Q}_h = \mathbf{W}_{Q,h}^\top \mathbf{X}, \quad \forall h \in \{1, \dots, H\}$, $\mathbb{R}^{p \times n} \ni \mathbf{V}_h = \mathbf{W}_{V,h}^\top \mathbf{X}, \quad \forall h \in \{1, \dots, H\}$, $\mathbb{R}^{r \times n} \ni \mathbf{K}_h = \mathbf{W}_{K,h}^\top \mathbf{X}, \quad \forall h \in \{1, \dots, H\}$. Then, the scaled dot product attention are applied to generate the H output $\{\mathbf{Z}_h\}_{h=1}^H$ and $\mathbf{Z}_h = [z_{1,h}, \dots, z_{n,h}] \in \mathbb{R}^{r \times n}$. The outputs are concatenated over different heads as $\mathbf{Z}_{\text{Multi-heads}} = [\|_{h=1}^H z_{1,h}, \|_{h=1}^H z_{2,h}, \dots, \|_{h=1}^H z_{n,h},] \in \mathbb{R}^{rH \times n}$, where $\|$ represents concatenation [21]. Then, by a linear projection \mathbf{W}_O^\top , the total attention value is obtained: $\mathbf{z}_i^{\text{Total}} := \mathbf{W}_O^\top \|_{h=1}^H z_{i,h}, \mathbf{Z}^{\text{Total}} := \mathbf{W}_O^\top \mathbf{Z}_{\text{Multi-heads}}$ and $\mathbf{Z}^{\text{Total}} = [z_1^{\text{Total}}, \dots, z_n^{\text{Total}}] \in \mathbb{R}^{rH \times n}$. On the quantum circuit, the input encoding is represented by the blue box, as described in Section 3.1. The multi-head attention can be implemented by applying a multi-controlled unitary defined as $U_{\text{Multi-heads}} = \sum_h |h\rangle\langle h| \otimes U_{\text{Single-head}}$ where $U_{\text{Single-head}} = (U_{\mathbf{A}_h^\top} \otimes U_{\mathbf{W}_{V,h}^\top})$ representing the block-encoding of \mathbf{A}_h^\top and a parameterized quantum circuit for $\mathbf{W}_{V,h}^\top$ for each head.

and $\mathbf{Z}^{\text{Total}} = [z_1^{\text{Total}}, \dots, z_n^{\text{Total}}] \in \mathbb{R}^{rH \times n}$.

The above description of classical multihead attention can be illustrated in Fig.7. Next, we present its quantum implementation.

On a quantum circuit, we aim to obtain the following quantum state,

$$|\psi_{\mathbf{Z}^{\text{Total}}} \rangle := \sum_{i=1}^n |i\rangle \otimes |\mathbf{z}_i^{\text{Total}}\rangle \quad (25)$$

where $|\mathbf{z}_i^{\text{Total}}\rangle$ is the amplitude encoding of the vector $\mathbf{z}_i^{\text{Total}}$. This can be achieved via the quantum circuit depicted in Fig.7 in which the additional register $Reg(h)$ is hosting index h , and the multi-controlled unitary is defined as

$$U_{\text{Multi-heads}} = \sum_h |h\rangle \langle h| \otimes U_{\text{Single-head}} \quad (26)$$

with $U_{\text{Single-head}} = (U_{A_h^\top} \otimes U_{W_{V,h}^\top})$ representing the block-encoding of A_h^\top and a parameterized quantum circuit for $W_{V,h}^\top$ for each head.

For a single head, from Eq. 19, when $U_{\text{Single-head}}$ acts on the state $|\psi_X\rangle \otimes |0\rangle$, the outcome state is

$$U_{\text{Single-head}} (|\psi_X\rangle \otimes |0\rangle) = |\psi_{Z_h}\rangle \otimes |0\rangle + \dots \quad (27)$$

where $|\psi_{Z_h}\rangle := \sum_{i=1}^n |i\rangle \otimes |\mathbf{z}_{i,h}\rangle$ and $|\mathbf{z}_{i,h}\rangle$ is the amplitude encoding of the vector $\mathbf{z}_{i,h}$.

When $U_{\text{Multi-heads}}$ acts on the state prepared as $\sum_h |h\rangle \otimes (|\psi_X\rangle \otimes |0\rangle)$ by the blue box and Hadamard gates on $Reg(h)$, the outcome state is

$$|\psi_{\text{Multi-heads}}\rangle = U_{\text{Multi-heads}} \left(\sum_h |h\rangle \otimes (|\psi_X\rangle \otimes |0\rangle) \right) = \sum_h |h\rangle \otimes (|\psi_{Z_h}\rangle \otimes |0\rangle + \dots) \quad (28)$$

upon post-selecting, we obtain the state

$$|\psi_{\text{Multi-heads}}\rangle = \sum_{i=1}^n |i\rangle \otimes \sum_h |h\rangle |\mathbf{z}_{i,h}\rangle \quad (29)$$

$$= \sum_{i=1}^n |i\rangle \otimes \left| \left\|_{h=1}^H \mathbf{z}_{i,h} \right\| \right\rangle \quad (30)$$

where $\left| \left\|_{h=1}^H \mathbf{z}_{i,h} \right\| \right\rangle$ is the amplitude encoding of the vector $\left\|_{h=1}^H \mathbf{z}_{i,h} \right\|$.

Applying a parameterized quantum circuit implementing W_O^\top on $Reg(k)$ and $Reg(h)$, which act as $U_{W_O^\top} \left| \left\|_{h=1}^H \mathbf{z}_{i,h} \right\| \right\rangle = \left| \mathbf{z}_i^{\text{Total}} \right\rangle$, obtain

$$U_{W_O^\top} |\psi_{\text{Multi-heads}}\rangle = \sum_{i=1}^n |i\rangle \otimes U_{W_O^\top} \left| \left\|_{h=1}^H \mathbf{z}_{i,h} \right\| \right\rangle \quad (31)$$

$$= \sum_{i=1}^n |i\rangle \otimes \left| \mathbf{z}_i^{\text{Total}} \right\rangle \quad (32)$$

$$= |\psi_Z^{\text{Total}}\rangle \quad (33)$$

which is the desired state in Eqn.25.

3.3. Residual-connection on Quantum computer

After the multihead attention module, the input to the transformer block \mathbf{x}_i and the total attention value $\mathbf{z}_i^{\text{Total}}$ are added (often referred as "residual-connection" introduced by ResNet [22]):

$$\mathbf{z}'_i := \mathbf{z}_i^{\text{Total}} + \text{concat}(\mathbf{x}_i, \mathbf{x}_i, \dots, \mathbf{x}_i) \in \mathbb{R}^{rH} \quad (34)$$

where $\text{concat}(\mathbf{x}_i, \mathbf{x}_i, \dots, \mathbf{x}_i)$ represents the concatenation of H identical vectors¹⁰ \mathbf{x}_i . For later use, define $|\text{concat}(\mathbf{x}_i, \mathbf{x}_i, \dots, \mathbf{x}_i)\rangle$ as the amplitude encoding of the vector $\text{concat}(\mathbf{x}_i, \mathbf{x}_i, \dots, \mathbf{x}_i)$ and $\mathbf{Z}' := [\mathbf{z}'_1, \dots, \mathbf{z}'_n] \in \mathbb{R}^{rH \times n}$.

On the quantum circuit, we aim to obtain the following quantum state

$$|\psi_{\mathbf{Z}'}\rangle := \sum_{i=1}^n |i\rangle \otimes |\mathbf{z}'_i\rangle \quad (35)$$

where $|\mathbf{z}'_i\rangle$ is the amplitude encoding of the vector \mathbf{z}'_i . This can be achieved via the quantum circuit depicted in Fig.8.

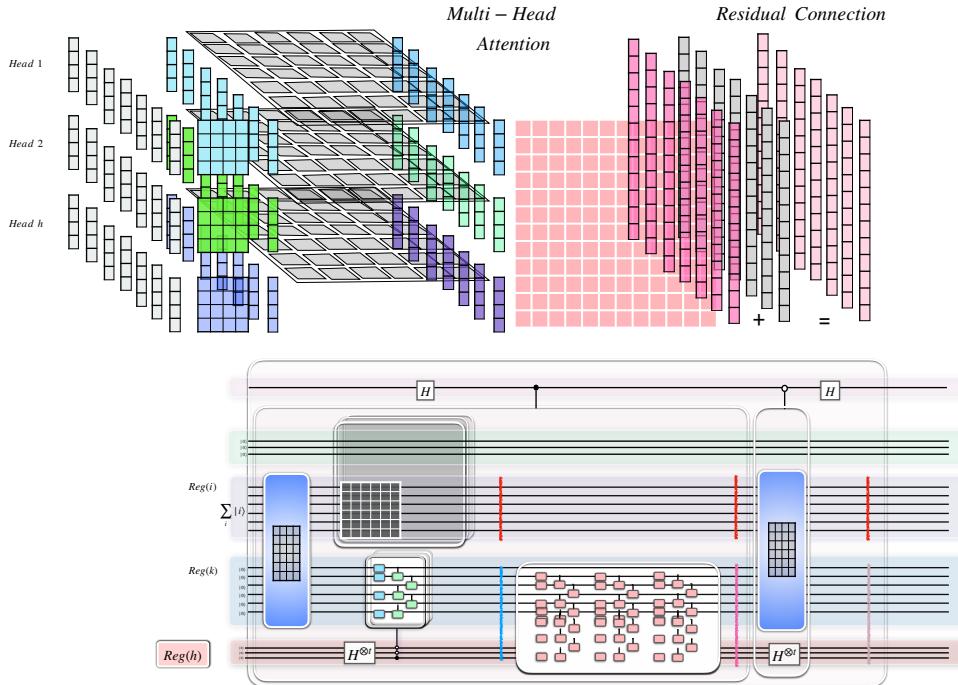


FIG. 8. *Residual-connection on Quantum computer* The upper part of the figure provides the illustration of classical Multihead-Attention followed by Residual-connection, the lower part of the figure depicts their quantum implementation. After the multihead attention module, the data (containing positional encoding) \mathbf{x}_i and the total attention value $\mathbf{z}_i^{\text{Total}}$ are added (often referred as "residual-connection" introduced by ResNet [22]): $\mathbf{z}'_i := \mathbf{z}_i^{\text{Total}} + \text{concat}(\mathbf{x}_i, \mathbf{x}_i, \dots, \mathbf{x}_i) \in \mathbb{R}^{rH}$. The quantum circuit in this figure generates the following quantum state $|\psi_{\mathbf{Z}'}\rangle := \sum_{i=1}^n |i\rangle \otimes |\mathbf{z}'_i\rangle$ where $|\mathbf{z}'_i\rangle$ is the amplitude encoding of the vector \mathbf{z}'_i . The circuit implements Linear Combination of Unitaries of two operators grouped in the two transparent boxes controlled by the top ancillary qubit.

