QUAN-CEFT: QUANTUM-INSPIRED COMPUTATIONAL EFFICIENT FINE-TUNING VIA TENSOR RING VARIATIONAL GENERATORS

Anonymous authors
Paper under double-blind review

ABSTRACT

Advancements in Generative Artificial Intelligence have led to rapid progress in Large Language Models (LLMs). However, training and fine-tuning LLMs presents its own set of challenges. The immense number of parameters in these models requires vast computational resources, making conventional Parameter Efficient Fine-Tuning approaches and adapters computationally expensive and impractical for many use cases. Recently, Quantum-inspired Parameter Adaptation (QPA) methods have demonstrated notable parameter reduction in Small-scale Language Models (SLMs). However, their practical application is hindered by challenges in trainability and scalability of Variational Quantum Circuits (VQC). To address these issues, we proposed a novel and scalable Quantum-inspired Computational Efficient Fine-Tuning (Quan-CEFT)[1] framework introducing a Tensor Ring simulation of Hamming-weight preserving variational Quantum Generator (TR-HQG) circuit, fully delegating the generation of VQC parameters to the TR network. As all trainable parameters reside in the classical TR cores and a Tensor Train mapping network, the circuit acts only as a fixed forward-pass evaluator, which is designed to mitigate barren plateaus and to improve training convergence, thereby enabling computationally efficient fine-tuning for SLMs. On WikiText-2 causal language modeling, the proposed Quan-CEFT framework uses $8.1\times$ to $60.5\times$ fewer trainable parameters than rank-4 LoRA and 2-$6\times$ fewer than the most compact quantum baseline across GPT-2, OPT-125M, Pythia-160M, and TinyLlama-1.1B, with 5 to 29% lower per-step training latency than QPA in CPU simulation, while retaining competitive perplexity and ROUGE-L under a short, five-step CPU adaptation budget. The reduced parameter footprint and faster CPU training of Quan-CEFT nevertheless show promise towards on-device fine-tuning on resource-constrained devices.

1 INTRODUCTION

Small-scale pre-trained language models like GPT-2 Radford et al. (2019), OPT Zhang et al. (2022), Pythia Biderman et al. (2023), and TinyLlama Zhang et al. (2024) have achieved state-of-the-art performance on various language, vision and multimodal tasks. However, fine-tuning these models for specific downstream tasks typically requires updating all model parameters at a lower learning rate, which becomes computationally expensive as model sizes grow to billions of parameters (e.g., Llama-2 series models) Touvron et al. (2023). This has led to the development of Parameter-Efficient Fine-Tuning (PEFT) methods Houlsby et al. (2019), which adapt large foundation models to new tasks by updating only a small subset of parameters or introducing lightweight adaptation modules. One of the most popular PEFT techniques is Low-Rank Adaptation (LoRA) Hu et al. (2022), which injects low-rank trainable matrices into transformer layers, significantly reducing the number of parameters that need to be updated. Other methods, such as Adapters Houlsby et al. (2019), BitFit Zaken et al. (2022), and Prompt Tuning Lester et al. (2021), have also shown effectiveness in various settings. Despite their efficiency, these methods often involve a trade-off between parameter savings and downstream performance and can be sensitive to choices such as the adaptation mechanism, its placement within the model, and the number of trainable parameters. More recently, Orthogonal

[1] The Pytorch implementation of our Quan-CEFT is available in `https://anonymous.4open.science/r/QUAN-CEFT/`

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Fine-Tuning (OFT) Qiu et al. (2023) and its Butterfly specification (BOFT) Liu et al. (2024c) have been proposed to mitigate catastrophic forgetting during fine-tuning by applying orthogonal transformations. While these methods show promising performance–efficiency trade-offs, the required transformations can introduce computational overhead, limiting their practicality in some use cases. Of late, Quantum Parameter Adaptation (QPA) Liu et al. (2025), Quantum-PEFT Koike-Akino et al. (2025) and Quantum-inspired Adapter (QiA)-based Raj & Coyle (2025) fine-tuning approaches have shown significant reduction of parameters in benchmarking with relatively small-scale SLMs (GPT-2, Gemma-2). Although these approaches hold considerable promise, their practical implementation faces significant challenges specific to trainability and scalability for SLMs with billion-scale parameters. Key obstacles for the QPA, Quantum-PEFT and QiA models include the computational cost of CNOT and SWAP gates in VQCs Peddireddy et al. (2023); Konar et al. (2025), as well as the barren plateau phenomenon in Hamming Weight Preserving (HWP) quantum circuits Kerenidis & Prakash (2022); Cherrat et al. (2023). In barren plateaus, gradients vanish exponentially with increasing qubit count or circuit depth, severely compromising result fidelity and destabilizing training Larocca et al. (2025). A qualitative comparison is provided in Tab. 1.

Table 1: Qualitative comparison of adaptation methods: LoRA, QPA, Quantum-PEFT (Q-PEFT), and our Quan-CEFT (!-CPU simulations are often timed out)

Method	CPU Simulations	Parameters	Training Time	Memory	Latency
LoRA	!	Very High	Moderate	Very High	Moderate
QPA	!	High	High	Low	High
Q-PEFT	!	Moderate	Very High	Low	Very High
Quan-CEFT	✓	Low	Low	Low	Low

To enable scalable training of HWP quantum circuits built from Reconfigurable Beam Splitter (RBS) gates for fine-tuning Raj & Coyle (2025), we introduce a Tensor Ring (TR) simulated Hamming-weight preserving variational Quantum Generator (TR-HQG), enabling computationally efficient training of HWP circuits using TR. Our approach also simplifies the RBS gate operation through TR-approximation, enabling computationally efficient fine-tuning for Generative Artificial Intelligence models. Here, the TR serves dual purposes: (1) training the HWP generator and (2) simulating its two-qubit RBS gates. The significant contributions are as follows:

1. We propose a novel TR-HQG circuit that obviates the expensive training of HWP circuits and RBS gate operations. An n-qubit TR-HQG exposes 2^n measurement bases, each of which is expanded by a classical mapping network into a chunk of target parameters, so that the number of simulated qubits grows only logarithmically with the number of generated adapter parameters (Subsec. A.5).
2. Our Quan-CEFT with TR-HQG is free from classical-to-quantum data encoding (the generator operates on a fixed reference state and no activations are loaded into the circuit), and the TR simulation replaces the $\mathcal{O}(2^n)$ statevector by n cores of bond dimension $\mathcal{B}$ with contraction cost $\mathcal{O}(n\mathcal{B}^3)$ (Subsec. A.5.2).
3. We employ a Tensor Train Network (TTN) of fully connected Multi-layer Perceptron (MLP) layers to map the context vectors generated by our TR-HQG circuits to the adapter weight updates, further reducing the parameter count; we give the explicit end-to-end contraction from TR cores to adapter update in Subsec. 4.1.
4. Our experiments on WikiText-2 causal language modeling across four backbones (GPT-2, OPT-125M, Pythia-160M, and TinyLlama-1.1B) demonstrate Quan-CEFT's parameter efficiency, achieving an $8.1\times$ to $60.5\times$ reduction in trainable parameters compared to rank-4 LoRA while preserving competitive perplexity and ROUGE-L under a short CPU budget.

2 Related Work

Parameter-Efficient Fine-Tuning (PEFT): PEFT methods focus on minimizing the number of trainable parameters to tackle the challenge of fine-tuning LLMs, while preserving or even enhancing model performance. Key approaches include LoRA Hu et al. (2022), Weight-Decomposed LoRA (DoRA) Liu et al. (2024b) and NormAL LoRA Aastik et al. (2025), which assume that fine-tuning induced weight changes occur within a low-rank subspace. Another technique involves adapters-small feed-forward layers inserted between existing neural network layers Houlsby et al. (2019); Lin et al. (2020). Prefix-Tuning (PT) Li & Percy (2021) adopts a distinct strategy incorporating tunable prefix vectors prepended to input or hidden states at each layer. These methods enhance

fine-tuning efficiency by concentrating on smaller, task-specific modifications. Of late, Liu et al. (2022) present a novel PEFT technique, which scales activations using learned vectors, achieving enhanced performance with the addition of only a minimal number of new parameters.

Orthogonal and Ultra Low Parameter Adapters: OFT Qiu et al. (2023) re-parameterizes the update as a multiplicative orthogonal transformation of the pre-trained weight, which preserves the pairwise angles between neurons and thereby mitigates catastrophic forgetting; Butterfly OFT (BOFT) Liu et al. (2024c) factorizes the orthogonal matrix into a product of sparse butterfly blocks so that the transformation can be expressed with far fewer parameters. Both, however, require materializing or applying dense (or butterfly-structured) orthogonal transformations at every adapted projection, whose cost grows with the hidden dimension. Quantum-inspired Adapters (QiA) Raj & Coyle (2025) inherit this multiplicative, orthogonality-preserving design through compound matrices. A complementary line of purely classical work without any circuit simulation: Vector-based Random-matrix Adaptation (VeRA) Kopiczko et al. (2024) shares frozen random low-rank factors across layers and learns only per-layer scaling vectors, Kronecker Adapters (KronA) Edalati et al. (2022) factor the update as a Kronecker product, and Tensor train adaptation (LoRETTA) Yang et al. (2024) represents the update as a low-rank tensor train. Despite the ultra low-parameter reduction, these methods suffer from high computational cost in training.

Quantum Circuit-based Parameter Efficient Fine-Tuning: Recently, QPA Liu et al. (2025), Quantum-PEFT Koike-Akino et al. (2025), and QiA-based Raj & Coyle (2025) fine-tuning approaches show great promise to reduce the number of fine-tune parameters for SLMs (Gemma-2, GPT-2). The QPA framework combines Quantum Neural Networks (QNNs) with a classical MLP mapping model, eliminating the need for classical to quantum data encoding to produce parameters for fine-tuning techniques. In Quantum-PEFT, with the use of Pauli parameterization, the number of trainable parameters grows only logarithmically with the ambient dimension, as opposed to linearly as in LoRA-based PEFT methods. In the QiA-based fine-tuning method, it introduces a classically efficient design of QiA utilizing compound matrices, demonstrating their parameter efficiency and orthogonality properties.

However, these models suffer from high computational overhead due to the high complexity of quantum circuit simulation (detailed in Subsec A.1), which is not feasible for fine-tuning a large number of parameters on resource-constrained devices. To circumvent this computational complexity, we introduce a low-rank TR network simulation of a scalable HWP quantum circuit for computationally efficient fine-tuning of SLMs.

3 Low-Rank Tensor Ring Networks

Tensor networks are a collection of tensor factorization models that utilize multiple sparsely connected low-order tensors, including TT decomposition Oseledets (2011), TR decomposition Yuan et al. (2019), and Hierarchical Tucker (HT) decomposition Song et al. (2020), to represent a higher-order tensor. Our work primarily focuses on TR decomposition and the contraction operator, which are discussed in more detail in subsequent section. The circular multi-linear product involving a sequence of third-order core tensors, depicted in Fig. 1, is defined as follows:

$$\mathcal{T}_{\mathcal{R}}(x_1, x_2, \cdots, x_{\mathcal{N}}) = \mathcal{TR}\left(\prod_{\iota=1}^{\mathcal{N}} \mathcal{G}(\iota)^{x_\iota}_{b_\iota b_{\iota+1}}\right) = \sum_{b_1,\cdots,b_{\mathcal{N}}=1}^{\mathcal{I}_1,\cdots,\mathcal{I}_{\mathcal{N}}} \mathcal{G}(1)^{x_1}_{b_1 b_2} \cdots \mathcal{G}(\mathcal{N})^{x_{\mathcal{N}}}_{b_{\mathcal{N}} b_1} \tag{1}$$

where $\mathcal{G}(\iota) \in \mathbb{R}^{\mathcal{I}_\iota \times \mathcal{L}_\iota \times \mathcal{I}_{\iota+1}}$, $\iota \in [\mathcal{N}]$ and $\mathcal{I}_{\mathcal{N}+1} = \mathcal{I}_1$. The TR decomposition represents an $\mathcal{N}^{th}$-order tensor $\mathcal{T}_R \in \mathbb{R}^{\mathcal{L}_1 \times \cdots \times \mathcal{L}_{\mathcal{N}}}$ Qiu et al. (2022).

