QEA: A Quantum-Enhanced Adapter for Parameter-Efficient Fine-Tuning of LLMs

Fanqi Dong[1,†], Chenhao Zhang[1,†], Yuxin Deng[2,1,*], Chenyang Xu[1,*]
[1]Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, Shanghai 200062, China
[2] MoE Key Laboratory of Interdisciplinary Research of Computation and Economics,
Shanghai University of Finance and Economics, Shanghai 200433, China
51275902169@stu.ecnu.edu.cn, 51275902178@stu.ecnu.edu.cn, yxdeng@msg.sufe.edu.cn, cyxu@sei.ecnu.edu.cn

***Abstract*—Quantum neural architectures are rapidly emerging as a new paradigm for efficient representation learning. Their intrinsic structure, built upon superposition and entanglement, enables expressive transformations with remarkably few parameters. Meanwhile, parameter-efficient fine-tuning (PEFT) has become a key approach for adapting large language models (LLMs), typically by inserting lightweight adapters or low-rank modules such as LoRA. However, strict low-rank constraints may limit representational capacity under complex domain shifts. This raises a natural question: *what if quantum parameterization is introduced into the adapter framework?* To address this, we propose a Quantum-Enhanced Adapter (QEA), which adds a parameterized quantum circuit branch in parallel to each Transformer feed-forward network while keeping all pretrained weights frozen. QEA encodes token features as rotation angles, reads out Pauli-Z expectations, and fuses quantum and classical paths via a token-wise gate fusion with a soft-to-hard schedule for stable training. On LLaMA-7B and LLaMA-3.2-1B across WikiText-2 and WikiText-103, QEA achieves comparable or lower perplexity than strong PEFT baselines while using only about 0.066-0.068% of the model parameters. These results highlight QEA's ability to deliver substantial parameter savings without sacrificing performance. Our work demonstrates the promise of quantum-enhanced, parameter-efficient fine-tuning for LLMs, providing a scalable quantum-classical solution.**

***Index Terms*—parameter-efficient fine-tuning, large language models, parameterized quantum circuit, gate fusion, fine-tuning**

I. Introduction

Large language models (LLMs) are commonly adapted via parameter-efficient fine-tuning (PEFT), which freezes the backbone and introduces lightweight trainable components, such as adapters [1], prefix tuning [2], and LoRA [3]. While effective and economical, methods that rely on strict low-rank assumptions can inherently limit representational expressiveness under complex domain shifts or nuanced downstream tasks. Although recent advances [4]–[6] relax these constraints or reshape update parameterizations, the fundamental trade-off between parameter parsimony and representational capacity remains.

Concurrently, a burgeoning line of research in quantum-enhanced methods offers a fundamentally different route to high expressivity with minimal parameters. These approaches can be broadly categorized: (1) Quantum-inspired parameterizations, such as QuanTA [7], which emulate high-rank updates using tensor networks; (2) Logarithmic-complexity unitary updates, exemplified by Quantum-PEFT [8], which leverages quantum circuit principles; and (3) Quantum-generated classical weights, as in QPA [9], where a quantum network acts as a parameter generator. Collectively, these works suggest that quantum principles may help alleviate the rank limitations of classical PEFT. Motivated by this insight, we design hybrid modules that plug into existing Transformer architectures and enrich representational capacity under extreme parameter constraints.

We propose QEA, a simple, simulator-friendly module placed in parallel to each feed-forward network (FFN). Concretely, given the hidden state, QEA (i) projects it to an n_q-dimensional embedding, (ii) feeds the result to a parameterized quantum circuit (PQC) with layered R_X/R_Y/R_Z rotations and entangling patterns, (iii) performs Pauli-Z expectation readout to obtain a quantum feature vector, and (iv) maps it back to d_{model} dimensions before token-wise fusion with the frozen FFN via a learnable two-way gate. A two-phase training schedule first forces the quantum path to warm up and then enables temperature-controlled soft routing, improving stability and branch utilization.

Our contributions are summarized as follows: (1) We introduce QEA, a parallel PQC branch with token-wise gated fusion that augments FFN expressiveness while keeping the backbone frozen. (2) We provide a practical, bottlenecked classical to quantum to classical implementation and a soft-to-hard gating schedule that stabilizes optimization. (3) On LLaMA-7B and LLaMA-3.2-1B with WikiText-2/103, QEA achieves competitive or lower perplexity (PPL) than strong PEFT baselines with only 0.066–0.068% trainable parameters, alongside reduced activation memory during training.

II. Related Work

a) Full Fine-tuning: Early approaches adapted large pretrained models via full fine-tuning [10]. While effective, this strategy becomes infeasible for billion-scale models due to excessive computation and storage, and also risks overfitting and catastrophic forgetting. These limitations motivated research on parameter-efficient alternatives.

This work was supported by the National Key R&D Program of China under Grant No. 2023YFA1009403 and the National Natural Science Foundation of China under Grant No. 62472175.

† These authors contributed equally to this work.