The first transparent box controlled by the top ancillary qubit in Fig.8 implements $U_{\mathbf{Z}^{\text{Total}}}$ which acts as

¹⁰ Note that this is a bit different from the standard classical residual connection.

$$U_{Z^{\text{Total}}} : |i\rangle \otimes |0\rangle \rightarrow |i\rangle \otimes |\mathbf{z}_i^{\text{Total}}\rangle \quad (36)$$

The blue box represents the data encoding (containing positional encoding) which acts as

$$U_X : |i\rangle \otimes |0\rangle \rightarrow |i\rangle \otimes |\mathbf{x}_i\rangle \quad (37)$$

Including the Hadamard gates, the second transparent box controlled by the top ancillary qubit acts on the input state as

$$U_X \otimes H^{\otimes t} \left(\sum_{i=1}^n |i\rangle \otimes |0\rangle \otimes |0\rangle \right) = \sum_{i=1}^n |i\rangle \otimes |\mathbf{x}_i\rangle \otimes \sum_h |h\rangle = |\psi_X\rangle \otimes \sum_h |h\rangle \quad (38)$$

where $t = \log_2(H)$, and the H here is the number of "heads" in the multi-head attention defined in Section 3.2.2.

The circuit in Fig.8 implements a linear combination of two unitaries as in the two transparent boxes controlled by the top ancillary qubit; it generates the state $(|\psi_{Z^{\text{Total}}}\rangle + |\psi_X\rangle \otimes \sum_h |h\rangle) |0\rangle + \dots$ in which $(|\psi_{Z^{\text{Total}}}\rangle + |\psi_X\rangle \otimes \sum_h |h\rangle)$ can be rewritten as follows:

$$\begin{aligned} |\psi_{Z^{\text{Total}}}\rangle + |\psi_X\rangle \otimes \sum_h |h\rangle &= \sum_{i=1}^n |i\rangle \otimes |\mathbf{z}_i^{\text{Total}}\rangle + \sum_{i=1}^n |i\rangle \otimes |\mathbf{x}_i\rangle \otimes \sum_h |h\rangle \\ &= \sum_{i=1}^n |i\rangle \otimes \left(|\mathbf{z}_i^{\text{Total}}\rangle + |\mathbf{x}_i\rangle \otimes \sum_h |h\rangle \right) \\ &= \sum_{i=1}^n |i\rangle \otimes \left(|\mathbf{z}_i^{\text{Total}}\rangle + \sum_h |h\rangle |\mathbf{x}_i\rangle \right) \\ &= \sum_{i=1}^n |i\rangle \otimes \left(|\mathbf{z}_i^{\text{Total}}\rangle + |\text{concat}(\mathbf{x}_i, \mathbf{x}_i, \dots, \mathbf{x}_i)\rangle \right) \\ &= \sum_{i=1}^n |i\rangle \otimes |\mathbf{z}'_i\rangle = |\psi_{Z'}\rangle. \end{aligned} \quad (39)$$

which is the desired state in Eqn.35.

In our quantum implementation of GPT, the layer normalization procedure is omitted, our quantum approach leverages the inherent unitarity of quantum state evolutions to achieve normalization.

3.4. Feed-Forward Network on Quantum computer

After the multihead attention module and residual connection, a position-wise Feed-Forward Network(FFN) is applied[9]. The FFN is a fully connected feed-forward module that operates separately and identically on each \mathbf{z}'_i :

$$\text{FFN}(\mathbf{z}'_i) = \mathbf{W}^{2\top} \text{ReLU} \left(\mathbf{W}^{1\top} \mathbf{z}'_i + \mathbf{b}^1 \right) + \mathbf{b}^2,$$

where $\mathbf{W}^1 \in \mathbb{R}^{rH \times d_{ff}}$, $\mathbf{W}^2 \in \mathbb{R}^{d_{ff} \times rH}$, $\mathbf{b}^1 \in \mathbb{R}^{d_{ff}}$, $\mathbf{b}^2 \in \mathbb{R}^{rH}$ are trainable parameters, d_{ff} is the intermediate dimension of the FFN. For simplicity, we omit $\mathbf{b}^1, \mathbf{b}^2$ in the following discussion. Similar to how we defined $\mathbf{X} := [\mathbf{x}_1, \dots, \mathbf{x}_n]$ before, we can write $\mathbf{W}^1 = [\mathbf{w}_1, \dots, \mathbf{w}_m, \dots, \mathbf{w}_{d_{ff}}]$ where $\mathbf{w}_m \in \mathbb{R}^{rH}$.

Denote $\mathbf{y}_i := \mathbf{W}^{1\top} \mathbf{z}'_i \in \mathbb{R}^{d_{ff}}$, we have its elements as

$$\mathbf{y}_i^{(m)} = \mathbf{z}'_i \cdot \mathbf{w}_m, \forall m \in \{1, \dots, d_{ff}\} \quad (40)$$

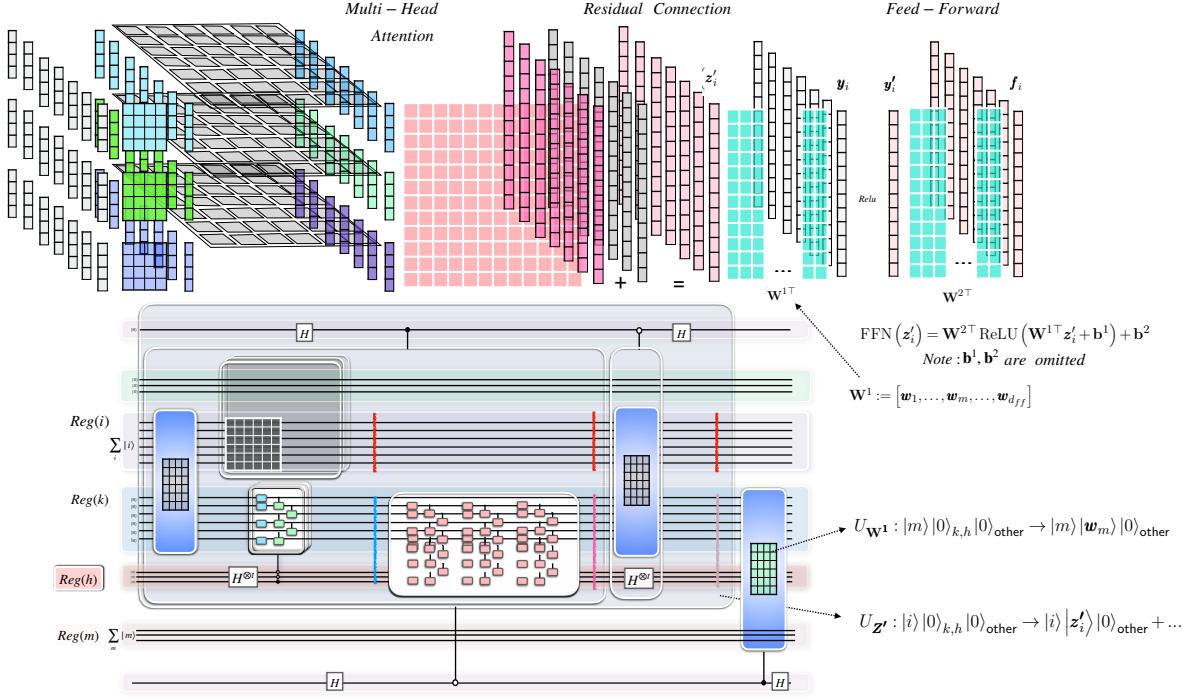


FIG. 9. *Feed-Forward Network on Quantum computer 1* The upper part of the figure provides the illustration of classical Multihead-Attention followed by residual connection and Feed-Forward Network, and the lower part of the figure depicts their quantum implementation. After the multihead attention module and residual connection, a position-wise Feed-Forward Network(FFN) is applied. The FFN is a fully connected feed-forward module that operates separately and identically on each \mathbf{z}'_i : $\mathbf{W}^{2\top} \text{ReLU}(\mathbf{W}^{1\top} \mathbf{z}'_i + \mathbf{b}^1) + \mathbf{b}^2$, we can write $\mathbf{W}^1 := [\mathbf{w}_1, \dots, \mathbf{w}_m, \dots, \mathbf{w}_{d_{ff}}]$ where $\mathbf{w}_m \in \mathbb{R}^{rH}$, d_{ff} is the intermediate dimension d_{ff} of the FFN. Denote $\mathbf{y}_i := \mathbf{W}^{1\top} \mathbf{z}'_i$, we have its elements as $\mathbf{y}_i^{(m)} = \mathbf{z}'_i \cdot \mathbf{w}_m$. Recall we created state on registered $\text{Reg}(i)$, $\text{Reg}(k)$, $\text{Reg}(h)$ and ancillas: $|\psi_{\mathbf{Z}'}\rangle = \sum_{i=1}^n |i\rangle \otimes |\mathbf{z}'_i\rangle \otimes |0\rangle + \dots$, by the unitary circled in the overall transparent box in this figure, denoted as $U_{\mathbf{Z}'}$, which act as $U_{\mathbf{Z}'} : |i\rangle |0\rangle_{k,h} |0\rangle_{\text{other}} \rightarrow |i\rangle |\mathbf{z}'_i\rangle |0\rangle_{\text{other}} + \dots, \forall i \in \{1, \dots, n\}$. For implementing \mathbf{W}^1 , we can create a trainable unitary $U_{\mathbf{W}^1} : |m\rangle |0\rangle_{k,h} |0\rangle_{\text{other}} \rightarrow |m\rangle |\mathbf{w}_m\rangle |0\rangle_{\text{other}}, \forall m \in \{1, \dots, d_{ff}\}$, with $|\mathbf{w}_m\rangle$ on $\text{Reg}(k)$, $\text{Reg}(h)$ and $|m\rangle$ on an additional registered $\text{Reg}(m)$. $U_{\mathbf{W}^1}$, depicted as the blue box with a green centre in this figure, can be implemented as a series of controlled parameterised quantum circuits as $U_{\mathbf{W}^1} = \sum_m |m\rangle \langle m| U_m$ where each U_m , acting as $U_m : |0\rangle_{k,h} \rightarrow |\mathbf{w}_m\rangle$, is a parameterised quantum circuit. $\mathbf{y}_i^{(m)} = \langle \mathbf{z}'_i | \mathbf{w}_m \rangle$ can be evaluated using Parallel Swap test for each \mathbf{z}'_i and \mathbf{w}_m , via the quantum circuit depicted in this figure.