3.1 Tensor Ring Contraction

TR contraction is a fundamental operation in the TR decomposition process, symbolized by a connected edge linking two nodes. In particular, the contraction of two TR-cores, $\mathcal{G}(\iota)$ and $\mathcal{G}(\iota+1)$, with a shared dimension $b_{\iota+1}$, can be calculated utilizing Yu et al. (2021) as:

$$\mathcal{X}(\iota, \iota+1)^{x_\iota x_{\iota+1}}_{b_\iota b_{\iota+2}} = \sum_{b_{\iota+1}=1}^{\mathcal{I}_{\iota+1}} \mathcal{G}(\iota)^{x_\iota}_{b_\iota b_{\iota+1}} \mathcal{G}(\iota+1)^{x_{\iota+1}}_{b_{\iota+1} b_{\iota+2}}, \tag{2}$$

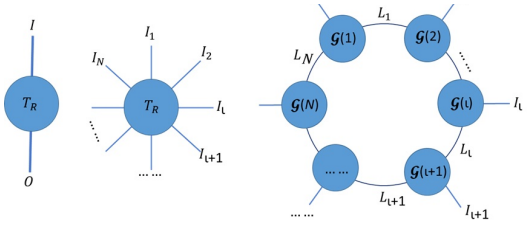

Figure 1: Illustration of Tensor Ring (TR) structure.

where the tensors of $\mathcal{G}(\iota)$ and $\mathcal{G}(\iota+1)$ is contracted as $\mathcal{X}$.
The contraction of three TR cores, $\mathcal{G}(\iota-1)$, $\mathcal{G}(\iota)$, and $\mathcal{G}(\iota+1)$, with shared modes b_ι and $b_{\iota+1}$, can be contracted as follows.

$$\mathcal{X}(\iota-1,\iota,\iota+1)^{x_{\iota-1}x_\iota x_{\iota+1}}_{b_{\iota-1}b_{\iota+2}} = \sum_{b_\iota,b_{\iota+1}=1}^{\mathcal{I}_\iota,\mathcal{I}_{\iota+1}} \mathcal{G}(\iota-1)^{x_{\iota-1}}_{b_{\iota-1}b_\iota}\mathcal{G}(\iota)^{x_\iota}_{b_\iota b_{\iota+1}}\mathcal{G}(\iota+1)^{x_{\iota+1}}_{b_{\iota+1}b_{\iota+2}}. \quad (3)$$

The associativity of these pairwise contractions allows us to efficiently handle more complex scenarios involving multiple TR cores and shared dimensions. Contracting all TR-cores along the common bond modes $b_1, b_2, \cdots, b_\mathcal{N}$ recovers the full tensor as a circular trace of the slice matrices $[\mathbf{G}(\iota)^{x_\iota}]_{b_\iota b_{\iota+1}} = \mathcal{G}(\iota)^{x_\iota}_{b_\iota b_{\iota+1}}$:

$$\mathcal{T}_\mathcal{R}(x_1,x_2,\cdots,x_\mathcal{N}) = \sum_{b_1=1}^{\mathcal{I}_1}\cdots\sum_{b_\mathcal{N}=1}^{\mathcal{I}_\mathcal{N}} \mathcal{G}(1)^{x_1}_{b_1b_2}\,\mathcal{G}(2)^{x_2}_{b_2b_3}\cdots\mathcal{G}(\mathcal{N})^{x_\mathcal{N}}_{b_\mathcal{N}b_1} = \mathrm{Tr}\left(\prod_{\iota=1}^{\mathcal{N}}\mathbf{G}(\iota)^{x_\iota}\right). \quad (4)$$

Three states, including super-critical ($\mathcal{L}_\iota < \mathcal{I}_\iota\mathcal{I}_{\iota+1}$), critical ($\mathcal{L}_\iota = \mathcal{I}_\iota\mathcal{I}_{\iota+1}$), and sub-critical ($\mathcal{L}_\iota > \mathcal{I}_\iota\mathcal{I}_{\iota+1}$), can be identified using TR decomposition. The full Tucker rank tensor data exhibit either critical or supercritical TR states Qiu et al. (2022); intuitively, a super-critical bond dimension is too small to represent the tensor exactly (lossy compression), whereas critical and sub-critical bond dimensions can represent it exactly.

4 Quantum-inspired Computational Efficient Fine-Tuning

We present a novel scalable Quantum-inspired Computational Efficient Fine-Tuning (Quan-CEFT) within the framework of quantum parameter generation, as shown in Fig. 2. The proposed Quan-CEFT combines a novel Tensor Ring simulated Hamming-weight preserving variational Quantum Generator (TR-HQG) with a classical Tensorized multi-layer perceptron mapping model to produce parameters for fine-tuning of Large AI models. Our Quan-CEFT utilizes TR-HQG to generate the weights of a target classical neural network through training of classical TR parameters, thereby transferring the entire parameter optimization process to the classical domain. The quantum circuits serve as a fixed forward pass evaluator, with no parameters within the HQG being trained. Conversely, gradient calculations and parameter updates are performed exclusively in the classical TR-cores using standard back-propagation. In our Quan-CEFT framework, the generated update is merged analytically into the frozen backbone weights so that the deployed model is purely classical without any additional latency.

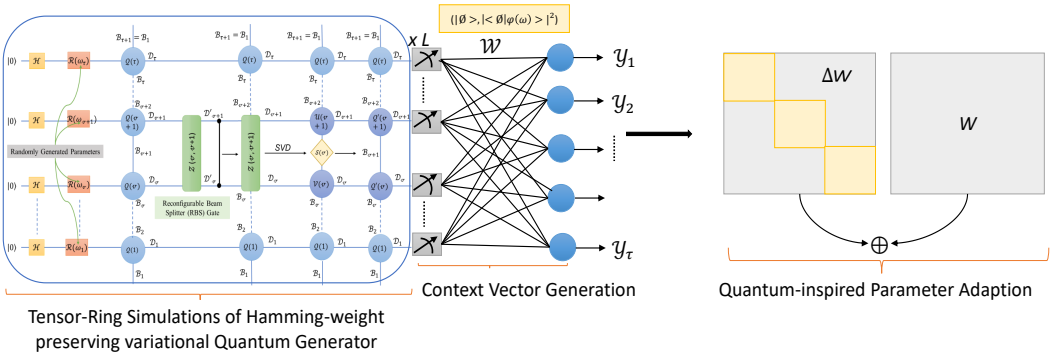

Figure 2: Tensor Ring simulated Hamming-weight Preserving Quantum Circuit used in Quantum Parameter Adaption.

4.1 Tensor Ring Simulated Hamming-weight Preserving Variational Quantum Generator

We introduce a multi-layer TR-simulated HWP generator featuring cascading V-shaped Reconfigurable Beam Splitter (RBS) gates to efficiently manage quantum entanglement among model parameters, distinguishing it from previously proposed TR-VQC Peddireddy et al. (2023); Konar et al. (2025) which suffers from noisy data encoding and training of VQC. The proposed TR-HQG framework comprises four primary components: TR encoding, variational learning parameters (ω) through TR, TR-approximation of RBS gate, and quantum measurement. TR encoding compresses

quantum states into a compact format by leveraging the TR structure. The TR approximation employs Singular Value Decomposition (SVD) to compress quantum states while retaining essential features, enabling efficient representation and manipulation within TR-HQG, as illustrated in Fig. 3. In our TR-HQG, single-qubit rotation gates $\mathcal{R}_y(\omega)$ and $\mathcal{R}_z(\omega)$ are utilized to represent rotations along the Y and Z axes, respectively. The rotation angles (ω) are randomly initialized and learned during training to determine optimal values that minimize the objective function. Integrating TR encoding with variational learning parameters, the proposed TR-HQG framework facilitates the training of quantum circuits through tensor cores parameters. Within TR structures, using τ tensors, each with bond dimension $\mathcal{B}$, denoted by $\mathcal{Q}(\upsilon)$, the TR approximation describes a quantum state $|\psi\rangle$ as follows Aggarwal et al. (2018):

$$|\psi\rangle = \sum_{y_1,\cdots,y_\tau} \sum_{x_1,\cdots,x_\tau} \mathcal{Q}(1)^{y_1}_{x_1 x_2} \mathcal{Q}(2)^{y_2}_{x_2 x_3} \cdots \mathcal{Q}(\tau)^{y_\tau}_{x_\tau x_1} |y_1, \cdots y_\tau\rangle \ . \tag{5}$$

Here, the 2^τ-dimensional Hilbert space is spanned by the physical indices $y_\upsilon \in \{0, 1\}$, and the maximum amount of entanglement recorded by TR-HQG is controlled by the bond indices $x_\upsilon \in \{1, \cdots \mathcal{B}_\upsilon\}$. The entanglement captured by a TR is controlled by the parameter ($\mathcal{B}_\upsilon$). Allowing $\mathcal{B}_\upsilon$ increases enables the TR to capture more information about the quantum state, although at a computational cost. We assume for all $\upsilon, \mathcal{B}_\upsilon{=}\mathcal{B}$, reducing the number of hyper-parameters. The parametrization of a TR with a n-qubit state is illustrated in Fig. 3. In our TR-HQG, each $\mathcal{Q}(\upsilon)$ in TR represents a tensor of dimension $\mathcal{B} \times \mathcal{B} \times \mathcal{D}$, storing the variational parameters associated with the corresponding qubit in HQG. The tensor, $\mathcal{Q}(\upsilon)$, has three indices, two of which have a bond dimension $\mathcal{B}$, and the third index has a dimension $\mathcal{D}$. A RBS gate operation from the final qubit to the initial qubit becomes feasible by employing a cascading configuration of the TR.

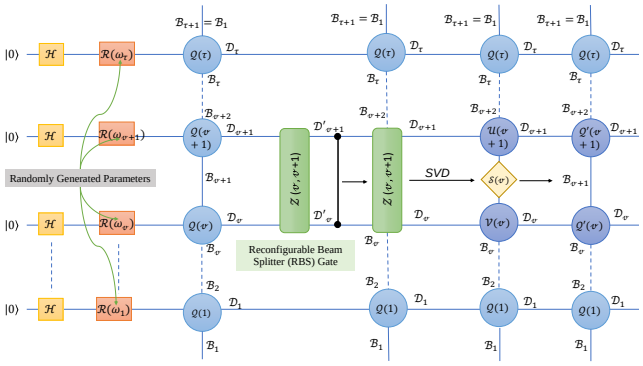

Figure 3: The initial tensor $\mathcal{Q}(\upsilon)$ is contracted with the unitary rotation matrix $\mathcal{R}_y$. The common bond index is contracted between $\mathcal{Q}(\upsilon)$ and $\mathcal{Q}(\upsilon + 1)$ to apply a two-qubit controlled rotation operation. SVD is applied on the resultant tensor $\mathcal{Z}'$ after reshaping it as $(\mathcal{I}'_\upsilon \times \mathcal{L}_\upsilon) \times (\mathcal{I}'_{\upsilon+1} \times \mathcal{L}_{\upsilon+1})$.

The single-qubit rotation gate, $\mathcal{R}_y(\omega)$, encodes rotations along the Y axes in the encoding section. Initially, all the input Quantum bits (qubits) of TR-HQG are in the reset state ($|0\rangle$) and these reset qubits are passed through Hadamard gates ($\mathcal{H}$) to provide an equal superposition as $\mathcal{H}|0\rangle^n = \frac{1}{\sqrt{2^n}}(|0\rangle + |1\rangle)^{\otimes n}$. In TR-HQG, the single-qubit states $|\psi(\omega)\rangle$ correspond to the rotation of this reference state by the TR-generated angle ω as, $|\psi(\omega)\rangle = \cos\omega|0\rangle + \sin\omega|1\rangle$, which is normalized for every ω. It should be stressed that the reference state $\mathcal{H}|0\rangle^n$ and the product of rotated states $\bigotimes_\upsilon |\psi(\omega_\upsilon)\rangle$ are superpositions over all Hamming-weight sectors. HWP therefore refers to the entangling layer: every RBS gate is block-diagonal with a 2×2 rotation on the $\{|01\rangle, |10\rangle\}$ subspace and acts as the identity on $\{|00\rangle, |11\rangle\}$, so the circuit acts independently on each fixed-Hamming-weight subspace and leaves the Hamming-weight distribution of the input product state unchanged. Consequently all 2^n computational-basis outcomes remain reachable, while the dynamics inside each weight-k sector are confined to a $\binom{n}{k}$-dimensional invariant subspace, which is what makes the circuit classically tractable to simulate. To encode phase information, the quantum layer of TR-HQG uses the rotation gates $\mathcal{R}_y$ and $\mathcal{R}_z$. Complementary quantum states are created with the help of the RBS gate. In the Bloch sphere projection, the $\mathcal{R}_y(\omega)$ and $\mathcal{R}_z(\omega)$ gates represent the following single-qubit rotations about the Y and Z-axes, respectively, as follows:

$$\mathcal{R}_y(\omega) = \begin{bmatrix} \cos\omega/2 & -\sin\omega/2 \\ \sin\omega/2 & \cos\omega/2 \end{bmatrix}, \ and, \ \mathcal{R}_z(\omega) = \begin{bmatrix} \exp(-i\omega/2) & 0 \\ 0 & \exp(i\omega/2) \end{bmatrix} . \tag{6}$$

The initial tensor $\mathcal{X}(\upsilon)$ is contracted with the 2×2 unitary rotation matrix $\mathcal{R}_y$ to accomplish the one-qubit rotation. The rotated state of the υth qubit is represented by the resultant tensor $\mathcal{Q}(\upsilon)$ as follows:

$$\mathcal{Q}(\upsilon)^{y_\upsilon}_{x_\upsilon x_{\upsilon+1}} = \sum_{y'_\upsilon \in \{0,1\}} [\mathcal{R}_y]_{y_\upsilon y'_\upsilon} \mathcal{X}(\upsilon)^{y'_\upsilon}_{x_\upsilon x_{\upsilon+1}} \ . \tag{7}$$

Before conducting the RBS gate operations, the TR must be converted into an orthogonal form centered around the qubits of interest, υ and $(\upsilon+1)$ in the proposed TR-HQG. To build a new tensor, the common bond index is contracted between the tensor cores $\mathcal{Q}(\upsilon)$ and $\mathcal{Q}(\upsilon+1)$ as follows:

$$\mathcal{Z}(\upsilon,\upsilon+1)^{y_\upsilon y_{\upsilon+1}}_{x_\upsilon x_{\upsilon+2}} = \sum_{x_{\upsilon+1}=1}^{\mathcal{L}_{\upsilon+1}} \mathcal{Q}(\upsilon)^{y_\upsilon}_{x_\upsilon x_{\upsilon+1}} \mathcal{Q}(\upsilon+1)^{y_{\upsilon+1}}_{x_{\upsilon+1} x_{\upsilon+2}} \tag{8}$$

where $\mathcal{Z}$ is the contracted tensors of $\mathcal{Q}(\upsilon)$ and $\mathcal{Q}(\upsilon+1)$.
We restructure the two-qubit RBS into an operator acting on the joint state of qubits υ and $(\upsilon+1)$ to apply it to the two-qubit tensor derived from Eq. 8.

$$\mathcal{Z}'(\upsilon,\upsilon+1)^{y_\upsilon y_{\upsilon+1}}_{x_\upsilon x_{\upsilon+2}} = \sum_{y_\upsilon y_{\upsilon+1}} RBS(\upsilon,\upsilon+1)^{y'_\upsilon y'_{\upsilon+1}}_{y_\upsilon y_{\upsilon+1}} \mathcal{Z}(\upsilon,\upsilon+1)^{y_\upsilon y_{\upsilon+1}}_{x_\upsilon x_{\upsilon+2}} \, . \tag{9}$$

We apply SVD on the resulting tensor $\mathcal{Z}'$ after reshaping it as $(y'_\upsilon \times x_\upsilon) \times (y'_{\upsilon+1} \times x_{\upsilon+1})$ in the following manner:

$$\mathcal{Z}'(\upsilon,\upsilon+1)^{y_\upsilon y_{\upsilon+1}}_{x_\upsilon x_{\upsilon+2}} = \sum_{x_{\upsilon+1}} \mathcal{V}(\upsilon)^{y'_\upsilon}_{x_\upsilon x_{\upsilon+1}} \mathcal{S}_{x_\upsilon} \mathcal{M}(\upsilon+1)^{y'_{\upsilon+1}}_{x_{\upsilon+1} x_{\upsilon+2}} \, . \tag{10}$$

In this context, $\mathcal{V}$ and $\mathcal{M}$ consist of orthogonal vectors, while $\mathcal{S}_{x_\upsilon}$ is formed by the singular values of matrix $\mathcal{Z}'$. The matrix always has a total of $2\mathcal{B}$ singular values, regardless of the structure of the two-qubit gate. Here, $\mathcal{B}$ represents the bond dimension of the TR. After truncating the matrix $\mathcal{S}_{x_\upsilon}$ to retain only the $\mathcal{N}$ largest singular values, we refer to the resulting matrix as $\mathcal{S'}_{x_\upsilon}$. $\mathcal{V}$ and $\mathcal{M}$ are truncated to retain only the orthogonal vectors associated with the $\mathcal{N}_s$ largest singular values and resultant matrices, $\mathcal{Q}'(\upsilon)$ and $\mathcal{Q}'(\upsilon+1)$ are evaluated as follows.

$$\mathcal{Q}'(\upsilon)^{y_\upsilon}_{x_\upsilon x_{\upsilon+1}} = \mathcal{V}^{y_\upsilon}_{x_\upsilon x_{\upsilon+1}} \mathcal{S'}_{x_\upsilon}, \mathcal{Q}'(\upsilon+1)^{y_{\upsilon+1}}_{x_{\upsilon+1} x_{\upsilon+2}} = \mathcal{M}^{y_{\upsilon+1}}_{x_{\upsilon+1} x_{\upsilon+2}} \, . \tag{11}$$

The TR-approximation procedures are illustrated in Fig. 3 and provided in Algorithm 1 (*Appendix*).
Role of the Classical Tensor Networks: Two classical tensor networks appear in our Quan-CEFT and play different roles (Fig. 2). The TR cores $\mathcal{Q}(\upsilon)$ sit upstream of the circuit. The TTN mapping network sits downstream of the circuit and expands the measured context vector into the adapter update. No classical data is ever encoded into a quantum state: the circuit always starts from the fixed reference state $\mathcal{H}|0\rangle^n$, and $|\psi(\omega^\upsilon)\rangle$ denotes this reference state after the rotations with randomly-generated angles ω_υ have been applied, i.e. $|\psi(\omega^\upsilon)\rangle = \bigotimes_{\upsilon=1}^{n} \mathcal{R}_y(\omega^\upsilon)\mathcal{H}|0\rangle$. Subsequently, the state undergoes processing through TR-HQG with the RBS layers $\mathcal{U}(\omega_1,\omega_2,\cdots,\omega_n)$. Finally, by measuring all qubits in the Pauli-Z basis, we obtain the 2^n computational-basis outcomes λ_j together with their probabilities:

$$\mathcal{E}_{ij} = \langle\psi(\omega^\upsilon)|\mathcal{U}^\dagger(\omega)|\lambda_j\rangle\langle\lambda_j|\mathcal{U}(\omega)|\psi(\omega^\upsilon)\rangle, \qquad j = 0,\ldots,2^n-1, \tag{12}$$

where the complete operation $\mathcal{U}(\omega)$ is defined as

$$\mathcal{U}(\omega) = \prod_{i=1} \mathcal{U}_i(\omega_i) = \mathcal{U}_n(\omega_n)\mathcal{U}_{n-1}(\omega_{n-1})\cdots\mathcal{U}_1(\omega_1) \, . \tag{13}$$

In the TR simulation, $\mathcal{E}_{ij}$ is evaluated by contracting the ring of cores $\mathcal{Q}'(\upsilon)$ obtained after the SVD sweep with its complex conjugate along the physical indices fixed to the bit-string λ_j; the probabilities of the 2^n outcomes are read out in chunks so that the cost stays $\mathcal{O}(n\mathcal{B}^3)$ per chunk.
From Context Vector to Adapter Update: Let $\mathbf{h} \in \mathbb{R}^{2^n}$ be the context vector with entries $\mathcal{E}_{ij}$ (Fig. 2, Context Vector Generation), and let the target adapter for the ℓ-th adapted projection $\ell \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ be the rank-r pair (A_ℓ, B_ℓ) with $m_\ell = r(d_{\text{in}} + d_{\text{out}})$ entries. The 2^n basis outcomes are partitioned into $\lceil m_\ell / n_{\text{MLP}} \rceil$ chunks (n_{MLP} is the mapping chunk size) and each outcome probability is expanded by the shared TTN mapping $\mathcal{W}_{\text{TT}}$ into n_{MLP} target entries:

$$\text{vec}([A_\ell;\, B_\ell]) = \Pi_\ell \, \mathcal{W}_{\text{TT}}(\mathbf{h}) \, , \qquad \Delta\mathcal{W}_\ell = B_\ell A_\ell \, , \qquad \mathcal{W}_\ell^{\text{merged}} = \mathcal{W}_\ell + \Delta\mathcal{W}_\ell \, , \tag{14}$$

where $\mathcal{W}_{\text{TT}} : \mathbb{R}^{2^n} \to \mathbb{R}^{2^{n_{\text{MLP}}}}$ is the tensor-train factorized linear map (its cores are the second block of trainable parameters) and Π_ℓ selects and reshapes the first m_ℓ generated entries for projection ℓ. Since, a single generator is shared across all adapted layers, the circuit is evaluated once per optimization step and only $\Pi_\ell \mathcal{W}_{\text{TT}}$ is applied per layer. After training, $\Delta\mathcal{W}_\ell$ is merged into the frozen backbone, so inference is identical to the pre-trained model.
Training Objective: The fine-tuning objective is the standard auto-regressive cross-entropy of the adapted backbone over the vocabulary $\mathcal{V}$; with $\Theta = \{\mathcal{Q}(\upsilon)\}_{\upsilon=1}^{n} \cup \{\mathcal{W}_{\text{TT}}\}$ denoting all trainable tensors,

$$\mathcal{L}_{\mathrm{LM}}(\Theta) = -\frac{1}{T}\sum_{t=1}^{T} \log p(z_t \mid z_{<t};\, \{\mathcal{W}_\ell + \Delta\mathcal{W}_\ell(\Theta)\}_\ell)\,, \tag{15}$$

where $z_{1:T}$ is a training block. Gradients of $\mathcal{L}_{\mathrm{LM}}$ flow by ordinary back-propagation through $\Delta\mathcal{W}_\ell$, the TTN mapping, the TR contractions that implement the circuit, and finally into the TR cores; no parameter of the circuit itself is updated. All experiments in this paper optimize Θ jointly with AdamW on $\mathcal{L}_{\mathrm{LM}}$. The end-to-end procedure is: (i) approximate the qubits as TR cores comprising with randomly generated angles ω^v; (ii) apply $\mathcal{R}_y$ and the SVD sweep to the reference state (Eqs. 8–11) and read out $\mathbf{h}$ (Eq. 12); (iii) expand $\mathbf{h}$ into $\{\Delta\mathcal{W}_\ell\}$ (Eq. 14); (iv) run the adapted backbone on a mini-batch and evaluate Eq. 15; (v) back-propagate to Θ and take an *AdamW* step. The projector-based measurement loss and its analytical gradient with respect to the generated angles, which we use as a diagnostic of the circuit landscape (Appendix A.4), are given in Appendix A.3: there, $\theta_j = |\lambda_j\rangle\langle\lambda_j|$ is the projector onto outcome λ_j and t_j a target value. Additional details including a training loop (Algorithm 2) are elaborated in Subsection A.2 in *Appendix*.

5 Results

5.1 Dataset

We evaluate our Quan-CEFT and three parameter-efficient adaptation baselines: LoRA (Hu et al., 2022), QPA (Liu et al., 2025), and Quantum-PEFT (Koike-Akino et al., 2025), on causal language modeling with WikiText-2 (Merity et al., 2017). The processed corpus contains 9,327 training blocks, 964 validation blocks, and 1,104 test blocks (block size=256). We use a maximum sequence length of 1024 tokens and apply identical preprocessing, optimization, and evaluation procedures to all methods, so that differences in outcome can be attributed to the adapter parameterization rather than to data or protocol-level factors.

5.2 Experimental Settings

All methods use adapter rank r=4 (LoRA scaling α=2r, no dropout) and are optimized with *AdamW* using a batch size of 2 on CPU. For GPT-2, adaptation is injected into the fused c_attn projection of each Transformer layer; for Pythia $query_key_value$ and $dense$; for OPT and TinyLlama q_proj, k_proj, v_proj, and the output projection. These are the attention projections that LoRA adapts by default for each architecture, so every method is compared under the same target module configuration for a given backbone. The key design choice is that the TTN-MLP's hidden dimensions $(128, 256, 256, 128)$ with TT-rank=8, chunk size= 1024, AdamW learning rate= $1e-3$, and TR rank $\mathcal{B} = 4$.

Evaluation Protocol: Perplexity is computed as the exponentiated mean token-level cross-entropy over the 1,104 test blocks. For ROUGE-L, each test block of 1024 tokens is split into a prompt prefix and a held-out reference continuation; the adapted model generates a continuation of the same length as the reference from the prefix, and ROUGE-L is computed between the generated and reference continuations and averaged over the evaluated blocks.

Quan-CEFT employs a simulated Hamming weight preserving quantum generator with a Tensor-Ring (TR) parameterization, a periodic (ring-trace) angle activation, and a single generator shared across all adapted layers. The number of simulated qubits follows the model and projection dimensions: 8 for GPT-2 and Pythia, 9 for OPT, and 11 for TinyLlama. For GPT-2, the generator contains 192 TR angle parameters and a Tensor Train (TT) mapping network with 18,056 parameters, giving 18,248 trainable parameters in total. The quantum circuit itself contributes no trainable parameters, since it acts only as a fixed forward-pass evaluator. QPA combines a dense-statevector circuit with a dense mapping network ($\approx$ 105K parameters on every backbone); Quantum-PEFT use a Pauli parameterization whose parameter count grows only logarithmically with the number and dimension of adapted modules; LoRA scales directly with the dimensions and number of adapted projections.

5.3 Experimental Results

Scaling of the Parameter Budget: Tab. 2 reports the trainable parameter budget of Quan-CEFT and LoRA for six target combinations. Since, the generator produces the entire adaptation tensor from a fixed $\approx$ 18K-parameter TR-HQG and TTN mapping, our Quan-CEFT budget is essentially flat (18,248–18,632) regardless of how many modules are adapted, whereas the LoRA budget grows with both the number of adapters and the hidden dimension. The reduction factor therefore increases with scale, from 8.1$\times$ (GPT-2, c_attn) to 77.2$\times$ (Gemma-2-2B, all_attn).