Next, we present a quantum implementation of the FFN module similar to the approach proposed in Ref. [23].

Recall the unitary circled in the overall transparent box in Fig.8, denoted as $U_{\mathbf{Z}'}$, act as

$$U_{\mathbf{Z}'} : |i\rangle |0\rangle_{k,h} |0\rangle_{\text{other}} \rightarrow |i\rangle |\mathbf{z}'_i\rangle |0\rangle_{\text{other}} + \dots, \forall i \in \{1, \dots, n\}. \quad (41)$$

where $|0\rangle_{k,h}$ represents the "all-zero" state ¹¹ of the qubits in $\text{Reg}(k)$ and $\text{Reg}(h)$, $|0\rangle_{\text{other}}$ represents the "all-zero" state of the ancillary qubits(qubits that are not included in $\text{Reg}(i), \text{Reg}(k), \text{Reg}(h)$ in Fig.8).

For implementing $\mathbf{W}^1 := [\mathbf{w}_1, \dots, \mathbf{w}_m, \dots, \mathbf{w}_{d_{ff}}]$, we can create a trainable unitary

$$U_{\mathbf{W}^1} : |m\rangle |0\rangle_{k,h} \rightarrow |m\rangle |\mathbf{w}_m\rangle, \forall m \in \{1, \dots, d_{ff}\}. \quad (42)$$

with $|\mathbf{w}_m\rangle$ on $\text{Reg}(k), \text{Reg}(h)$ and $|m\rangle$ on an additional registered $\text{Reg}(m)$.

Notice that

$$\mathbf{y}_i^{(m)} = \langle \mathbf{z}'_i | \mathbf{w}_m \rangle, \forall m \in \{1, \dots, d_{ff}\} \quad (43)$$

This can be evaluated using parallel swap test for each $\mathbf{z}'_i \in \mathbb{R}^{rH}$ and $\mathbf{w}_m \in \mathbb{R}^{rH}$, via the quantum circuit depicted in Fig.9.

The input state to the circuit is

$$|\Psi_0\rangle = \sum_i \sum_m |i\rangle |m\rangle |0\rangle_{k,h} |0\rangle_{\text{other}} |0\rangle \quad (44)$$

For each branch $|i\rangle |m\rangle |0\rangle_{k,h} |0\rangle_{\text{other}} |0\rangle$, applying a Hadamard gate on the bottom ancillary qubit, and controlled $U_{\mathbf{Z}'}, U_{\mathbf{W}^1}$ we obtain

$$|i\rangle |m\rangle (|\mathbf{z}'_i\rangle |0\rangle_{\text{other}} + \dots) |0\rangle + |i\rangle |m\rangle |\mathbf{w}_m\rangle |0\rangle_{\text{other}} |1\rangle \quad (45)$$

Applying another Hadamard gate on the bottom ancillary qubit yield

$$|\psi_{im}\rangle = |i\rangle |m\rangle ((|\mathbf{z}'_i\rangle |0\rangle_{\text{other}} + \dots) + |\mathbf{w}_m\rangle |0\rangle_{\text{other}}) |0\rangle + |i\rangle |m\rangle ((|\mathbf{z}'_i\rangle |0\rangle_{\text{other}} + \dots) - |\mathbf{w}_m\rangle |0\rangle_{\text{other}}) |1\rangle \quad (46)$$

Denote $|u_{im}\rangle$ and $|v_{im}\rangle$ as the normalized states of $((|\mathbf{z}'_i\rangle |0\rangle_{\text{other}} + \dots) + |\mathbf{w}_m\rangle |0\rangle_{\text{other}})$ and $((|\mathbf{z}'_i\rangle |0\rangle_{\text{other}} + \dots) - |\mathbf{w}_m\rangle |0\rangle_{\text{other}})$ respectively.

Then there is a real number θ_{im} such that

$$|\psi_{im}\rangle = |i\rangle |m\rangle (\underbrace{\sin \theta_{im} |u_{im}\rangle |0\rangle + \cos \theta_{im} |v_{im}\rangle |1\rangle}_{|\phi_{im}\rangle}) = |i\rangle |m\rangle |\phi_{im}\rangle \quad (47)$$

θ_{im} satisfies $\cos \theta_{im} = \sqrt{1 - \langle \mathbf{z}'_i | \mathbf{w}_m \rangle} / \sqrt{2}$, $\sin \theta_{im} = \sqrt{1 + \langle \mathbf{z}'_i | \mathbf{w}_m \rangle} / \sqrt{2}$, and we have:

$$\langle \mathbf{z}'_i | \mathbf{w}_m \rangle = -\cos 2\theta_{im}. \quad (48)$$

¹¹ in this paper, the 'all-zero' state refers to all the relevant qubits being in the $|0\rangle$ state.

To summarize, the quantum circuit depicted in Fig.9, denoted as U , acts as

$$U : |i\rangle|m\rangle|0\rangle_{k,h}|0\rangle_{\text{other}}|0\rangle \rightarrow |i\rangle|m\rangle(\underbrace{\sin \theta_{im}|u_{im}\rangle|0\rangle + \cos \theta_{im}|v_{im}\rangle|1\rangle}_{|\phi_{im}\rangle}) = |i\rangle|m\rangle|\phi_{im}\rangle, \quad (49)$$

where $\mathbf{y}_i^{(m)} = \langle \mathbf{z}'_i | \mathbf{w}_m \rangle$ are encoded as

$$\langle \mathbf{z}'_i | \mathbf{w}_m \rangle = -\cos 2\theta_{im}. \quad (50)$$

When acting on the input state to the circuit $|\Psi_0\rangle = \sum_i \sum_m |i\rangle|m\rangle|0\rangle_{k,h}|0\rangle_{\text{other}}|0\rangle$, U , also depicted as the pink box in Fig.10, produces the following state

$$|\Psi_1\rangle = \sum_i \sum_m |i\rangle|m\rangle(\underbrace{\sin \theta_{im}|u_{im}\rangle|0\rangle + \cos \theta_{im}|v_{im}\rangle|1\rangle}_{|\phi_{im}\rangle}) = \sum_i \sum_m |i\rangle|m\rangle|\phi_{im}\rangle \quad (51)$$

Next use amplitude estimation [24] to extract and store $\mathbf{y}_i^{(m)} = \langle \mathbf{z}'_i | \mathbf{w}_m \rangle$ into an additional register which we call the “amplitude register” $|0\rangle_{\text{amplitude}}^t$ and the output state $|\Psi_1\rangle$ (using the same notation) becomes

$$|\Psi_3\rangle = \sum_i \sum_m |i\rangle|m\rangle|\phi_{im}\rangle|0\rangle_{\text{amplitude}}^t, \quad (52)$$

where $|\phi_{im}\rangle$ can be decomposed as

$$|\phi_{im}\rangle = \frac{-i}{\sqrt{2}} \left(e^{i\theta_{im}} |\omega_+\rangle_{im} - e^{i(-\theta_{im})} |\omega_-\rangle_{im} \right). \quad (53)$$

where $|w_{\pm}\rangle_{im} = \frac{1}{\sqrt{2}}(|0\rangle|u_{im}\rangle \pm |1\rangle|v_{im}\rangle)$.

Hence, we have

$$|\Psi_1\rangle = \sum_i \sum_m \frac{-i}{\sqrt{2}} \left(e^{i\theta_{im}} |i\rangle|m\rangle|\omega_+\rangle_{im} - e^{i(-\theta_{im})} |i\rangle|m\rangle|\omega_-\rangle_{im} \right) |0\rangle_{\text{amplitude}}^t. \quad (54)$$

The overall Grover operator G is defined as

$$G := UC_2U^{-1}C_1, \quad (55)$$

where C_1 is the Z gate on the bottom ancilla qubit in the pink box, and $C_2 = (I - 2|0\rangle\langle 0|) \otimes I_{i,m}$ is the “flip zero state” on registers other than $Reg(i), Reg(m)$ (represented as S_0 in Fig.10).

Utilising Eqn. 49, it can be shown that G can be expressed as

$$G = \sum_i \sum_m |i\rangle|m\rangle\langle m|\langle i| \otimes G_{im}, \quad (56)$$

where G_{im} is defined as

$$G_{im} = (I - 2|\phi_{im}\rangle\langle\phi_{im}|)(Z \otimes I) \quad (57)$$

It is easy to check that $|w_{\pm}\rangle_{im}$ are the eigenstates of G_{im} , that is,

$$G_{im}|w_{\pm}\rangle_{im} = e^{\pm i2\theta_{im}}|w_{\pm}\rangle_{im}. \quad (58)$$

Therefore, the overall Grover operator G possesses the following eigen-relation:

$$G|i\rangle|m\rangle|\omega_{\pm}\rangle_{im} = e^{i(\pm 2\theta_{im})}|i\rangle|m\rangle|\omega_{\pm}\rangle_{im}. \quad (59)$$

Next, we apply phase estimation of the overall Grover operator G on the input state $|\Psi_1\rangle$. The resulting state $|\Psi_2\rangle$ can be written as

$$|\Psi_2\rangle = \sum_i \sum_m \frac{-i}{\sqrt{2}} \left(e^{i\theta_{im}} |i\rangle|m\rangle|\omega_+\rangle_{im}|2\theta_{im}\rangle - e^{i(-\theta_{im})} |i\rangle|m\rangle|\omega_-\rangle_{im}|-2\theta_{im}\rangle \right). \quad (60)$$

Note here in Eq. 60, $|\pm 2\theta_{im}\rangle$ denotes the eigenvalues $\pm 2\theta_{im}$ being stored in the amplitude register with some finite precision.