Overall Comparison: Tab. 3 summarizes parameter counts, test-split perplexity after adaptation,

Table 2: Trainable-parameter budget of Quan-CEFT versus LoRA (r=4) for different adaptation targets. lm_head adapts the output projection only; c_attn adapts the fused attention projection in every layer; all_attn adapts all attention projections. Reduction is relative to LoRA.

Model	Target	Adapters	LoRA	Quan-CEFT	Qubits	Reduction
GPT-2	lm_head	1	204,100	18,248	8	11.2×
GPT-2	c_attn	12	147,456	18, 248	8	8.1×
GPT-2	all_attn	24	221,184	18,248	8	12.1×
Gemma-2-2B	lm_head	1	1,033,216	18,504	10	55.8×
Gemma-2-2B	c_attn	26	958, 464	18,504	10	51.8×
Gemma-2-2B	all_attn	52	1,437,696	18,632	11	77.2×

and ROUGE-L. Quan-CEFT is the most parameter-efficient method in every setting. Its trainable parameter count varies by only 2.1% across the four backbones (18,248–18,632), while LoRA increases from 147,456 parameters for GPT-2 to 1,126,400 for TinyLlama. As a result, the compression factor of Quan-CEFT increases with model scale: 8.1× on GPT-2, 12.1× on Pythia, 16.1× on OPT, and 60.5× on TinyLlama. QPA also exhibits a comparatively stable parameter count, ranging

Table 3: CPU evaluation on WikiText-2 (test split). All methods use rank-4 adapters, AdamW, batch size 2 and five optimization steps. PPL denotes test perplexity after adaptation; ROUGE-L is measured on generated continuations. The reduction column is relative to LoRA for the same backbone (Base PPL for GPT-2 (124M): 25.18, OPT-125M: 26.09, Pythia-160M: 25.60, TinyLlama-1.1B: 7.57)

Model	Method	Qubits	Train. Params.(↓)	Reduction(↑)	PPL(↓)	ROUGE-L(↑)	Peak-Mem (MB)↓	Latency (ms/step)↓
GPT-2 (124M)	LoRA	–	147, 456	1.0×	**25.78**	**0.2502**	2213.0	2280
	QPA	8	105, 208	1.4×	26.46	0.1649	1080.7	1700
	Q-PEFT	8	36, 864	4.0×	26.83	0.1441	2520.9	22500
	Quan-CEFT	8	**18,248**	**8.1**×	26.21	0.2348	**1093.2**	**1420**
OPT-125M	LoRA	–	294, 912	1.0×	**26.16**	0.1806	1809.5	2220
	QPA	8	105, 248	2.8×	28.71	0.1710	1636.2	1520
	Q-PEFT	8	49,152	6.0×	28.93	0.1837	2147.4	18920
	Quan-CEFT	8	**18,376**	**16.1**×	26.73	**0.1998**	1678.3	**1440**
Pythia-160M	LoRA	–	221, 184	1.0×	**25.14**	0.1143	1796.6	2540
	QPA	9	105, 208	2.1×	25.96	0.1119	**1077.1**	1800
	Q-PEFT	9	49, 152	4.5×	26.15	0.1060	2399.4	27520
	Quan-CEFT	9	**18,248**	**12.1**×	25.37	**0.1694**	1080.8	1680
TinyLlama-1.1B	LoRA	–	1, 126, 400	1.0×	7.88	**0.1976**	6381.0	16560
	QPA	11	105, 328	10.7×	8.25	0.1724	5958.9	14360
	Q-PEFT	11	240, 416	4.7×	8.79	0.1680	8495.9	166460
	Quan-CEFT	11	**18,632**	**60.5**×	**7.46**	0.1968	**5890.0**	**10200**

from 105,208 to 105,328 parameters. However, its dense mapping network is substantially larger than the tensor-factorized mapping used in Quan-CEFT. Quantum-PEFT are more compact than LoRA, however, their parameter count increases from 36,864 on GPT-2 to 240,416 on TinyLlama. Consequently, Quan-CEFT is 2.0×–13.0× smaller than Quantum-PEFT and 5.7×–5.8× smaller than QPA across the tested models with the lowest per-step CPU training latency of all four methods on every backbone: 5–29% (mean ≈ 14%) lower than QPA, 34–38% lower than LoRA, and 13–16× lower than Quantum-PEFT, under a short, five-step CPU adaptation budget.

Text Generation Quality: Despite its substantially smaller parameter budget, Quan-CEFT achieves the highest ROUGE-L score on two of the four backbones, as reported in Tab. 3. It improves over LoRA by 0.0551 on Pythia (0.1694 vs. 0.1143) and by 0.0192 on OPT (0.1998 vs.0.1806). On TinyLlama, Quan-CEFT nearly matches LoRA (0.1968 vs.0.1976), with a gap of only 0.0008, while using 60.5× fewer trainable parameters. On GPT-2, Quan-CEFT scores 0.2348 compared with LoRA's 0.2502. These results suggest that Quan-CEFT can maintain competitive generation quality under a strict parameter budget within the five-step training horizon. The ROUGE-L and perplexity rankings also differ: Quan-CEFT achieves higher ROUGE-L despite slightly higher perplexity on Pythia and OPT, whereas it achieves lower perplexity on TinyLlama (7.46 vs. 7.88) with nearly identical ROUGE-L. Qualitative text generations are provided in Tab. 9 in the *Appendix*.

Rank Ablation Study: On Pythia-160M with a block size of 256, Quan-CEFT exhibits increasing post-adaptation perplexity as the adapter rank grows from 1 to 32, as reported in Tab. 4. Rank $r = 1$ achieves the lowest PPL among the tested settings, reducing it from the unadapted baseline of 25.60 to 24.78, a 3.2% improvement. Rank r=4 yields a smaller reduction to 25.37, corresponding to a 0.9% improvement. In contrast, ranks $r = 16$ and $r = 32$ increase PPL above the baseline to 26.22 and 28.19, respectively. Meanwhile, the trainable parameter count grows modestly, from 17,992 at r=1 to 18,632 at r=32. Thus, increasing rank offers no perplexity benefit under the evaluated training configuration. Rank

Table 4: Rank ablation study for Quan-CEFT on Pythia-160M (Base PPL: 25.60) with a block size of 256

Rank	Trainable Parameters	Post PPL
1	17, 992	**24.78**
4	18, 248	25.37
16	18, 504	26.22
32	18, 632	28.19

$r = 1$ provides both the lowest perplexity and the smallest trainable parameter count in this sweep, while r=4 retains a modest improvement over the baseline. Additional experimental results: Poly-logarithmic Qubit-Scaling (Fig. 7), TR-HQG circuit health analysis (Tab. 8), and fidelity of the TR simulation (Fig. 8) are provided in Subsec A.5.

6 DISCUSSIONS

In our proposed Quan-CEFT framework, the quantum circuit serves solely as a fixed forward-pass evaluator, executing with parameters provided by the TR network to compute the objective function value. This meta-parameterization approach fundamentally transforms the VQC optimization landscape, offering clear benefits for scalability and robustness in parameter-efficient fine-tuning. By restricting the optimization process to the classical TR network, the proposed TR-HQG is designed to mitigate the barren plateau problem: the trainable parameters are the TR cores, and by the chain rule their gradient is the circuit-angle gradient multiplied by the Jacobian of the TR contraction, so a well-conditioned TR mapping and the restricted HWP dynamics of the RBS layers are both required for trainability. Appendix A.4 gives the variance analysis together with empirical gradient variance measurements, which show that beyond $\mathcal{N} = 6$ qubits the variance no longer decays at an exponential rate for the moderate depths used here. Since the trainable parameters are entirely within the classical domain, gradient updates are shielded from the randomness of quantum measurements, and exact simulation removes shot noise altogether.

The experimental results suggest a favorable parameter vs. latency vs. quality trade-off for the proposed Quan-CEFT. LoRA provides the strongest quality metrics in the current short-run experiment but requires substantially more trainable parameters. QPA introduces greater circuit-simulation and mapping overhead, whereas Quantum-PEFT achieve parameter compression at the expense of substantially increased CPU latency. Quan-CEFT, in contrast, combines the smallest trainable parameter count ($18{,}248$–$18{,}632$) with the lowest per-step CPU latency on every backbone (Table 3), whose compound-matrix evaluation dominates its runtime. Its peak training memory is also below LoRA on all four backbones and the lowest of all methods on GPT-2 and TinyLlama. Quan-CEFT maintains language-modeling quality close to LoRA under the evaluated training budget.

Quantum-PEFT preserve near-initial perplexity on GPT-2, Pythia, and OPT, consistent with their near-identity multiplicative construction. However, CPU-based generation evaluation timed out for all models, and the TinyLlama perplexity evaluation also timed out because of the compound-matrix computation over many high-dimensional modules. These outcomes underline that parameter count alone does not fully characterize practical efficiency: the computational structure of an adapter can become a bottleneck for large models. Across four architectures and model sizes, Quan-CEFT consistently provides the smallest trainable parameter budget while maintaining competitive short-budget adaptation quality. The method improves ROUGE-L over LoRA on three backbones while using $12.1\times$–$60.5\times$ fewer parameters in those settings. LoRA remains stronger in short-horizon perplexity reduction, whereas QPA and Quantum-PEFT show unfavorable initialization behavior or CPU scalability limitations. Taken together, these results support Quan-CEFT as a promising parameter-efficient alternative for adapting larger language models, while motivating longer-budget and accelerator-based experiments to assess final convergence and wall-clock efficiency.

7 CONCLUSION

In this research, we propose a novel Quan-CEFT framework with the proposed TR-HQG circuit, achieving competitive performance with the fewest trainable parameters on WikiText-2 causal language modeling as a proof-of-concept. Our Quan-CEFT offers several advantages: 1) our TR-HQG is free from classical-to-quantum data encoding and hence incurs less computational overhead; 2) it maintains the entire training process within the classical TR and TTN-based MLP model; 3) it eliminates any reliance on quantum computing hardware during inference, as the generated update is merged into the classical backbone; and 4) it significantly reduces the number of trainable parameters by mapping quantum measurement bases to the target neural network parameters, with a qubit count that grows logarithmically in the number of generated parameters.

However, the present evaluation is limited to five optimization steps on CPU, to LoRA and quantum-inspired baselines, and to text-only benchmarks. It remains to investigate the converged behavior of Quan-CEFT on Vision models, and the fine-tuning and inference efficiency of TR simulations on resource-constrained devices.

AI USE STATEMENT

In this work, we used generative AI tools solely for language editing to improve the grammar, clarity, and readability of this manuscript. The authors reviewed and verified all edits and take full responsibility for the manuscript's content.

REPRODUCIBILITY STATEMENT

To support reproducibility, Section 4 describes the Quan-CEFT formulation, adapter construction, and training objective, while Appendix A.2 provides implementation details and pseudocode for the generator and end-to-end training procedure (Algorithms 1 and 2). Sections 5.1-5.2 document the dataset, adapter configurations, optimization settings, and evaluation protocols, and Appendix A.5 specifies the five-step adaptation budget and exact statevector simulation used for the reported benchmarks. Appendices A.3-A.4 present the analytical gradient derivation and gradient-variance analysis, while Appendices A.5.1-A.5.2 report generator diagnostics and validation of the Tensor Ring simulator against exact simulation. The anonymous PyTorch implementation of our Quan-CEFT is available in `https://anonymous.4open.science/r/QUAN-CEFT/`.

REFERENCES

Aastik, Meghana Topu, Chinmay Kulkarni, and Pragya Paramita Sahu. NormAL LoRA: What is the perfect size? In *Findings of the Association for Computational Linguistics (EMNLP 2025)*, pp. 19716–19731, Suzhou, China, November 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.findings-emnlp.1074. URL `https://aclanthology.org/2025.findings-emnlp.1074/`.

Vaneet Aggarwal, Wenlin Wang, Brian Eriksson, Yifan Sun, and Wenqi Wang. Wide compression: Tensor ring nets, 2018. URL `https://ieeexplore.ieee.org/document/8579070`.

J. Biamonte, P. Wittek, N. Pancotti, et al. Quantum machine learning. *Nature*, 549:195–202, 2017. doi: https://doi.org/10.1038/nature23474.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, US Prashanth, Edward Raff, et al. Pythia: A suite for analyzing large language models across training and scaling. In *Proc. of the 40th International Conference on Machine Learning (ICML 2023)*, pp. 2397–2430. PMLR, 2023.

M.C. Caro, HY. Huang, M. Cerezo, et al. Generalization in quantum machine learning from few training data. *Nat. Commun.*, 13(4919), 2022. doi: https://doi.org/10.1038/s41467-022-32550-3.

M. Cerezo and HYt Verdon, G.and Huang. Challenges and opportunities in quantum machine learning. *Nat. Comput. Sci.*, 2:567–576, 2022. doi: https://doi.org/10.1038/s43588-022-00311-3.

Enrique Cervero Martín, Kirill Plekhanov, and Michael Lubasch. Barren plateaus in quantum tensor network optimization. *Quantum*, 7:974, 2023. doi: 10.22331/q-2023-04-13-974.

El Amine Cherrat, Snehal Raj, Iordanis Kerenidis, et al. Quantum Deep Hedging. *Quantum*, 7: 1191, 2023. ISSN 2521-327X.

I. Cong, S. Choi, and M.D. Lukin. Quantum convolutional neural networks. *Nat. Phys.*, 15:1273–1278, 2019. doi: https://doi.org/10.1038/s41567-019-0648-8.

Luciano S. de Souza, Jonathan H. A. de Carvalho, and Tiago A. E. Ferreira. Classical artificial neural network training using quantum walks as a search procedure. *IEEE Transactions on Computers*, 71(2):378–389, 2022. doi: 10.1109/TC.2021.3051559.

Ali Edalati, Marzieh S. Tahaei, Ivan Kobyzev, Vahid Partovi Nia, James J. Clark, and Mehdi Rezagholizadeh. Krona: Parameter efficient tuning with kronecker adapter. *arXiv*, 2022. URL `https://doi.org/10.48550/arXiv.2212.10650`.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, et al. Parameter-efficient transfer learning for nlp. In *Proc. of the 36 th International Conference on Machine Learning (ICML), Long Beach, California, 2019*, 2019. URL `https://arxiv.org/abs/1902.00751`.

Edward J Hu, Yelong Shen, Phillip Wallis, et al. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Iordanis Kerenidis and Anupam Prakash. Quantum machine learning with subspace states, 2022. URL https://arxiv.org/abs/2202.00054.

Toshiaki Koike-Akino, Francesco Tonin, Yongtao Wu, Frank Zhengqing Wu, Leyla Naz Candogan, and Volkan Cevher. Quantum-peft: Ultra parameter-efficient fine-tuning. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 31581–31602, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/4e52bbb99690d1e05c7ef7b4c8b3569a-Paper-Conference.pdf.

D. Konar, D. Peddireddy, B.K. Panigrahi, and V. Aggarwal. Tensor ring optimized quantum-enhanced tensor neural networks. *Quantum Mach. Intell.*, 7(57), 2025. doi: https://doi.org/10.1007/s42484-025-00281-5.

Dawid Jan Kopiczko, Tijmen Blankevoort, and Yuki M. Asano. Vera: Vector-based random matrix adaptation. In *International Conference on Learning Representations*, 2024. URL https://doi.org/10.48550/arXiv.2310.11454.

M. Larocca, S. Thanasilp, S. Wang, et al. Barren plateaus in variational quantum computing. *Nat. Rev. Phys.*, 7:174–189, 2025. doi: https://doi.org/10.1038/s42254-025-00813-9.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *The 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2021. URL https://arxiv.org/abs/2104.08691.

Xiang Lisa Li and Liang Percy. Prefix-tuning: Optimizing continuous prompts for generation. *Proc. of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, 2021. doi: 10.18653/v1/2021.acl-long.353.

Zhaojiang Lin, Andrea Madotto, and Pascale Fung. Exploring versatile generative language model via parameter-efficient transfer learning. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 441–459. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.findings-emnlp.41.

Chen-Yu Liu, En-Jui Kuo, Chu-Hsuan Abraham Lin, et al. Training classical neural networks by quantum machine learning. In *2024 IEEE International Conference on Quantum Computing and Engineering (QCE)*, volume 02, pp. 34–38, 2024a. doi: 10.1109/QCE60285.2024.10248.

Chen-Yu Liu, Chao-Han Huck Yang, Hsi-Sheng Goan, and Min-Hsiu Hsieh. A quantum circuit-based compression perspective for parameter-efficient learning. In *Proc. of the 13th International Conference on Learning Representations (ICLR)*, 2025. doi: https://doi.org/10.48550/arXiv.2410.09846.

Haokun Liu, Derek Tam, Mohammed Muqeeth, et al. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 1950–1965, 2022.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, et al. DoRA: weight-decomposed low-rank adaptation. In *Proc. of the 41st International Conference on Machine Learning*, 2024b.

Weiyang Liu, Zeju Qiu, Yao Feng, Yuliang Xiu, Yuxuan Xue, Longhui Yu, Haiwen Feng, Zhen Liu, Juyeon Heo, Songyou Peng, Yandong Wen, Michael J. Black, Adrian Weller, and Bernhard Schölkopf. Parameter-efficient orthogonal finetuning via butterfly factorization. In *The Twelfth International Conference on Learning Representations*, 2024c. doi: https://openreview.net/forum?id=7NzgkEdGyr.

S. Mangini, F. Tacchino, D. Gerace, D. Bajoni, and C. Macchiavello. Quantum computing models for artificial neural networks. *Europhysics Letters*, 134(1):10002, 2021. doi: https://doi.org/10.1209/0295-5075/134/10002.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *Proc. International Conference on Learning Representations (ICLR 2017)*, pp. 1–15, 2017. URL https://arxiv.org/abs/1609.07843.

I. V. Oseledets. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317, 2011. doi: https://doi.org/10.1137/090752286.

Dheeraj Peddireddy, Utkarsh Priyam, and Vaneet Aggarwal. Noisy tensor-ring approximation for computing gradients of a variational quantum eigensolver for combinatorial optimization. *Phys. Rev. A*, 108:042429, 2023. doi: 10.1103/PhysRevA.108.042429.

Adrián Pérez-Salinas, Alba Cervera-Lierta, Elies Gil-Fuster, and José I. Latorre. Data re-uploading for a universal quantum classifier. *Quantum*, 4:226, 2020. ISSN 2521-327X. doi: 10.22331/q-2020-02-06-226. URL https://doi.org/10.22331/q-2020-02-06-226.

Yuning Qiu, Guoxu Zhou, Qibin Zhao, and Shengli Xie. Noisy tensor completion via low-rank tensor ring. *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–15, 2022. doi: 10.1109/TNNLS.2022.3181378.

Zeju Qiu, Weiyang Liu, Haiwen Feng, et al. Controlling text-to-image diffusion by orthogonal finetuning. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, 2023. URL 10.5555/3666122.3669594.

Alec Radford, Jeffrey Wu, Rewon Child, et al. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9, 2019.

Snehal Raj and Brian Coyle. Hyper compressed fine-tuning of large foundation models with quantum inspired adapters, 2025. URL https://arxiv.org/abs/2502.06916.

Maria Schuld and Nathan Killoran. Quantum machine learning in feature hilbert spaces. *Phys. Rev. Lett.*, 122:040504, 2019. doi: https://link.aps.org/doi/10.1103/PhysRevLett.122.040504.

Maria Schuld, Ryan Sweke, and Johannes Jakob Meyer. Effect of data encoding on the expressive power of variational quantum-machine-learning models. *Phys. Rev. A*, 103:032430, 2021. doi: https://link.aps.org/doi/10.1103/PhysRevA.103.032430.

Dechun Song, Peiyong Zhang, and Feiteng Li. Speeding up deep convolutional neural networks based on tucker-cp decomposition. In *Proceedings of the 2020 5th International Conference on Machine Learning Technologies*, pp. 56–61, New York, NY, USA, 2020. doi: 10.1145/3409073.3409094.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, et al. Llama: Open and efficient foundation language models, 2023.

Steven R. White. Density matrix renormalization group algorithms with a single center site. *Phys. Rev. B*, 72:180403(R), Nov 2005. doi: 10.1103/PhysRevB.72.180403. URL https://link.aps.org/doi/10.1103/PhysRevB.72.180403.

Yifan Yang, Jiajun Zhou, Ngai Wong, and Zheng Zhang. LoRETTA: Low-rank economic tensor-train adaptation for ultra-low-parameter fine-tuning of large language models. In *Proc. of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 3161–3176, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.174. URL https://aclanthology.org/2024.naacl-long.174/.

Jinshi Yu, Guoxu Zhou, Chao Li, Qibin Zhao, and Shengli Xie. Low tensor-ring rank completion by parallel matrix factorization. *IEEE Transactions on Neural Networks and Learning Systems*, 32(7):3020–3033, 2021. doi: 10.1109/TNNLS.2020.3009210.

Longhao Yuan, Chao Li, Danilo Mandic, Jianting Cao, and Qibin Zhao. Tensor ring decomposition with rank minimization on latent space: an efficient approach for tensor completion. In *AAAI'19/IAAI'19/EAAI'19: Proc. of the Thirty-Third AAAI Conference on AI, Honolulu, Hawaii, USA*, pp. 9151–9158, 2019. doi: https://doi.org/10.1609/aaai.v33i01.33019151.

Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In *60th Annual Meeting of the Association for Computational Linguistics*, 2022. URL https://arxiv.org/abs/2106.10199.

Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. Tinyllama: An open-source small language model. *arXiv*, 2024.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. *arXiv*, 2022. doi: https://doi.org/10.48550/arXiv.2205.01068.

A Appendix

A.1 Introduction to Quantum Machine Learning

Quantum Machine Learning (QML) leverages quantum properties like superposition and entanglement, offering a theoretically promising avenue to accelerate training and learning processes Caro et al. (2022); Cerezo & Verdon (2022); Biamonte et al. (2017); Schuld & Killoran (2019). However, encoding classical data into quantum states increases the depth of quantum circuits, further extending the depth of quantum neural networks (QNNs) Pérez-Salinas et al. (2020); Schuld et al. (2021). Proposals have emerged suggesting the integration of deep learning networks into quantum computing paradigms Cong et al. (2019), where QNN architectures are implemented using Variational Quantum Circuits (VQC). In such cases, input data is encoded into quantum states via amplitude/angle encoding methods, necessitating access to quantum computers during inference. Utilizing quantum computing to train classical Neural Networks (NNs) presents a promising approach where the resulting model remains entirely classical, effectively overcoming challenges related to data encoding and reliance on quantum hardware during inference Mangini et al. (2021); Liu et al. (2024a). One example involves employing quantum walks as a search process to optimize classical NN parameters de Souza et al. (2022). Another method proposes using a Tensor Ring-simulated quantum circuit to train classical Tensor Networks Konar et al. (2025). However, the practical utility of these approaches is limited to training binary NNs, restricting their applicability in broader contexts.

A.2 Implementation Details of Quan-CEFT: Block Diagrams, Per-Stage Cost and Pseudo-Code

This section collects the implementation-level description of Quan-CEFT that complements Sec. 4: (i) the TR-approximation of input qubits and, (ii) the translation of the Hamming-weight preserving circuit specification into the Tensor-Ring operations that are actually executed (Fig. 4), (iii) the end-to-end forward/backward flow (Fig. 5), (iv) the per-stage output and cost of the TR-HQG generator (Tab. 5) together with the per-gate cost of the exact statevector and TR simulations (Tab. 6), and (v) pseudo-code for the TR-approximated generator (Algorithm 1) and for the complete training loop (Algorithm 2). Throughout, the notation follows Sec. 4.1: $\mathcal{Q}(v)$ are the TR cores, $\mathcal{B}$ is the TR bond dimension (rank), $\mathcal{D} = 2$ is the physical dimension, ω are the generated rotation angles, and $\mathcal{Z}$, $\mathcal{Z}'$, $\mathcal{V}$, $\mathcal{S}$, $\mathcal{M}$ are the intermediate tensors of Eqs. 8–11.

Tab. 6 makes explicit the source of the computational advantage claimed in Sec. 1: at $n = 8$ qubits, the setting used for GPT-2 and Pythia, exact simulation is still cheaper per gate than TR simulation at $\mathcal{B} = 16$, and the benefit of the TR simulator lies in its constant per-gate cost, which becomes decisive as n grows ($\approx 4{,}000\times$ cheaper per gate at $n = 24$ for $\mathcal{B} = 16$). This is why the fidelity study of Sec. A.5.2 is performed against the exact statevector at the small qubit counts used in the benchmark, while the TR simulator is the intended execution path for larger generators.