Then we apply an oracle U_O (implemented by arithmetic circuit) on the amplitude register and an extra ancilla register $|0\rangle_{ReLU}$, which acts as

$$U_O|0\rangle_{ReLU}|\pm 2\theta_{im}\rangle = |\text{ReLU}(\mathbf{y}_i^{(m)})\rangle|\pm 2\theta_{im}\rangle, \quad (61)$$

The state after the oracle can be written as

$$|\Psi_3\rangle = \sum_i \sum_m \frac{-i}{\sqrt{2}} |\text{ReLU}(\mathbf{y}_i^{(m)})\rangle \left(e^{i\theta_{im}} |i\rangle|m\rangle|\omega_+\rangle_{im}|2\theta_{im}\rangle - e^{i(-\theta_{im})} |i\rangle|m\rangle|\omega_-\rangle_{im}|-2\theta_{im}\rangle \right). \quad (62)$$

Then we perform the uncomputation of Phase estimation, resulting in the state,

$$|\Psi_4\rangle = \sum_i \sum_m \frac{-i}{\sqrt{2}} |\text{ReLU}(\mathbf{y}_i^{(m)})\rangle \left(e^{i\theta_{im}} |i\rangle|m\rangle|\omega_+\rangle_{im}|0\rangle_{\text{amplitude}}^t - e^{i(-\theta_{im})} |i\rangle|m\rangle|\omega_-\rangle_{im}|0\rangle_{\text{amplitude}}^t \right) \quad (63)$$

$$= \sum_i \sum_m |\text{ReLU}(\mathbf{y}_i^{(m)})\rangle |i\rangle|m\rangle|\phi_{im}\rangle|0\rangle_{\text{amplitude}}^t \quad (64)$$

Finally, we perform U^\dagger and the resulting state is

$$|\Psi_5\rangle = \sum_i \sum_m |\text{ReLU}(\mathbf{y}_i^{(m)})\rangle |i\rangle|m\rangle|0\rangle_{k,h}|0\rangle_{\text{other}}|0\rangle|0\rangle_{\text{amplitude}}^t. \quad (65)$$

The above steps, as gathered in the grey box in Fig. 10, implement an oracle O_{FFN} such that:

$$O_{FFN}: |i\rangle|m\rangle|0\rangle_{k,h}|0\rangle_{\text{other}}|0\rangle|0\rangle_{ReLU} \rightarrow |i\rangle|m\rangle|0\rangle_{k,h}|0\rangle_{\text{other}}|0\rangle|\text{ReLU}(\mathbf{y}_i^{(m)})\rangle \quad (66)$$

Which produces the state

$$\sum_i \sum_m |i\rangle|m\rangle|\mathbf{y}'_i^{(m)}\rangle|0\rangle_{k,h}|0\rangle_{\text{other}}|0\rangle \quad (67)$$

where we denote $\mathbf{y}'_i^{(m)} = \text{ReLU}(\mathbf{y}_i^{(m)})$.

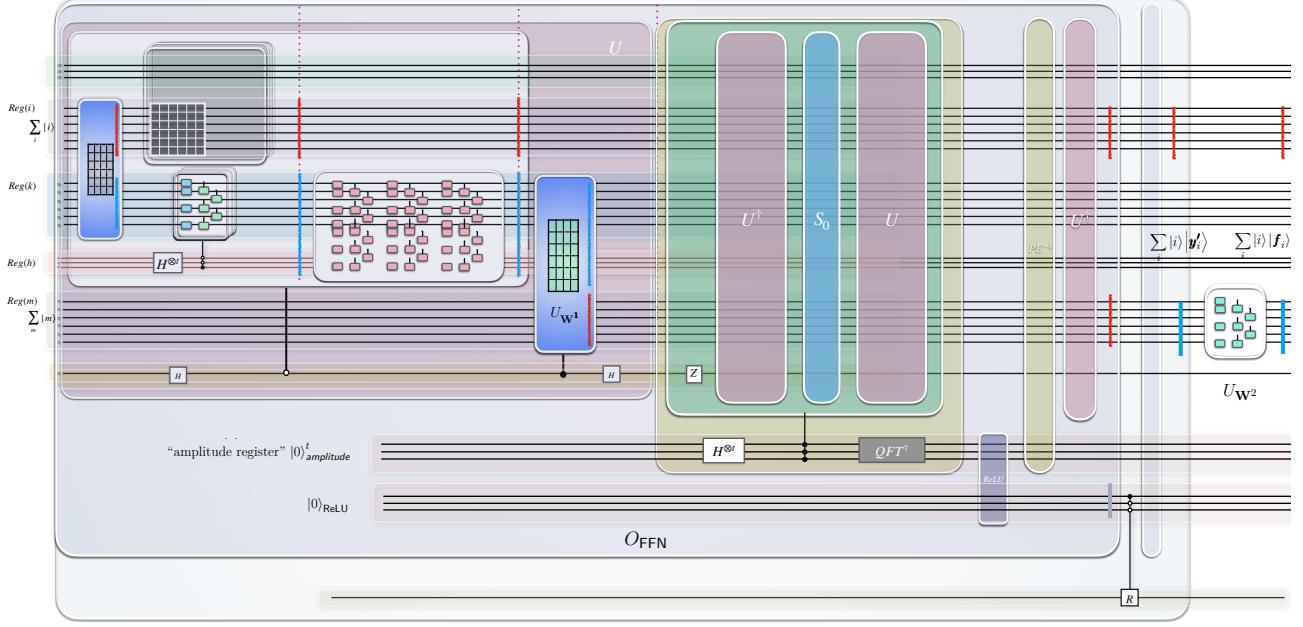


FIG. 10. *Feed-Forward Network on Quantum computer 2* The figure illustrates the description from Eqn.49 to 69. The pink box in this figure, denoted as U , is meant to be the circuit in Fig.9, but for simplicity, we omitted the residual connection as in Fig.9. However, the derivation follows the same logic.

Next, the "Conditional Rotation" (Theorem 3.5 in Ref. [25]) and uncomputation of O_{FFN} are applied, obtaining

$$\sum_i \sum_m |i\rangle \mathbf{y}'_i^{(m)} |m\rangle = \sum_i |i\rangle \sum_m \mathbf{y}'_i^{(m)} |m\rangle = \sum_i |i\rangle |\mathbf{y}'_i\rangle \quad (68)$$

where $|\mathbf{y}'_i\rangle := \sum_m \mathbf{y}'_i^{(m)} |m\rangle$ and we omitted the registers being in "all-zero" state.

Finally, a trainable unitary $U_{\mathbf{W}^2}$ (a parameterised quantum circuit) implementing $\mathbf{W}^{2\top}$ is applied

$$|\Psi_F\rangle = \sum_i |i\rangle U_{\mathbf{W}^2} |\mathbf{y}'_i\rangle = \sum_i |i\rangle |\mathbf{f}_i\rangle \quad (69)$$

where $\mathbf{f}_i := \mathbf{W}^{2\top} \mathbf{y}'_i = \text{FFN}(\mathbf{z}'_i) \in \mathbb{R}^{rH}$ and we define $\mathbf{F} := [\mathbf{f}_1, \dots, \mathbf{f}_n] \in \mathbb{R}^{rH \times n}$.

By the end of the quantum circuit in Fig. 10, we have obtained the final state from a Transformer block.

A tomography procedure (Theorem 4.3 from Ref.[25]) is performed to read out the amplitudes of the final state of a Transformer block. The results are then used as the input for the next Transformer block.

3.5. Generative Pre-training on Quantum Computer

Using the same notation as in the previous section, we denote the final state from the last Transformer block in GPT as:

$$|\Psi_{\mathbf{F}}\rangle = \sum_i |i\rangle |\mathbf{f}_i\rangle \quad (70)$$

where $\mathbf{f}_i \in \mathbb{R}^{rH}$ and $\mathbf{F} := [\mathbf{f}_1, \dots, \mathbf{f}_n] \in \mathbb{R}^{rH \times n}$.

For language modeling, we then apply a linear layer $\mathbf{W}_E \in \mathbb{R}^{V \times rH}$ to map the output back to the vocabulary space as $\mathbf{f}'_i = \mathbf{W}_E \mathbf{f}_i \in \mathbb{R}^V$. \mathbf{W}_E can be implemented on a quantum circuit by applying a trainable unitary $U_{\mathbf{W}_E}$ (a parameterised quantum circuit) on $Reg(k), Reg(m)$ and some extra qubits (since $V \gg rH$) and we obtain the state

$$|\Psi_{\mathbf{F}'}\rangle = U_{\mathbf{W}_E}(|\Psi_{\mathbf{F}}\rangle \otimes |0\rangle) = \sum_i |i\rangle U_{\mathbf{W}_E}(|\mathbf{f}_i\rangle \otimes |0\rangle) = \sum_i |i\rangle |\mathbf{f}'_i\rangle \quad (71)$$

where $\mathbf{F}' := [\mathbf{f}'_1, \dots, \mathbf{f}'_n] \in \mathbb{R}^{V \times n}$.

For the t th batch in Generative Pre-training, we examine \mathbf{f}'_{t+1} in the output, given the input $[\mathbf{x}_1, \dots, \mathbf{x}_t]$. The loss function for this batch is defined as the cross-entropy between softmax(\mathbf{f}'_{t+1}) and the one-hot encoding of the $(t + 1)$ th token in the training text, denoted by $\mathbf{b}_{t+1} \in \mathbb{B}^V$. On the quantum circuit, the loss function can be correspondingly defined as the overlap between $|\mathbf{f}'_{t+1}\rangle$ and $|\mathbf{b}_{t+1}\rangle$ ¹², which can be evaluated via swap test on $|\Psi_{\mathbf{F}'}\rangle = \sum_i |i\rangle |\mathbf{f}'_i\rangle$ and an additional state $|\Psi_{\mathbf{b}_{t+1}}\rangle = |t+1\rangle |\mathbf{b}_{t+1}\rangle$. The cumulative loss is calculated across all batches, followed by the execution of certain optimization methods to adjust the model's parameters.

4. Conclusion

In this paper, we have presented a comprehensive framework for translating key components of the GPT architecture—namely, the masked multi-head attention module, the feed-forward networks module, and the residual connection—along with the generative pre-training phase, into the quantum computing paradigm. This translation demonstrates the feasibility of implementing complex language models on quantum computers, marking an initial step toward harnessing the power of quantum computing to enhance the capabilities of LLMs. As we progress, it becomes imperative to conduct a detailed analysis of the quantum resources required for our quantum implementation of GPT, such as the number of qubits and circuit depths, and to perform a thorough comparison with classical GPT implementations.

In summary, the integration of quantum computing with LLMs, as demonstrated by our work, opens new avenues for research and development in the field of QML. By continuing to explore these possibilities, we move closer to realizing the full potential of quantum-enhanced artificial intelligence,

¹² $|\mathbf{f}'_{t+1}\rangle$ and $|\mathbf{b}_{t+1}\rangle$ represent the amplitude encoding of \mathbf{f}'_{t+1} and \mathbf{b}_{t+1} .

setting the stage for future breakthroughs in computational efficiency and model capabilities.