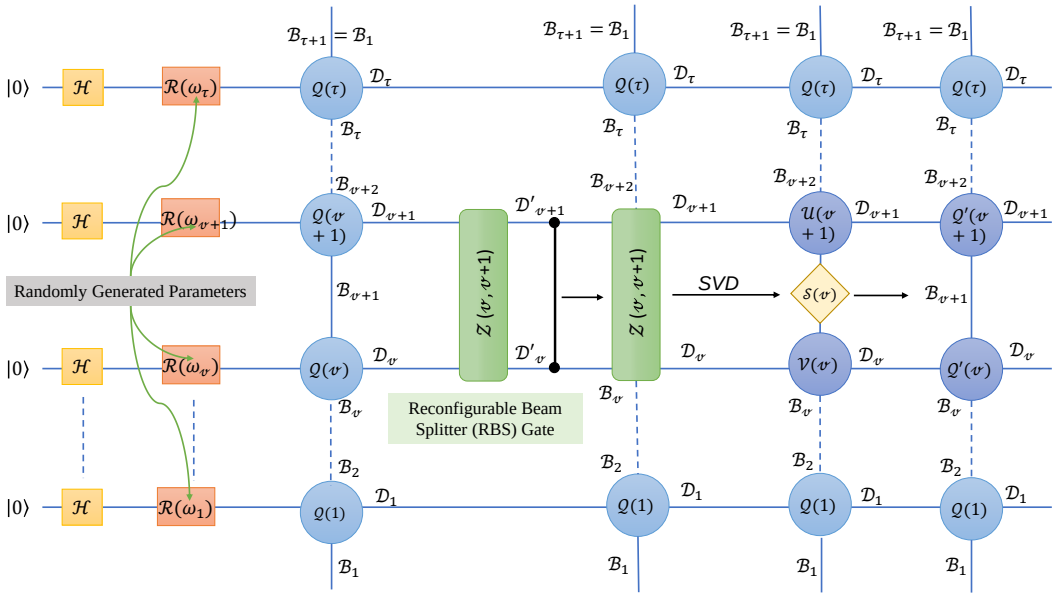

Figure 4: From circuit specification to Tensor-Ring execution. TR-HWP circuit view, i.e. Hadamard and $\mathcal{R}_y$ rotations on every qubit followed by a depth-L ladder of nearest-neighbour RBS gates and Pauli-Z measurement; Quan-CEFT never executes this circuit as a 2^n-dimensional statevector, it only serves as the specification. The Tensor-Ring realization that actually runs, in which each qubit is a core $\mathcal{G}_\upsilon \in \mathbb{R}^{\mathcal{B}\times 2\times \mathcal{B}}$ (denoted $\mathcal{Q}(\upsilon)$ in Sec. 4.1) and every RBS gate is a local operation on two adjacent cores (contract $\rightarrow$ SVD $\rightarrow$ truncate back to bond dimension, $\mathcal{B}$). The ring bond between the last and the first core closes the Tensor Ring.

Table 5: Per-stage implementation of the TR-HQG parameter generator for n qubits, bond dimension $\mathcal{B}$ and mapping chunk size n_{MLP}. Each stage is a classical tensor-network operation; the cost column gives the leading-order arithmetic cost of the stage.

#	Stage	Implementation (classical, CPU)	Output / cost
1	Qubit initialization	Hadamard gate on every reset qubit; no data-encoding stage, hence no input-dependent overhead.	n qubits in equal superposition; $\mathcal{O}(n)$
2	Learnable rotations	Single-qubit $\mathcal{R}_y$ (and $\mathcal{R}_z$) gates whose angles ω are randomly generated; the cores are randomly initialized and learned.	n angles ω; $\mathcal{O}(n)$
3	TR state transform	Each qubit is represented by a core $\mathcal{Q}(\upsilon) \in \mathbb{R}^{\mathcal{B}\times\mathcal{D}\times\mathcal{B}}$, $\mathcal{D}=2$, holding the variational parameters of that qubit.	n cores; $\approx 2n\mathcal{B}^2$ numbers
4	Two-qubit RBS operation	Adjacent cores are contracted (Eq. 8) and the 4×4 RBS unitary is applied to the joint physical indices, a local contraction instead of a 4×4 unitary acting on a 2^n statevector.	rank-4 tensor $\mathcal{Z}'$; $\mathcal{O}(\mathcal{B}^3)$ per gate
5	SVD decomposition	$\mathcal{Z}'$ is reshaped and factorized by SVD (Eq. 10); the $2\mathcal{B}$ singular values are truncated back to $\mathcal{B}$, which caps the growth of the representation.	two cores at $\mathcal{B}$; $\mathcal{O}(\mathcal{B}^3)$
6	Reconstruction	The truncated factors are re-absorbed into $\mathcal{Q}'(\upsilon)$, $\mathcal{Q}'(\upsilon+1)$ (Eq. 11) and the ring structure is restored for the next gate layer.	n cores at $\mathcal{B}$; $\mathcal{O}(n\mathcal{B}^2)$
7	Measurement and sampling	All qubits are measured in the Pauli-Z basis; the probabilities of the 2^n outcomes (Eq. 12) form the context vector $\mathbf{h}$.	$\mathbf{h}\in\mathbb{R}^{2^n}$; $\mathcal{O}(n\mathcal{B}^3)$ per chunk
8	TTN mapping $\rightarrow \Delta\mathcal{W}$	The tensor-train network maps $\mathbf{h}$ onto the rank-r factors (A_ℓ, B_ℓ) of every adapted layer (Eq. 14).	$\Delta\mathcal{W}_\ell = B_\ell A_\ell$ per adapted layer

A.3 Projector Measurement Loss and its Analytical Gradient

The fine-tuning objective actually optimized in all experiments is the token-level cross-entropy of Eq. 15 in the main text. In this Section, we provide, for completeness, the projector-based measurement loss on the generator output that is used to analyze the circuit landscape (Appendix A.4) and the analytical gradient of that loss with respect to the generated angles ω_j. Recall that the angles are functions of the TR cores, so the gradient with respect to the actual trainable parameters follows by the chain rule. With $\theta_j = |\lambda_j\rangle\langle\lambda_j|$, the gradient of the measurement loss $\mathcal{L}(\omega) = \sum_j f(\langle\theta_j\rangle_\omega, t_j)$

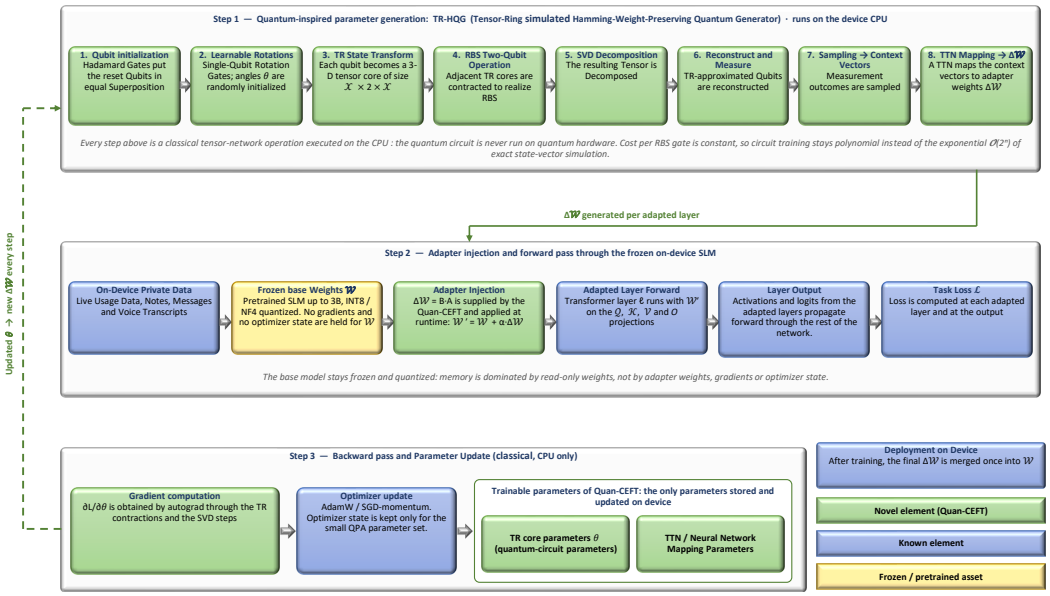

Figure 5: End-to-end flow of one Quan-CEFT optimization step. Step 1: quantum-inspired parameter generation by TR-HQG (eight stages, all classical tensor-network operations). Step 2: the generated $\Delta\mathcal{W}$ is injected into the frozen backbone and the adapted layers are run forward to the task loss. Step 3: gradients are obtained by automatic differentiation through the TTN mapping, the TR contractions and the SVD steps, and only the TR cores and TTN parameters are updated. After training, the final $\Delta\mathcal{W}$ is merged once into $\mathcal{W}$. The figure depicts the intended on-device deployment setting (frozen, possibly quantized SLM); the experiments in Sec. 5 use un-quantized backbones on a CPU. In the figure, $\mathcal{X}$ denotes the bond dimension $\mathcal{B}$ and θ the TR-core parameters.

Table 6: Illustrative per-gate cost (in multiply–accumulate units) of applying one two-qubit RBS gate with exact statevector simulation, $\mathcal{O}(2^n)$, versus the Tensor-Ring simulation, $\mathcal{O}(\mathcal{B}^3)$, which is independent of the qubit count n. Two bond dimensions are shown: $\mathcal{B} = 8$ and the $\mathcal{B} = 16$ setting used in this work (Sec. A.5.2). The trainable circuit parameters scale as $2n\mathcal{B}^2$, set by n and $\mathcal{B}$ only and independent of the hidden size of the adapted model.

Qubits	**Statevector** 2^n	**TR** ($\mathcal{B} = 8$) $\mathcal{B}^3$	**TR** ($\mathcal{B} = 16$) $\mathcal{B}^3$
8	256	512	4,096
12	4,096	512	4,096
16	65,536	512	4,096
20	1,048,576	512	4,096
24	16,777,216	512	4,096

is evaluated as follows:

$$\begin{aligned}\frac{\delta\mathcal{L}(\omega)}{\delta\omega_j} &= \langle 0|\psi^\dagger(\omega_j)\frac{\delta\mathcal{U}^\dagger(\omega)}{\delta\omega_j}\theta_j\mathcal{U}(\omega)\psi(\omega_j)|0\rangle + \langle 0|\psi^\dagger(\omega_j)\mathcal{U}^\dagger(\omega)\theta_j\frac{\delta\mathcal{U}(\omega)}{\delta\omega_j}\psi(\omega_j)|0\rangle \\ &= \langle 0|\psi^\dagger(\omega_j)\mathcal{U}_1^\dagger(\omega_1)\cdots\frac{\delta\mathcal{U}_j^\dagger(\omega_j)}{\delta\omega_j}\cdots\mathcal{U}^\dagger{}_n(\omega_n)\theta_j\mathcal{U}(\omega)\psi(\omega_j)|0\rangle + \\ &\qquad \langle 0|\psi^\dagger(\omega_j)\mathcal{U}^\dagger(\omega)\theta_j\mathcal{U}_n(\omega_n)\cdots\frac{\delta\mathcal{U}_j(\omega_j)}{\delta\omega_j}\cdots\mathcal{U}_1(\omega_1)\psi(\omega_j)|0\rangle \\ &= \langle 0|\psi^\dagger(\omega_j)\mathcal{U}_-^\dagger[i\psi_j]\mathcal{U}_+^\dagger\theta_j\mathcal{U}(\omega)\psi(\omega_j)|0\rangle + \langle 0|\psi^\dagger(\omega_j)\mathcal{U}^\dagger(\omega)\theta_j\mathcal{U}_+[-i\psi_j]\mathcal{U}_-\psi(\omega_j)|0\rangle\end{aligned} \tag{16}$$

where $\mathcal{U}_j(\omega_j) = e^{-i\omega_j G_j}$ with G_j the Hermitian generator of the j-th gate ($G_j = Y/2$ for $\mathcal{R}_y$ and the RBS generator $\frac{1}{2}(XY - YX)$ restricted to the two target qubits), and $\mathcal{U}_-$, $\mathcal{U}_+$ denote the products of gates preceding and following gate j, respectively. When the generator output is used as a binary predictor (a diagnostic setting, not the fine-tuning task), the measurement loss is the binary

Algorithm 1 TR-approximated Hamming-weight preserving Quantum Generator (one generator layer)

Require: Qubit count n, bond dimension $\mathcal{B}$, depth L, TR cores $\{\mathcal{Q}(\upsilon)\}_{\upsilon=1}^{n}$, physical dimension $\mathcal{D}=2$

Ensure: Context vector $\mathbf{h}\in\mathbb{R}^{2^n}$ with entries $\mathcal{E}_{ij}$ (Eq. 12)

Initialization. Set every qubit to $|0\rangle$; represent qubit υ by a core $\mathcal{X}(\upsilon)\in\mathbb{R}^{\mathcal{B}\times\mathcal{D}\times\mathcal{B}}$ whose only non-zero entry is $\mathcal{X}(\upsilon)^{0}_{11}=1$ $\{|0\cdots0\rangle$ in TR form$\}$

Apply the Hadamard matrix to the physical index of every core to obtain the equal superposition $\mathcal{H}|0\rangle^n$ {no data encoding}

Rotation Angles for TR cores. Randomly generate rotation angles as TR core parameters to train $\omega=(\omega_1,\ldots,\omega_n)$ {the only trainable circuit inputs}

for $\upsilon=1$ **to** n **do**

  Apply $\mathcal{R}_y(\omega_\upsilon)$ (and $\mathcal{R}_z$ if phase is used) to the physical index of $\mathcal{X}(\upsilon)$, giving $\mathcal{Q}(\upsilon)$ (Eq. **??**)

end for

for layer $l=1$ **to** L **do**

  for each adjacent pair $(\upsilon,\upsilon+1)$ in the V-shaped RBS ladder **do**

    Bring the ring into orthogonal (canonical) form centred at $(\upsilon,\upsilon+1)$