- [1] Travis L Scholten, Carl J Williams, Dustin Moody, Michele Mosca, William Hurley, William J Zeng, Matthias Troyer, Jay M Gambetta, *et al.*, “Assessing the benefits and risks of quantum computers,” arXiv preprint arXiv:2401.16317 (2024).
- [2] Marco Cerezo, Guillaume Verdon, Hsin-Yuan Huang, Lukasz Cincio, and Patrick J Coles, “Challenges and opportunities in quantum machine learning,” *Nature Computational Science* **2**, 567–576 (2022).
- [3] Junyu Liu, Minzhao Liu, Jin-Peng Liu, Ziyu Ye, Yunfei Wang, Yuri Alexeev, Jens Eisert, and Liang Jiang, “Towards provably efficient quantum algorithms for large-scale machine-learning models,” *Nature Communications* **15**, 434 (2024).
- [4] Yeqi Gao, Zhao Song, Xin Yang, and Ruizhe Zhang, “Fast quantum algorithm for attention computation,” arXiv preprint arXiv:2307.08045 (2023).
- [5] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, *et al.*, “Improving language understanding by generative pre-training,” (2018).
- [6] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin, “Attention is all you need,” in *Advances in neural information processing systems*, Vol. 30 (2017).
- [7] Peter J Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Łukasz Kaiser, and Noam Shazeer, “Generating wikipedia by summarizing long sequences,” arXiv preprint arXiv:1801.10198 (2018).
- [8] Larry Medsker and Lakhmi C Jain, *Recurrent neural networks: design and applications* (CRC press, 1999).
- [9] Benyamin Ghojogh and Ali Ghodsi, “Attention mechanism, transformers, bert, and gpt: tutorial and survey,” OSF Preprints (2020).
- [10] Daniel Jurafsky and James H. Martin, *Speech and Language Processing*, 3rd ed. (Pearson, 2021).
- [11] Peter F Brown, Vincent J Della Pietra, Stephen A Della Pietra, and Robert L Mercer, “Class-based n-gram models of natural language,” *Computational linguistics* **18**, 467–479 (1992).
- [12] Sepp Hochreiter and Jürgen Schmidhuber, “Long short-term memory,” *Neural Computation* **9**, 1735–1780 (1997).
- [13] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeff Dean, “Efficient estimation of word representations in vector space,” arXiv preprint arXiv:1301.3781 (2013).
- [14] Jeffrey Pennington, Richard Socher, and Christopher D Manning, “Glove: Global vectors for word representation,” in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (2014) pp. 1532–1543.
- [15] Yoav Goldberg and Omer Levy, “word2vec explained: deriving mikolov et al.’s negative-sampling word-embedding method,” arXiv preprint arXiv:1402.3722 (2014).
- [16] Tomas Mikolov, Quoc V Le, and Ilya Sutskever, “Computing numeric representations of words in a high-dimensional space,” in *Machine Learning and Knowledge Discovery in Databases* (Springer, 2015) pp. 522–536.
- [17] Pei Yuan and Shengyu Zhang, “Optimal (controlled) quantum state preparation and improved unitary synthesis by quantum circuits with any number of ancillary qubits,” *Quantum* **7**, 956 (2023).
- [18] Jiachen Lu, Jinghan Yao, Junge Zhang, Xiatian Zhu, Hang Xu, Weigu Gao, Chunjing Xu, Tao Xiang, and Li Zhang, “Soft: Softmax-free transformer with linear complexity,” *Advances in Neural Information Processing Systems* **34**, 21297–21309 (2021).
- [19] Zhen Qin, Weixuan Sun, Hui Deng, Dongxu Li, Yunshen Wei, Baohong Lv, Junjie Yan, Lingpeng Kong, and Yiran Zhong, “cosformer: Rethinking softmax in attention,” arXiv preprint arXiv:2202.08791 (2022).
- [20] Quynh T Nguyen, Bobak T Kiani, and Seth Lloyd, “Block-encoding dense and full-rank kernels using hierarchical matrices: applications in quantum numerical linear algebra,” *Quantum* **6**, 876 (2022).
- [21] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio, “Graph attention networks,” in *Proc. of ICLR* (2017).
- [22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, “Deep residual learning for image recognition,” in *Proc. of CVPR* (2016) pp. 770–778.
- [23] Jonathan Allcock, Chang-Yu Hsieh, Iordanis Kerenidis, and Shengyu Zhang, “Quantum algorithms for

- feedforward neural networks,” ACM Transactions on Quantum Computing **1** (2020).
- [24] Gilles Brassard, Peter Hoyer, Michele Mosca, and Alain Tapp, “Quantum amplitude amplification and estimation,” Contemporary Mathematics **305**, 53–74 (2002).
- [25] Jonas Landman, “Quantum algorithms for unsupervised machine learning and neural networks,” arXiv preprint arXiv:2111.03598 (2021).
- [26] Yidong Liao, Min-Hsiu Hsieh, and Chris Ferrie, “Quantum optimization for training quantum neural networks,” arXiv preprint arXiv:2103.17047 (2021).
- [27] Christoph Sünderhauf, Earl Campbell, and Joan Camps, “Block-encoding structured matrices for data input in quantum computing,” Quantum **8**, 1226 (2024).
- [28] Andrew M. Childs and Nathan Wiebe, “Hamiltonian simulation using linear combinations of unitary operations,” arXiv:1202.5822 (2012).
- [29] Lin Lin, “Lecture notes on quantum algorithms for scientific computation,” arXiv preprint arXiv:2201.08309 (2022).
- [30] Martin Larocca, Nathan Ju, Diego García-Martín, Patrick J. Coles, and M. Cerezo, “Theory of over-parametrization in quantum neural networks,” Nature Computational Science **3**, 542–551 (2023).
- [31] Abhinav Kandala, Antonio Mezzacapo, Kristan Temme, Maika Takita, Markus Brink, Jerry M. Chow, and Jay M. Gambetta, “Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets,” Nature **549**, 242–246 (2017).
- [32] Stuart Hadfield, Zhihui Wang, Bryan O’Gorman, Eleanor G Rieffel, Davide Venturelli, and Rupak Biswas, “From the quantum approximate optimization algorithm to a quantum alternating operator ansatz,” Algorithms **12**, 34 (2019).
- [33] Alexandre Choquette, Agustin Di Paolo, Panagiotis Kl Barkoutsos, David Sénéchal, Ivano Tavernelli, and Alexandre Blais, “Quantum-optimal-control-inspired ansatz for variational quantum algorithms,” Physical Review Research **3**, 023092 (2021).
- [34] Ian Goodfellow, Yoshua Bengio, and Aaron Courville, *Deep learning* (MIT press, 2016).

A. Quantum Attention Oracle

In this section, we introduce the construction of the quantum attention oracle $O_{\text{attention}}$, as mentioned in Section 3.2.1. The quantum attention oracle $O_{\text{attention}}$ is designed to coherently evaluate and store the attention score $a(\mathbf{x}_i, \mathbf{x}_j)$ for each pair of input vectors \mathbf{x}_i and \mathbf{x}_j . It acts as follows:

$$O_{\text{attention}} |i\rangle |j\rangle |0\rangle \rightarrow |i\rangle |j\rangle |a(\mathbf{x}_i, \mathbf{x}_j)\rangle. \quad (\text{A1})$$

The construction of $O_{\text{attention}}$ consists of the following two steps:

1. Evaluating Attention score in superposition

The attention score $a(\mathbf{x}_i, \mathbf{x}_j)$ can take one of the standard forms in the classical literature [9] — the inner product of the linearly transformed input vectors:

$$a(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T W_K^T W_Q \mathbf{x}_j, \quad (\text{A2})$$

where W_K, W_Q are trainable linear transformations.

In terms of Dirac notation, this can be written as:

$$a(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i | U_K^\dagger U_Q | \mathbf{x}_j \rangle \quad (\text{A3})$$

in which U_K, U_Q are trainable unitaries, $|\mathbf{x}_i\rangle := \sum_{k=1}^d \mathbf{x}_i^{(k)} |k\rangle$ is the amplitude encoding of the vector \mathbf{x}_i whose k -th elements are denoted as $\mathbf{x}_i^{(k)}$. Note here and throughout the paper, we omit the

normalization factors.

This attention score can be evaluated on a quantum circuit by parallel swap test [26] as depicted in Fig. 11 which we will discuss in detail below.

$$a(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i | U_K^\dagger U_Q | \mathbf{x}_j \rangle$$

$$a(\mathbf{x}_i, \mathbf{x}_j) = (\langle \mathbf{x}_i | U_K^\dagger) \otimes ((\langle \mathbf{x}_i | U_Q^\dagger) U_e^\dagger O U_e (U_K | \mathbf{x}_i \rangle) \otimes (U_Q | \mathbf{x}_i \rangle)$$

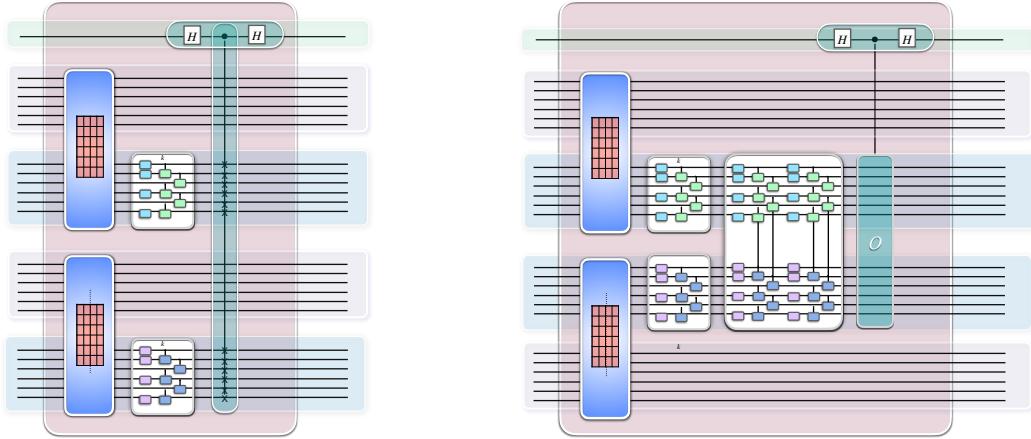


FIG. 11. *Evaluating Attention score in superposition* The attention score $a(\mathbf{x}_i, \mathbf{x}_j)$ can take one of the standard forms in the classical literature [9] — the inner product of the linearly transformed input vectors: $a(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T W_K^T W_Q \mathbf{x}_j$, in which W_K, W_Q are trainable linear transformations. This attention score can be evaluated in superposition on the quantum circuit by parallel swap test [26], depicted as the left side of this figure. On the right side of this figure, we illustrate an alternative form of the Attention score, which can be evaluated by parallel Hadamard test [26].