    $\mathcal{Z}(\upsilon,\upsilon+1)\leftarrow$ contract $\mathcal{Q}(\upsilon)$ and $\mathcal{Q}(\upsilon+1)$ over their shared bond (Eq. 8) {rank-4 tensor}

    $\mathcal{Z}'\leftarrow \mathrm{RBS}(\upsilon,\upsilon+1)\cdot\mathcal{Z}$ applied on the joint physical indices $(y_\upsilon,y_{\upsilon+1})$(Eq. 9){block-diagonal on $\{|01\rangle,|10\rangle\}$, Hamming weight preserved}

    Reshape $\mathcal{Z}'$ to $(y'_\upsilon x_\upsilon)\times(y'_{\upsilon+1}x_{\upsilon+2})$ and compute the SVD $\mathcal{Z}'=\mathcal{V}\mathcal{S}\mathcal{M}$ (Eq. 10) {$2\mathcal{B}$ singular values}

    Keep the $\mathcal{B}$ largest singular values: $\mathcal{S}\rightarrow\mathcal{S}'$, truncate $\mathcal{V},\mathcal{M}$ accordingly

    $\mathcal{Q}'(\upsilon)\leftarrow\mathcal{V}\mathcal{S}'$, $\mathcal{Q}'(\upsilon+1)\leftarrow\mathcal{M}$ (Eq. 11); replace $\mathcal{Q}(\upsilon),\mathcal{Q}(\upsilon+1)$ by $\mathcal{Q}'(\upsilon),\mathcal{Q}'(\upsilon+1)$ {ring restored at bond dimension $\mathcal{B}$}

  end for

end for

Measurement. For each basis outcome λ_j, $j=0,\ldots,2^n-1$ (processed in chunks), contract the ring with its conjugate with physical indices fixed to the bit-string λ_j to obtain $\mathcal{E}_{ij}=|\langle\lambda_j|\psi\rangle|^2$ (Eq. 12)

return $\mathbf{h}=[\mathcal{E}_{ij}]_j$

cross-entropy

$$\underset{\omega}{\operatorname{argmin}}\,\mathcal{L}_{(\omega)}=\sum_{j}^{\mathcal{N}}[t_j\log f(\omega_j)+(1-t_j)\log\{1-f(\omega_j)\}]\,. \tag{17}$$

where $t_j\in\{0,1\}$ is the target and $f(\omega_j,\omega)\in(0,1)$ is the probability of the outcome λ_j under the generated state, obtained from Eq. 12 as

$$f(\omega)=\langle0|\psi^\dagger(\omega_j)\mathcal{U}^\dagger(\omega)\theta_j\mathcal{U}(\omega)\psi(\omega_j)|0\rangle\,,\qquad \hat{t}_j=\begin{cases}1, & \text{if } f(\omega)>\frac{1}{2}\\ 0, & \text{otherwise,}\end{cases} \tag{18}$$

so that both logarithms in Eq. 17 are well defined and the thresholded prediction $\hat{t}_j$ is used only at evaluation time. As an alternative to gradient descent on all cores simultaneously, the TTN mapping can also be trained by a DMRG-like sweeping technique White (2005), in which a stochastic gradient step with a learning rate updates one local bond tensor $\mathcal{B}^{TT}_{j,j+1}$ at a time: the gradient of the loss with respect to $\mathcal{B}^{TT}_{j,j+1}$ is obtained by writing the network output as $f(\omega_j)=\mathcal{T}^{TT}\mathcal{B}^{TT}$, where $\mathcal{T}^{TT}$ is the contraction of every tensor in the TT-MLP other than the bond tensor $\mathcal{B}^{TT}$. All results reported in this paper, however, are obtained by updating the TR cores and all TT-MLP cores jointly with *AdamW* by back-propagation, as stated in Sec. 4.1; the DMRG sweep is described only for completeness.

A.4 Gradient Variance and Barren Plateaus in TR-HQG

To investigate the presence of barren plateaus within our TR-HQG framework, it is essential to evaluate the variance of the partial derivative of the empirical loss function with respect to a generated

Algorithm 2 Quan-CEFT Training Loop

Require: Frozen backbone weights $\{W_\ell\}$ for the adapted projections, rank r, chunk size n_{MLP}, training blocks $\{z_{1:T}\}$, optimizer ($AdamW$), number of steps S

1: Set $n = \lceil \log_2 \lceil \max_\ell m_\ell / n_{\mathrm{MLP}} \rceil \rceil$ with $m_\ell = r(d_{\mathrm{in}} + d_{\mathrm{out}})$; initialize TR cores $\{\mathcal{Q}(\upsilon)\}_{\upsilon=1}^{n}$ with randomly generated rotaion angles and the TTN mapping $\mathcal{W}_{\mathrm{TT}}$; $\Theta \leftarrow \{\mathcal{Q}(\upsilon)\} \cup \{\mathcal{W}_{\mathrm{TT}}\}$ {the only trainable and optimizer-tracked parameters}
2: **for** step $s = 1$ **to** S **do**
3: $\quad \mathbf{h} \leftarrow$ Algorithm $1(n, \mathcal{B}, L, \{\mathcal{Q}(\upsilon)\})$ {one shared generator evaluation per step}
4: $\quad$ **for** each adapted projection ℓ **do**
5: $\quad\quad \mathrm{vec}([A_\ell; B_\ell]) \leftarrow \Pi_\ell \mathcal{W}_{\mathrm{TT}}(\mathbf{h})$; $\Delta\mathcal{W}_\ell \leftarrow B_\ell A_\ell$ (Eq. 14) {only the TTN mapping is applied per layer}
6: $\quad$ **end for**
7: $\quad$ Sample a mini-batch of blocks and run the backbone forward with $W_\ell + \Delta\mathcal{W}_\ell$ in every adapted layer; compute $\mathcal{L}_{\mathrm{LM}}(\Theta)$ (Eq. 15)
8: $\quad$ Back-propagate $\mathcal{L}_{\mathrm{LM}}$ through $\Delta\mathcal{W}_\ell$, $\mathcal{W}_{\mathrm{TT}}$, the measurement contraction, the SVD/RBS sweeps and the ring contraction into Θ {no gradient or optimizer state for $\{W_\ell\}$}
9: $\quad \Theta \leftarrow \mathrm{AdamW}(\Theta, \nabla_\Theta \mathcal{L}_{\mathrm{LM}})$
10: $\quad$ (optional) record circuit-health diagnostics: quantum influence, entanglement entropy, fraction of saturated angles (Appendix A.5.1)
11: **end for**
12: **Deployment.** Regenerate $\{\Delta\mathcal{W}_\ell\}$ from the final Θ and merge once: $W_\ell \leftarrow \mathcal{W}_\ell + \Delta\mathcal{W}_\ell$; inference then follows the unmodified backbone path with no simulator and no added latency

angle,

$$\delta_\mu \mathcal{L}(\omega) = \frac{\delta \mathcal{L}(\omega)}{\delta \omega_\mu}\,, \tag{19}$$

where $\omega_\mu \in \omega$.
If there exists a barren plateau in the ω_μ direction, then

- $\mathbb{E}_\omega[\delta_\mu \mathcal{L}(\omega)] = 0$ and
- $Var[\delta_\mu \mathcal{L}(\omega)] \in O(c^{-\mathcal{N}})$ for some constant $c > 1$, where $\mathbb{E}[\cdot]$ and $Var[\cdot]$ are designated as expected value and variance over randomly chosen parameters, respectively. Conversely, the absence of a barren plateau requires a polynomial *lower* bound $Var[\delta_\mu \mathcal{L}(\omega)] \in \Omega(1/\mathrm{poly}(\mathcal{N}))$; an upper bound alone does not exclude exponential decay.

By randomly selecting parameters in TR-HQG, the probability of encountering a gradient magnitude $\delta_\mu \mathcal{L}(\omega) > \eta$ can be exponentially influenced by the qubit count $\mathcal{N}$. This relationship can be elucidated using Chebyshev's inequality, showcasing the exponential variation in the probability with respect to the gradient magnitude threshold and the qubit count Cervero Martín et al. (2023) as follows:

$$\mathcal{P}[|\delta_\mu \mathcal{L}(\omega) - \mathbb{E}_\omega[\delta_\mu \mathcal{L}(\omega)]| \geqslant \eta] \leqslant \frac{Var[\delta_\mu \mathcal{L}(\omega)]}{\eta^2} \in O(\frac{c^{-\mathcal{N}}}{\eta^2}) \tag{20}$$

For the local cost $\mathcal{L}_j(\omega) = \langle \theta_j \rangle_\omega$ of TR-HQG built from the projector $\theta_j = |\lambda_j\rangle\langle\lambda_j|$, with the angles ω drawn uniformly at random from $[0, 2\pi)$ and ρ the density matrix of the reference state, the mean gradient is

$$\mathbb{E}_\omega[\delta_\mu \mathcal{L}_j(\omega)] = \mathbb{E}_\omega\left[Tr\left(\rho\, \frac{\delta}{\delta \omega_\mu}\Big(\mathcal{U}^\dagger(\omega)|\lambda_j\rangle\langle\lambda_j|\mathcal{U}(\omega)\Big)\right)\right] = 0\,, \tag{21}$$

because $\delta_\mu \mathcal{U}$ is a commutator with the gate generator G_μ and its average over a full period of ω_μ vanishes. The second moment is bounded by expanding ρ in the computational basis,

$$Var[\delta_\mu \mathcal{L}_j(\omega)] \leqslant 4\, \mathbb{E}_\omega\left[\sum_{k=0}^{2^{\mathcal{N}}-1} \rho_{kk} \left|\langle k|\mathcal{U}^\dagger(\omega)|\lambda_j\rangle\langle\lambda_j|\mathcal{U}(\omega)|k\rangle\right|^2\right] \in O\left(\frac{1}{poly(\mathcal{N})}\right) \tag{22}$$

where the polynomial scaling follows from the block structure of $\mathcal{U}(\omega)$: RBS gates preserve Hamming weight, so $\langle\lambda_j|\mathcal{U}(\omega)|k\rangle$ is non-zero only if $|k\rangle$ and $|\lambda_j\rangle$ have the same Hamming weight h,

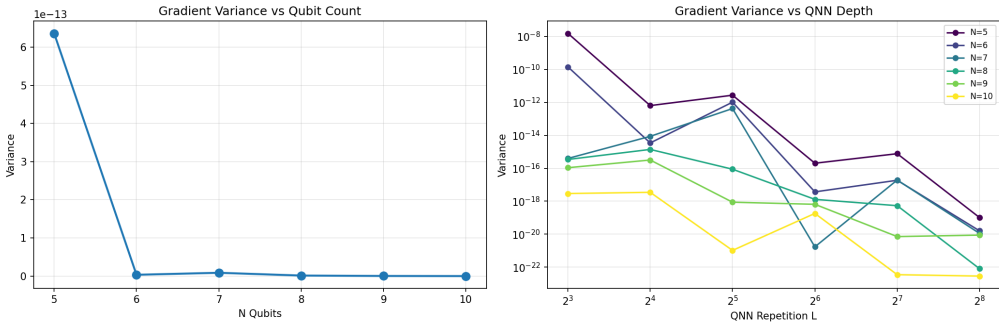

Figure 6: Variance of the loss gradient with respect to the TR angle parameters. Left: variance versus qubit count $\mathcal{N}$ at fixed depth. Right: variance versus generator depth L for $\mathcal{N} = 5, \ldots, 10$ (log scale).

and each term is an amplitude within a $\binom{\mathcal{N}}{h}$-dimensional invariant subspace rather than the full $2^{\mathcal{N}}$-dimensional space. We emphasize that this argument only establishes that the variance is not forced to decay faster than polynomially by the dimension of the reachable subspace; it is an upper bound and does not by itself constitute a proof of a polynomial lower bound $\Omega(1/\mathrm{poly}(\mathcal{N}))$, which would require a Haar-type analysis over the HWP subgroup at the specific depths used (moderate L) and is left for future work. By contrast, if the RBS entanglers were replaced by generic CNOT-based hardware-efficient layers, as in the TR-VQC of Peddireddy et al. (2023); Konar et al. (2025), the circuit would explore the full $2^{\mathcal{N}}$-dimensional space and a barren-plateau cost landscape would be expected, with exceptions only in exponentially small regions around local minima; TR-HQG contains no CNOT or SWAP gates. Finally, since the trainable parameters are the TR cores $\{\mathcal{Q}(v)\}$ rather than the angles, the gradient that drives training is $\nabla_{\mathcal{Q}}\mathcal{L} = J^{\top}\nabla_{\omega}\mathcal{L}$ with $J = \partial\omega/\partial\mathcal{Q}$ the Jacobian of the ring contraction; a well-conditioned J (bounded singular values, which we monitor through the entanglement and saturation diagnostics of Appendix A.5.1) neither amplifies nor suppresses the angle gradient exponentially, so the empirical variance measurements below, taken with respect to the angles, are indicative of the landscape seen by the optimizer.