We denote the unitary for the parallel swap test circuit, as circled by the pink box on the left side of Fig. 11, as U . The input to U , $|\Psi_0\rangle$, can be written as (note here and throughout the paper, we omit the normalization factors):

$$|\Psi_0\rangle = |0\rangle \otimes (\sum_i |i\rangle) \otimes |0\rangle_K^n \otimes (\sum_j |j\rangle) \otimes |0\rangle_Q^n \quad (\text{A4})$$

where $|0\rangle_K^n, |0\rangle_Q^n$ are the initial states of the two registers on which the input vectors will be loaded. The input encoding via Controlled Quantum State Preparation[17](mentioned in Section 3.2.1), depicted as the blue boxes in Fig. 11, can be written as $\sum_i |i\rangle \otimes U_{\mathbf{x}_i}$ where $U_{\mathbf{x}_i}$ act as $U_{\mathbf{x}_i} |0\rangle = |\mathbf{x}_i\rangle$.

Applying this input encoding results in the following state:

$$|\Psi_1\rangle = |0\rangle \otimes (\sum_i |i\rangle \otimes |\mathbf{x}_i\rangle^n) \otimes (\sum_j |j\rangle \otimes |\mathbf{x}_j\rangle^n) \quad (\text{A5})$$

Then, U_K, U_Q implemented by parametrized quantum circuits are applied on the two registers hosting the input vectors, yielding the following state:

$$|\Psi_2\rangle = |0\rangle \otimes \left(\sum_i |i\rangle \otimes U_K |\mathbf{x}_i\rangle^n \right) \otimes \left(\sum_j |j\rangle \otimes U_Q |\mathbf{x}_j\rangle^n \right) \quad (\text{A6})$$

We further define $\mathcal{K}_i, \mathcal{Q}_j$ and corresponding state $|k_i\rangle, |q_j\rangle$ as $\mathcal{K}_i |0\rangle_{\mathbb{K}}^n = U_K |\mathbf{x}_i\rangle = |k_i\rangle, \mathcal{Q}_j |0\rangle_{\mathbb{Q}}^n = U_Q |\mathbf{x}_j\rangle = |q_j\rangle$. Then U can be written explicitly as

$$\begin{aligned} U := & [H \otimes I \otimes I \otimes I \otimes I] \cdot \\ & [|0\rangle \langle 0| \otimes \left(\sum_i \sum_j |i\rangle \langle i| \otimes \mathcal{K}_i \otimes |j\rangle \langle j| \otimes \mathcal{Q}_j \right) + |1\rangle \langle 1| \otimes \left(\sum_i \sum_j |i\rangle \langle i| \otimes \mathcal{Q}_j \otimes |j\rangle \langle j| \otimes \mathcal{K}_i \right)] \\ & \cdot [H \otimes I \otimes I \otimes I \otimes I]. \end{aligned} \quad (\text{A7})$$

Reorganizing the terms in Eqn A7 we have

$$U = \sum_i \sum_j |i\rangle \langle i| \otimes |j\rangle \langle j| \otimes U_{ij}, \quad (\text{A8})$$

where

$$U_{ij} := [H \otimes I \otimes I] \cdot [|0\rangle \langle 0| \otimes \mathcal{K}_i \otimes \mathcal{Q}_j + |1\rangle \langle 1| \otimes \mathcal{Q}_j \otimes \mathcal{K}_i] \cdot [H \otimes I \otimes I], \quad (\text{A9})$$

Define $|\phi_{ij}\rangle := U_{ij} |0\rangle |0\rangle_{\mathbb{K}}^n |0\rangle_{\mathbb{Q}}^n$ and we have:

$$|\phi_{ij}\rangle = \frac{1}{\sqrt{2}} (|+\rangle |k_i\rangle |q_j\rangle + |-\rangle |q_j\rangle |k_i\rangle). \quad (\text{A10})$$

Expanding and rearranging the terms in Eq. A10 we have

$$|\phi_{ij}\rangle = \frac{1}{2} (|0\rangle \otimes (|k_i\rangle |q_j\rangle + |q_j\rangle |k_i\rangle) + |1\rangle \otimes (|k_i\rangle |q_j\rangle - |q_j\rangle |k_i\rangle)). \quad (\text{A11})$$

Denote $|u_{ij}\rangle$ and $|v_{ij}\rangle$ as the normalized states of $|k_i\rangle |q_j\rangle + |q_j\rangle |k_i\rangle$ and $|k_i\rangle |q_j\rangle - |q_j\rangle |k_i\rangle$ respectively. Then there is a real number $\theta_{ij} \in [\pi/4, \pi/2]$ such that

$$|\phi_{ij}\rangle = \sin \theta_{ij} |0\rangle |u_{ij}\rangle + \cos \theta_{ij} |1\rangle |v_{ij}\rangle. \quad (\text{A12})$$

θ_{ij} satisfies $\cos \theta_{ij} = \sqrt{1 - |\langle k_i | q_j \rangle|^2} / \sqrt{2}$, $\sin \theta_{ij} = \sqrt{1 + |\langle k_i | q_j \rangle|^2} / \sqrt{2}$.

The final output state from U , $|\Psi_3\rangle = U |\Psi_0\rangle$, can then be written as

$$|\Psi_3\rangle = \sum_i \sum_j |i\rangle |j\rangle \underbrace{(\sin \theta_{ij} |u_{ij}\rangle |0\rangle + \cos \theta_{ij} |v_{ij}\rangle |1\rangle)}_{|\phi_{ij}\rangle} = \sum_i \sum_j |i\rangle |j\rangle |\phi_{ij}\rangle \quad (\text{A13})$$

Note that $\langle k_i | q_j \rangle = \langle \mathbf{x}_i | U_K^\dagger U_Q | \mathbf{x}_j \rangle = a(\mathbf{x}_i, \mathbf{x}_j)$ being the attention scores are encoded in the amplitudes of the output state $|\Psi_3\rangle$ of swap test as:

$$|\langle k_i | q_j \rangle|^2 = -\cos 2\theta_{ij}. \quad (\text{A14})$$

2. Storing Attention score

The second step is to use amplitude estimation [24] to extract and store the attention scores into an additional register which we call the “amplitude register”.

After step 1, we introduce an extra register $|0\rangle_{\text{amplitude}}^t$ and the output state $|\Psi_3\rangle$ (using the same notation) becomes

$$|\Psi_3\rangle = \sum_i \sum_j |i\rangle |j\rangle |\phi_{ij}\rangle |0\rangle_{\text{amplitude}}^t, \quad (\text{A15})$$

where $|\phi_{ij}\rangle$ can be decomposed as

$$|\phi_{ij}\rangle = \frac{-i}{\sqrt{2}} \left(e^{i\theta_{ij}} |\omega_+\rangle_{ij} - e^{i(-\theta_{ij})} |\omega_-\rangle_{ij} \right). \quad (\text{A16})$$

Hence, we have

$$|\Psi_3\rangle = \sum_i \sum_j \frac{-i}{\sqrt{2}} \left(e^{i\theta_{ij}} |i\rangle |j\rangle |\omega_+\rangle_{ij} - e^{i(-\theta_{ij})} |i\rangle |j\rangle |\omega_-\rangle_{ij} \right) |0\rangle_{\text{amplitude}}^t. \quad (\text{A17})$$

The overall Grover operator G is defined as

$$G := UC_2U^\dagger C_1, \quad (\text{A18})$$

where C_1 is the Z gate on the swap ancilla qubit, and $C_2 = I - 2|0\rangle\langle 0|$ is the “flip zero state” on registers other than the two registers hosting indices i, j (represented as S_0 in Fig.12). It can be shown that G can be expressed as

$$G = \sum_i \sum_j |i\rangle |j\rangle \langle j| \langle i| \otimes G_{ij}, \quad (\text{A19})$$

where G_{ij} is defined as

$$G_{ij} = (I - 2|\phi_{ij}\rangle\langle\phi_{ij}|)C_1 \quad (\text{A20})$$

It is easy to check that $|w_\pm\rangle_{ij}$ are the eigenstates of G_{ij} , that is,

$$G_{ij}|w_\pm\rangle_{ij} = e^{\pm i2\theta_{ij}}|w_\pm\rangle_{ij}. \quad (\text{A21})$$

The overall Grover operator G possesses the following eigen-relation:

$$G|i\rangle |j\rangle |\omega_\pm\rangle_{ij} = e^{i(\pm 2\theta_{ij})} |i\rangle |j\rangle |\omega_\pm\rangle_{ij}. \quad (\text{A22})$$

Next, we apply phase estimation of the overall Grover operator G on the input state $|\Psi_3\rangle$. The resulting state $|\Psi_4\rangle$ can be written as

$$|\Psi_4\rangle = \sum_i \sum_j \frac{-i}{\sqrt{2}} \left(e^{i\theta_{ij}} |i\rangle |j\rangle |\omega_+\rangle_{ij} |2\theta_{ij}\rangle - e^{i(-\theta_{ij})} |i\rangle |j\rangle |\omega_-\rangle_{ij} |-2\theta_{ij}\rangle \right). \quad (\text{A23})$$

Note here in Eq. A23, $|\pm 2\theta_{ij}\rangle$ denotes the eigenvalues $\pm 2\theta_{ij}$ being stored in the amplitude register with some finite precision.

Next, we apply an oracle U_O (implemented by arithmetic circuit) on the amplitude register and an extra ancilla register, which acts as

$$U_O |0\rangle |\pm 2\theta_{ij}\rangle = |a(\mathbf{x}_i, \mathbf{x}_j)\rangle |\pm 2\theta_{ij}\rangle, \quad (\text{A24})$$

The state after the oracle can be written as

$$|\Psi_5\rangle = \sum_i \sum_j \frac{-i}{\sqrt{2}} |a(\mathbf{x}_i, \mathbf{x}_j)\rangle \left(e^{i\theta_{ij}} |i\rangle |j\rangle |\omega_+\rangle_{ij} |2\theta_{ij}\rangle - e^{i(-\theta_{ij})} |i\rangle |j\rangle |\omega_-\rangle_{ij} |-2\theta_{ij}\rangle \right). \quad (\text{A25})$$

Then we perform the uncomputation of Phase estimation, the resulting state is

$$|\Psi_6\rangle = \sum_i \sum_j \frac{-i}{\sqrt{2}} |a(\mathbf{x}_i, \mathbf{x}_j)\rangle \left(e^{i\theta_{ij}} |i\rangle |j\rangle |\omega_+\rangle_{ij} |0\rangle_{\text{amplitude}}^t - e^{i(-\theta_{ij})} |i\rangle |j\rangle |\omega_-\rangle_{ij} |0\rangle_{\text{amplitude}}^t \right) \quad (\text{A26})$$

$$= \sum_i \sum_j |a(\mathbf{x}_i, \mathbf{x}_j)\rangle |i\rangle |j\rangle |\phi_{ij}\rangle |0\rangle_{\text{amplitude}}^t \quad (\text{A27})$$

Finally, we perform the uncomputation of the swap test and the resulting state is

$$|\Psi_7\rangle = \sum_i \sum_j |a(\mathbf{x}_i, \mathbf{x}_j)\rangle |i\rangle |j\rangle |0\rangle |0\rangle_{\text{amplitude}}^t. \quad (\text{A28})$$

The above steps, as illustrated in Fig. 12, implemented the quantum attention oracle $O_{\text{attention}}$ such that:

$$O_{\text{attention}} |i\rangle |j\rangle |0\rangle \rightarrow |i\rangle |j\rangle |a(\mathbf{x}_i, \mathbf{x}_j)\rangle \quad (\text{A29})$$

B. Positional encoding via quantum circuit

The positional encoding mentioned in Section 2.1 can be described as follows[9]: Corresponding to the i -th vector in the sequence $\mathbf{x}_i \in \mathbb{R}^d$, define the position vector $\mathbf{p}_i \in \mathbb{R}^d$ as:

$$\begin{cases} \mathbf{p}_i^{(2j+1)} := \cos\left(\frac{i}{10000^{\frac{2j}{d}}}\right) \\ \mathbf{p}_i^{(2j)} := \sin\left(\frac{i}{10000^{\frac{2j}{d}}}\right) \end{cases} \quad (\text{B1})$$

for all $j \in \{0, 1, \dots, \lfloor d/2 \rfloor\}$, where $\mathbf{p}_i^{(2j+1)}$ and $\mathbf{p}_i^{(2j)}$ denote the odd and even elements of \mathbf{p}_i , respectively. For encoding of positional information into data, the position vectors are added directly to the input vectors:

$$\mathbf{x}'_i = \mathbf{x}_i + \mathbf{p}_i, \quad (\text{B2})$$

where \mathbf{x}'_i is the appended input vectors that contain positional information, and we define $\mathbf{X}' := [\mathbf{x}'_1, \dots, \mathbf{x}'_n] \in \mathbb{R}^{d \times n}$. as the matrix by stacking \mathbf{x}'_i .

Our quantum algorithm for positional encoding aims to create the quantum state

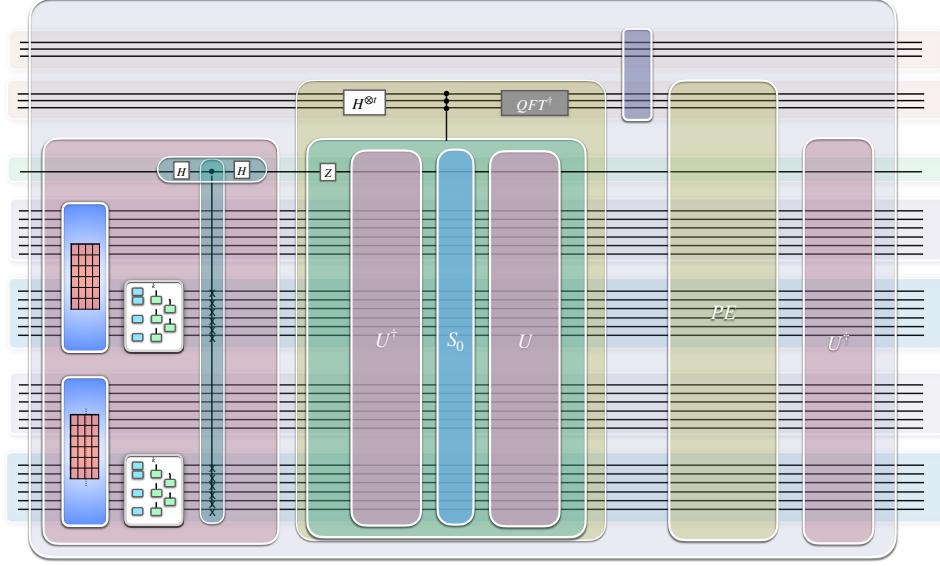


FIG. 12. *Quantum attention oracle* $O_{\text{attention}}$ The quantum attention oracle aims to coherently evaluate and store attention score $a(\mathbf{x}_i, \mathbf{x}_j)$ for each pair of the input vectors, it acts as $O_{\text{attention}} |i\rangle |j\rangle |0\rangle \rightarrow |i\rangle |j\rangle |a(\mathbf{x}_i, \mathbf{x}_j)\rangle$. The construction of the quantum attention oracle, depicted in this figure, is detailed in the appendix subsections 1 and 2.

$$|\psi_{\mathbf{X}'}\rangle := \sum_{i=1}^n |i\rangle |\mathbf{x}'_i\rangle, \quad (\text{B3})$$

where $|\mathbf{x}'_i\rangle := \sum_{k=1}^d \mathbf{x}'_i^{(k)} |k\rangle$ is the amplitude encoding of the vector \mathbf{x}'_i whose k -th elements are denoted as $\mathbf{x}'_i^{(k)}$. Similarly, we define $|\mathbf{p}_i\rangle := \sum_{k=1}^d \mathbf{p}_i^{(k)} |k\rangle$ which is the amplitude encoding of the vector \mathbf{p}_i whose k -th elements are denoted as $\mathbf{p}_i^{(k)}$ and

$$|\psi_{\mathbf{P}}\rangle := \sum_{i=1}^n |i\rangle |\mathbf{p}_i\rangle \quad (\text{B4})$$

From the above definitions of $|\mathbf{x}'_i\rangle$, $|\mathbf{p}_i\rangle$ and Eqn. 2,B2,B3,B4 we have

$$|\psi_{\mathbf{X}'}\rangle = |\psi_{\mathbf{X}}\rangle + |\psi_{\mathbf{P}}\rangle = \sum_{i=1}^n |i\rangle |\mathbf{x}_i\rangle + \sum_{i=1}^n |i\rangle |\mathbf{p}_i\rangle \quad (\text{B5})$$

Note that

$$|\mathbf{p}_i\rangle = \sum_{k=1}^d \mathbf{p}_i^{(k)} |k\rangle = \sum_j (\mathbf{p}_i^{(2j)} |2j\rangle + \mathbf{p}_i^{(2j+1)} |2j+1\rangle) \quad (\text{B6})$$

Denote the unitary that prepares $|\psi_{\mathbf{P}}\rangle$ from $\sum_{i=1}^n |i\rangle |0\rangle$ as $U_{\mathbf{P}}$, then $|\psi_{\mathbf{X}'}\rangle$ can be achieved by applying LCU to $U_{\mathbf{X}}, U_{\mathbf{P}}$. As the construction of $U_{\mathbf{X}}$ is given by the CQSP mentioned in Section 3.1, we can focus on the construction of $U_{\mathbf{P}}$. Next, we present the construction of $U_{\mathbf{P}}$ as depicted in Fig.13.

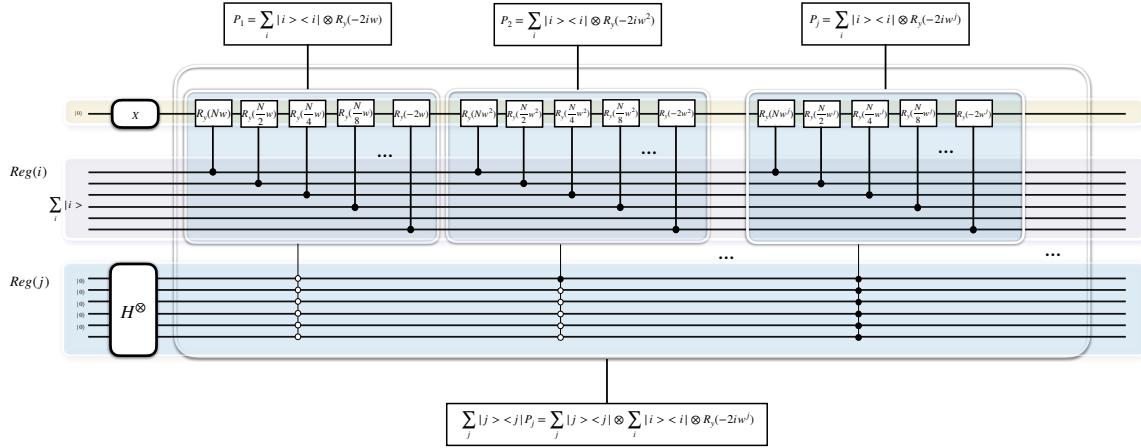


FIG. 13. *Quantum circuit for positional encoding* Build upon the two registers $Reg(i)$ and $Reg(k)$ hosting the index i and k respectively (illustrated in Fig. 6), we set up a register $Reg(j)$ hosting the index j below $Reg(i)$ and an ancillary qubit above $Reg(i)$. The blue boxes that group the series of controlled R_Y gates implement the following unitaries: $P_j = \sum_i |i\rangle\langle i| \otimes R_y(-2iw^j)$, each of which is controlled by the qubits in $Reg(j)$ and the entire controlled sequences grouped in the transparent box implement the unitary $U_c = \sum_j |j\rangle\langle j| \otimes P_j = \sum_j |j\rangle\langle j| \otimes \sum_i |i\rangle\langle i| \otimes R_y(-2iw^j)$. The whole circuit implements U_P . Note that in this figure, $N = -n$.