Fig. 6 reports the empirical variance of $\partial\mathcal{L}/\partial\omega$ over random initializations. Beyond $\mathcal{N} = 6$ qubits the variance flattens into the 10^{-15}–10^{-18} range and no longer decays with $\mathcal{N}$ at the rate expected of an exponential barren plateau, while the largest drop occurs between $\mathcal{N} = 5$ and $\mathcal{N} = 6$ where the mapping network first becomes the dominant parameter block. With respect to depth, variance decreases gradually with L for every $\mathcal{N}$, resembling the depth dependence of classical deep networks rather than a circuit-induced plateau. This supports operating the generator at moderate depth ($L \leqslant 2^5$).

A.5 Additional Results

To test whether the proposed parameterization transfers across architectures, we consider four causal language-model backbones with different attention layouts and scales: GPT-2 (124M), Pythia-160M, OPT-125M, and TinyLlama-1.1B. GPT-2 and Pythia use fused attention projections, whereas OPT and TinyLlama use separate query, key, value, and output projections. The benchmark therefore also probes whether Quan-CEFT remains effective when the number and shape of the adapted modules change. Parameter-budget and generator-setting analyses additionally report Gemma-2-2B, following the protocol of Liu et al. (2025).

All benchmark numbers are obtained with exact statevector simulation on CPU under a quick-budget protocol of five optimization steps. This budget is intended to compare initialization behavior, parameter efficiency, and early adaptation trends; it is not a measure of fully converged fine-tuning performance, and we interpret perplexity and generation scores accordingly. The TR simulator used for scaling experiments is validated against the exact statevector in Sec. A.5.2.

Language Modeling Behavior: Under the five-step adaptation budget, LoRA attains the lowest perplexity on GPT-2, OPT, and Pythia, with PPLs of 25.78, 26.16, and 25.14, respectively. Quan-CEFT remains close on these backbones (26.21, 26.73, and 25.37), despite using $8.1\times$ to $16.1\times$ fewer trainable parameters. In contrast, Quan-CEFT achieves the best result on TinyLlama, reducing PPL to 7.46 compared with LoRA's 7.88 while using $60.5\times$ fewer trainable parameters.

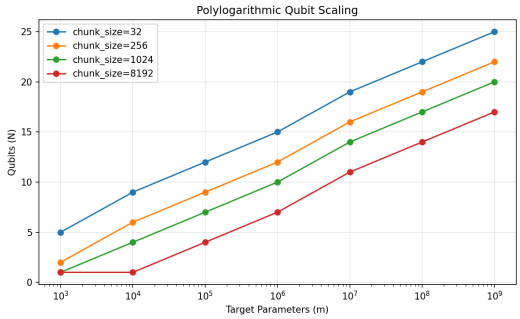

Figure 7: Polylogarithmic qubit scaling for Quan-CEFT: The number of simulated qubits $\mathcal{N} = \lceil \log_2 \lceil m_\ell / n_{\text{MLP}} \rceil \rceil$ required to generate m target parameters grows logarithmically in m for all mapping chunk sizes n_{MLP}.

Table 7: Wall-clock time to regenerate all adaptation weights on GPT-2 (CPU) with one shared generator versus one generator per adapted layer.

Adapted layers	Shared (ms)	Per-layer (ms)	Speed-up
2	50.1	104.6	2.1×
4	60.3	253.3	4.2×
8	87.0	655.7	7.5×
12	100.4	1221.1	12.2×

Relative to the unadapted models, both LoRA and Quan-CEFT improve PPL on Pythia, by 1.8% and 0.9%, respectively. Quan-CEFT also improves TinyLlama PPL by 1.5%, whereas LoRA increases it by 4.1%. On GPT-2 and OPT, neither method improves over the base-model PPL within this very short training budget. Overall, these results show that a larger trainable parameter count can be advantageous on some backbones, but does not alone determine short-budget adaptation performance: the compressed Quan-CEFT generator is competitive throughout and performs best on TinyLlama.

Polylogarithmic Qubit-Scaling: Fig. 7 shows the polylogarithmic relationship between qubits and generated parameters. An $\mathcal{N}$-qubit generator exposes $2^{\mathcal{N}}$ measurement bases, each mapped to a chunk of n_{MLP} target parameters; the required qubit count is therefore $\mathcal{N} = \lceil \log_2 \lceil m_\ell / n_{\text{MLP}} \rceil \rceil$. A $1000\times$ increase in the number of target parameters costs only ≈ 10 additional qubits, and larger chunk sizes shift the curve downward by a constant offset.

Comparison with Quantum-inspired Baselines: From Tab. 3, QPA also has a nearly constant budget (105,208–105,328), but its dense mapping network is roughly 5.7–$5.8\times$ larger than the TTN mapping used by Quan-CEFT. Quantum-PEFT are more compact than LoRA on the small backbones but grow from 36,864 on GPT-2 to 240,416 on TinyLlama, so Quan-CEFT is $2.0\times$–$12.9\times$ smaller than Quantum-PEFT. Across all backbones, Quan-CEFT uses 2–$6\times$ fewer parameters than the most compact quantum baseline for that backbone.

A.5.1 Sensitivity to Generator Settings

In each optimization step, the generator must be re-evaluated to materialize the current adaptation weights. Tab. 7 compares a single generator shared by all layers with independent per-layer generators. The per-layer design scales linearly (≈ 100 ms per layer), whereas the shared design scales sub-linearly, since the circuit is evaluated once and only the TTN mapping is applied per layer; at 12 layers the shared generator is $12.2\times$ faster (100 ms vs. 1,221 ms). This design choice is what keeps our Quan-CEFT per-step latency (1,870.8 ms on GPT-2) below that of the other quantum-inspired baselines despite the additional circuit simulation; per-step latency and peak CPU memory of LoRA, QPA and Quan-CEFT on GPT-2 are reported in Tab. 9 (Appendix).

TR-HQG Circuit Health Analysis: Since the TR-HQG circuit is a fixed evaluator, a degenerate generator could silently collapse to a near-classical mapping. We therefore monitor three diagnostics during training (Tab. 8). With a `tanh` squashing of the TR outputs into $[-\pi, \pi]$, a high learning rate

Table 8: Circuit-health diagnostics on GPT-2 after a short training run for two angle activations and two learning rates. Quantum influence is the relative ℓ_2 change of the generated adaptation attributable to the circuit; entropy is the maximum bipartite entanglement entropy (bits); Saturated is the fraction of angles at $\pm\pi$.

Config	Loss	Quantum infl.	Max entropy	Saturated
tanh, lr$= 10^{-2}$	$5.719 \to 5.664$	0.0050	0.331	0%
tanh, lr$= 10^{-1}$	$5.719 \to 5.731$	0.0011	0.476	58%
periodic, lr$= 10^{-2}$	$5.719 \to 5.712$	**0.0306**	**0.767**	0%
periodic, lr$= 10^{-1}$	$5.719 \to 5.685$	0.0023	0.759	0%

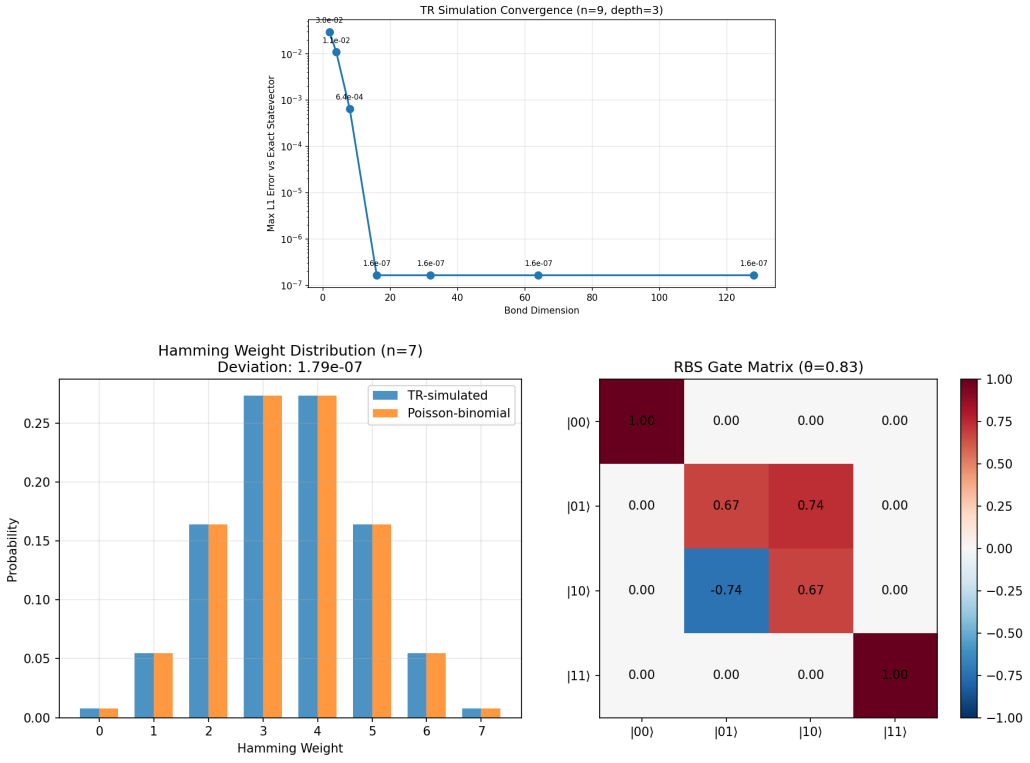

Figure 8: Left: maximum ℓ_1 error of the TR-simulated state against the exact statevector as a function of bond dimension $\mathcal{B}$ (n=9 qubits, depth 3). Right: Hamming-weight distribution of the TR-simulated state versus the analytic Poisson-binomial invariant (n=7), together with the RBS gate matrix, which acts only on the $\{|01\rangle, |10\rangle\}$ subspace.

drives 58% of the angles to $\pm\pi$, where the RBS gate becomes diagonal; entanglement collapses, the quantum contribution falls to 0.1%, and the loss increases. The periodic activation, which feeds the ring trace directly to the 2π-periodic gates, has no saturation regime: it retains the highest entanglement entropy (0.77 bits) and, at lr$= 10^{-2}$, the largest quantum influence (3.1%), while still reducing the loss. We therefore adopt the periodic activation by default and report `circuit_health()` alongside the training loss.

A.5.2 Fidelity of the Tensor Ring Simulation

The TR simulator replaces the $\mathcal{O}(2^n)$ statevector with n cores of bond dimension $\mathcal{B}$, at cost $\mathcal{O}(n\mathcal{B}^3)$. Fig. 8 (left) shows that the maximum ℓ_1 error decreases monotonically with $\mathcal{B}$, from 2.97×10^{-2} at B=2 to 6.4×10^{-4} at $\mathcal{B}$=8, and reaches the single-precision floor of 1.64×10^{-7} at $\mathcal{B}$=16, beyond which further increases in $\mathcal{B}$ bring no change. A bond dimension of 16 is therefore sufficient for exact agreement at the depths used in this work, which is the setting we use throughout. Fig. 8 (right) verifies the Hamming-weight-preserving property: since the RBS gate is block-diagonal with a 2×2 rotation on $\{|01\rangle, |10\rangle\}$, the Hamming-weight distribution of the output state must equal the

Poisson-binomial distribution induced by the input rotations; the TR-simulated distribution matches it with a maximum deviation of 1.79×10^{-7}.

From Tab. 9, it is evident that our Quan-CEFT generates coherent, but less concise summaries than LoRA with minimal parameters, LoRA captures fluent, key information details with maximum parameters, and QPA yields many non-meaningful sentences.

Table 9: Text Summarization using standard LoRA, QPA and our Quan-CEFT on GPT-2 (124 M) model with an end-to-end training on CPU

Input Text	**LoRA**	**QPA**	**Quan-CEFT**
"The quantum computer uses quantum bits, or qubits, which can exist in multiple states simultaneously due to superposition. This allows quantum computers to process vast amounts of information in parallel, potentially solving complex problems much faster than classical computers. However, qubits are extremely fragile and require near absolute zero temperatures to maintain coherence"	"Quantum computers are extremely fast and perform extremely well in parallel. However they are extremely vulnerable to quantum fluctuations due to their low energy states. Quantum computers can store trillions of bits simultaneously and store millions of bits concurrently. However qubits require extremely high temperatures to process complex data simultaneously. Therefore qubits can only"	"The quantum computer is a quantum computer. The qubits The first qubits were discovered in the early 1990s. Quantum computers are a quantum computing phenomenon. In the early days of quantum computing. A few years ago, a few years later. This is the quantum."	"The quantum computer is a quantum computer that can perform computations in a single state at a rate of about 1/10th of the speed of light. It is also capable of performing many other computations at the same time. Quantum computers are extremely fast and can perform many other calculations at."
Model Parameters:	147, 456	105, 208	18, 248
Peak CPU Memory Usage (MB) :	2498	1413	1426
Latency (ms/step): :	1426	3563	1871