Build upon the two registers $Reg(i)$ and $Reg(k)$ hosting the index i and k respectively (illustrated in Fig. 6), in Fig. 13, we set up a register $Reg(j)$ hosting the index j below $Reg(i)$ and an ancillary qubit above $Reg(i)$. For reasons that will be clear soon, we combine $Reg(j)$ and the ancillary qubit as a single register which coincides with $Reg(k)$. The blue boxes that group the series of controlled R_Y gates implement the following unitaries:

$$P_j = \sum_i |i\rangle\langle i| \otimes R_y(-2iw^j) \quad (B7)$$

each of which is controlled by the qubits in $Reg(j)$, and the entire controlled sequences grouped in the transparent box implement the unitary

$$U_c = \sum_j |j\rangle\langle j| \otimes P_j = \sum_j |j\rangle\langle j| \otimes \sum_i |i\rangle\langle i| \otimes R_y(-2iw^j) \quad (B8)$$

The Hadamard gates and X gate before U_c transform the input state to the state $\sum_j |j\rangle \otimes \sum_i |i\rangle \otimes |1\rangle$, after U_c , it becomes

$$U_c(\sum_j |j\rangle \otimes \sum_i |i\rangle \otimes |1\rangle) = \sum_j |j\rangle \otimes \sum_i |i\rangle \otimes R_y(-2iw^j) |1\rangle \quad (B9)$$

By placing $Reg(j)$ and the ancillary qubit next to each other we can rewrite the above state as:

$$\sum_i |i\rangle \otimes \sum_j |j\rangle \otimes R_y(-2iw^j) |1\rangle \quad (B10)$$

$$= \sum_i |i\rangle \otimes \sum_j |j\rangle \otimes (\sin(iw^j) |0\rangle + \cos(iw^j) |1\rangle). \quad (B11)$$

Combining $Reg(j)$ and the ancillary qubit as a single register that coincides with $Reg(k)$, the computational basis transform as $|j\rangle|0\rangle \rightarrow |2j\rangle, |j\rangle|1\rangle \rightarrow |2j+1\rangle$ and we can write the output state from the circuit in Fig.13 as:

$$|\text{output}\rangle = \sum_i |i\rangle \otimes \sum_j (\sin(iw^j) |2j\rangle + \cos(iw^j) |2j+1\rangle) \quad (\text{B12})$$

set $w = \frac{1}{10000^{\frac{1}{d}}}$, and from Eqn. B1,B4,B6,B12 we have

$$|\text{output}\rangle = |\psi_P\rangle \quad (\text{B13})$$

That is, the circuit in Fig.13 implements U_P .

C. Masked-Attention

This section presents the masking operation in attention and its quantum implementation.

The masked attention is defined as:[9]

$$\begin{aligned} \mathbb{R}^{r \times n} \ni \mathbf{Z}_m &:= \text{maskedAttention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) \\ &= \mathbf{V} \text{ softmax} \left(\frac{1}{\sqrt{p}} (\mathbf{Q}^\top \mathbf{K} + \mathbf{M}) \right), \end{aligned}$$

where the mask matrix $\mathbf{M} \in \mathbb{R}^{n \times n}$ is:

$$\mathbf{M}_{ij} := \begin{cases} 0 & \text{if } j \leq i \\ -\infty & \text{if } j > i \end{cases}$$

For positions $j \leq i$ in a sequence (representing current or previous words), the mask doesn't alter the softmax output, this allows these positions to contribute to the softmax computation. In contrast, for positions $j > i$ (corresponding to future words in the sequence), the nature of softmax function indicates that the mask effectively nullifies their contribution as $e^{-\infty}$ is 0. This selective masking ensures that the model's attention is appropriately focused on the relevant parts of the sequence: considering past and present words while ignoring future words.

Denoting $\mathbf{A}_0^{\text{Mask}} \equiv \text{softmax} \left(\frac{1}{\sqrt{p}} (\mathbf{Q}^\top \mathbf{K} + \mathbf{M}) \right)$, we have its elements as

$$\mathbf{A}_0^{\text{Mask}}_{ij} := \begin{cases} \mathbf{A}_{0ij} & \text{if } j \leq i \\ 0 & \text{if } j > i \end{cases}$$

where \mathbf{A}_{0ij} is the elements of $\mathbf{A}_0 \equiv \text{softmax} \left(\frac{1}{\sqrt{p}} \mathbf{Q}^\top \mathbf{K} \right)$.

In the quantum case, we aim to implement an alternative version of $\mathbf{A}_0^{\text{Mask}}$ denoted as \mathbf{A}^{Mask} whose elements are defined as

$$\mathbf{A}^{\text{Mask}}_{ij} := \begin{cases} \mathbf{A}_{ij} & \text{if } j \leq i \\ 0 & \text{if } j > i \end{cases}$$

where \mathbf{A}_{ij} is the elements of $\mathbf{A} \equiv \mathbf{Q}^\top \mathbf{K}$.

For masked-attention, we aim to design a quantum circuit implementing the following computation:

$$\mathbf{Z}'_m = \mathbf{W}_V^\top \mathbf{X} \mathbf{A}^{\text{Mask}}, \quad (\text{C1})$$

This can be done in a similar way as in the case of self-attention where we implemented Eqn.12, the only difference is that we now need the block-encoding of $\mathbf{A}^{\text{Mask}}{}^\top$ (instead of \mathbf{A}^\top) whose elements are

$$\Lambda_{ij}^{\text{Mask}} := \mathbf{A}^{\text{Mask}}{}^\top_{ij} := \begin{cases} 0 & \text{if } j < i \\ \mathbf{A}_{ji} & \text{if } j \geq i \end{cases}$$

Similar to the case of self-attention, let $\hat{\Lambda}^{\text{Mask}} \geq \max_{i,j} |\Lambda_{ij}^{\text{Mask}}|$ and $\tilde{\Lambda}_{ij}^{\text{Mask}}$ is defined to be the (exact) b-qubit description of $\Lambda_{ij}^{\text{Mask}}/\hat{\Lambda}^{\text{Mask}}$. The block-encoding of $\mathbf{A}^{\text{Mask}}{}^\top$ can be constructed using lemma 3.2 from Ref.[20], given the following oracle

$$O_{\mathbf{A}^{\text{Mask}}{}^\top} : |i\rangle|j\rangle|0\rangle^{\otimes b} \rightarrow |i\rangle|j\rangle|\tilde{\Lambda}_{ij}^{\text{Mask}}\rangle, \quad (\text{C2})$$

where $0 \leq i, j < n$.

This oracle $O_{\mathbf{A}^{\text{Mask}}{}^\top}$ can be constructed by conditionally applying $O_{\mathbf{A}^\top}$: on the registers hosting $|i\rangle|j\rangle$, set up a circuit comparing the values of i, j with the result stored in an extra ancillary qubit (using Claim 3.1 in Ref.[25]). Then using this ancillary qubit as control qubit, apply controlled $O_{\mathbf{A}^\top}$ if $j \geq i$.

D. Block-encoding

Block encoding is a powerful modern quantum algorithmic technique that is employed in a variety of quantum algorithms for solving linear algebra problems on a quantum computer[27]. A unitary U is a block encoding of a not-necessarily-unitary square matrix A (A is scaled to satisfy $\|A\|_2 \leq 1$ [27]) if A is encoded in the top-left block of the unitary U as:

$$U = \begin{bmatrix} A & \cdot \\ \cdot & \cdot \end{bmatrix},$$

where the \cdot symbol stands for a matrix block. Equivalently, we can write

$$A = \left(\langle 0|^{\otimes a} \otimes I \right) U \left(|0\rangle^{\otimes a} \otimes I \right) \quad (\text{D1})$$

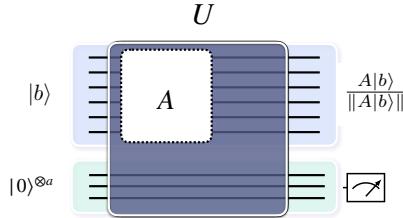


FIG. 14. *Block-encoding* U , the Block-encoding of a matrix A , can be considered as a probabilistic implementation of A : applying the unitary U to a given input state $|0\rangle^{\otimes a}|b\rangle$, measuring the first a -qubit register and post-selecting on the outcome $|0\rangle^{\otimes a}$, we get state proportional to $A|b\rangle$ in the second register.

where a is the number of ancilla qubits used for the block encoding of A . U can be considered as a probabilistic implementation of A : by applying the unitary U to an input state $|0\rangle^{\otimes a}|b\rangle$, measuring the first a -qubit register and post-selecting on the outcome $|0\rangle^{\otimes a}$, we obtain a state that is proportional to $A|b\rangle$ in the second register. This can be illustrated in Fig.14.

The circuit implementation of Block-encoding in general can be constructed using the Linear Combination of Unitaries(LCU)[28] technique.[29]

E. Parametrized quantum circuit for implementing trainable linear layer

Classical neural networks are fundamentally built on the structure of multilayer perceptrons which involve layers of trainable linear transformations and element-wise nonlinear transformations (activation functions such as ReLU, sigmoid, or tanh). On the other hand, Quantum Neural Networks (QNNs), which are often defined as parametrized quantum circuits with a predefined circuit ansatz, do not naturally exhibit this kind of structure. In QML literature, a QNN, denoted as $U(\boldsymbol{\theta})$, often has a structure of layers L of the form [30]

$$U(\boldsymbol{\theta}) = \prod_{l=1}^L U_l(\boldsymbol{\theta}_l), \quad U_l(\boldsymbol{\theta}_l) = \prod_{k=1}^K e^{-i\theta_{lk} H_k}, \quad (\text{E1})$$

where the index l represents the layer, and the index k covers the Hermitian operators H_k that generates the unitaries in the circuit ansatz, $\boldsymbol{\theta}_l = (\theta_{l1}, \dots, \theta_{lK})$ represents the parameters in a single layer, and $\boldsymbol{\theta} = \{\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_L\}$ represents the collection of adjustable parameters in the QNN. Examples of circuit ansatz represented by Eq. E1 include: the hardware-efficient ansatz [31], quantum alternating operator ansatz [32], and quantum optimal control Ansatz [33], among others.

In machine learning, a linear layer [34] is a fundamental component of neural networks architectures that maps input vectors to output vectors through affine transformation: Given an input vector $\mathbf{x} \in \mathbb{R}^n$, the linear layer transforms it to an output vector $\mathbf{y} \in \mathbb{R}^m$ using a weight matrix $W \in \mathbb{R}^{m \times n}$ and an optional bias vector $\mathbf{b} \in \mathbb{R}^m$ as $\mathbf{y} = W\mathbf{x} + \mathbf{b}$, and W , b contain the trainable parameters.

The input vector \mathbf{x} can be encoded in the amplitudes of a s -qubit quantum state ($n = 2^s$) $|\psi_x\rangle$ as $|\psi_x\rangle = \sum_{i=1}^n x_i |i\rangle$ where x_i is the i -th element of \mathbf{x} (after normalization), and $|i\rangle$ is the i -th compu-

tational basis state. Applying a parameterized quantum circuit $U(\boldsymbol{\theta}) \in \mathbb{C}^{n \times n}$ on $|\psi_x\rangle$ be interpreted as a special type of linear layer (represented as a square matrix) on \mathbf{x} . In this paper, we utilize parameterized quantum circuits as in Eq. E1 for implementing such linear layers. Note that in case where the weight matrix $W \in \mathbb{R}^{m \times n}$ is rectangular ($m \neq n$), by default we adjust the dimension of the output vector to be the same as the input vector in our quantum architecture, without specifying the adjustment in the prior description of the classical architecture.

It should be emphasized that the trainable circuit parameters $\boldsymbol{\theta}$ are not equivalent to the weights in the weight matrix, but rather are parameterized by them, as in $W(\boldsymbol{\theta})$. Ref [30] contains a discussion of the parametrization.