*Corresponding authors: Yuxin Deng and Chenyang Xu.

b) Parameter-Efficient Fine-Tuning: PEFT methods update only a small subset of parameters or introduce lightweight modules, while freezing most weights. Representative techniques include adapters [1], prefix tuning [2], and LoRA [3]. LoRA constrains updates to a low-rank subspace, drastically reducing trainable parameters. However, low-rank constraints may limit expressiveness. Recent extensions like MORA [11] and GORA [12] relax the rank assumption to capture more complex transformations.

c) Quantum Transformers and Potential Advantage: Recently, researchers have explored integrating PQC into Transformer architectures [13]. Quantum superposition and entanglement allow encoding high-dimensional dependencies that classical networks struggle to capture. Prior studies [14] propose replacing FFN or attention with PQCs, though most remain theoretical due to current hardware limitations.

d) Quantum-Inspired and Quantum-Enhanced Fine-tuning: Two directions have emerged: (i) quantum-inspired compression, e.g., QuanTA [7] using tensor decompositions, and (ii) hybrid quantum-classical PEFT, such as Quantum-PEFT [8], which parameterizes full-rank unitary adaptations with logarithmic complexity, and QPA [9], which employs quantum neural networks (QNNs) to generate LoRA-style updates during training but runs classically at inference. These methods demonstrate the potential of quantum-enhanced PEFT, though scalability and deployment remain open challenges.

III. Method

A. Quantum Adapter Framework

We first describe a general quantum adapter that can be plugged next to common host layers, such as linear layers, multi-layer perceptrons (MLPs), FFN blocks, and attention projections, without modifying pretrained weights, akin to classical adapter design principles [1], [15]. The framework exposes three stable interfaces:

- **Q-encoder (classical to quantum)** Maps a token hidden state $\mathbf{h} \in \mathbb{R}^{d_{\text{model}}}$ to n_q quantum parameters that drive a PQC. Angle-based encodings are common and practical.
- **PQCs** A layered circuit with single-qubit rotations and entangling gates that transforms the encoded state; outputs are measured as expectations of observables to form a quantum feature vector.
- **Q-decoder (quantum to classical)** Projects the measured quantum features back to d_{model} dimensions through lightweight mappings, optionally with a residual scale.

The adapter's output is fused token-wise with the host layer via a learnable gate. This modular design allows QEA to be applied at different granularities, ranging from individual sublayers such as FFN or attention to entire MLP or Transformer blocks, while keeping the pretrained backbone frozen. Unlike LoRA, which injects a low-rank update per selected weight matrix [3], our adapter is attached at the sublayer level: QEA forms a single parallel branch that covers the entire FFN mapping. In the SwiGLU-style FFN used by LLaMA, which contains three linear projections (W_{up}, W_{gate}, W_{down}) [16], a LoRA setup typically requires three separate adapters to cover all projections. In contrast, a single QEA branch can cover the FFN as a whole, while multiple QEA branches can be instantiated when finer-grained control is desired. In the following sections and experiments, we take the integration of QEA into the FFN sublayer as a representative example.

B. QEA: Instantiation under the Framework

We now instantiate the generic quantum adapter into QEA, which we place in parallel to the FFN in each Transformer block. Fig. 1 illustrates the overall architecture. The pretrained FFN remains frozen; QEA contributes a parallel update that is gated and added through the residual. The overall design follows the common Transformer FFN notation [17] while introducing a token-wise gate and a quantum branch.

a) Parallel Architecture and Fusion: Let the FFN input be $\mathbf{H} \in \mathbb{R}^{B\times S\times d_{\text{model}}}$. The frozen classical FFN computes

$$\mathbf{H}_{\text{cls}} = \phi\big(\mathbf{H}\mathbf{W}_1 + \mathbf{b}_1\big)\,\mathbf{W}_2 + \mathbf{b}_2, \tag{1}$$

where $\mathbf{W}_1 \in \mathbb{R}^{d_{\text{model}}\times d_{\text{ff}}}$ and $\mathbf{W}_2 \in \mathbb{R}^{d_{\text{ff}}\times d_{\text{model}}}$ are frozen projection matrices, $\mathbf{b}_1, \mathbf{b}_2$ are bias vectors, d_{ff} is the intermediate width, and $\phi(\cdot)$ denotes a nonlinear activation.

In parallel, we introduce a trainable quantum branch over the same input:

$$\mathbf{H}_{\text{q}} = f_{\text{q}}(\mathbf{H};\, \Omega). \tag{2}$$

Here $f_{\text{q}}(\cdot;\Omega)$ denotes the quantum branch mapping parameterized by Ω, which transforms the classical hidden state $\mathbf{H}$ into the quantum-enhanced representation $\mathbf{H}_{\text{q}}$. The parameter set Ω includes the trainable projections and circuit parameters in the quantum branch.

b) Token-wise Fusion via Learnable Gating: To adaptively fuse the two branch outputs, we introduce a lightweight gating module.

(1) Gating logits

$$\mathbf{Z} = \mathbf{H}\mathbf{W}_g + \mathbf{b}_g. \tag{3}$$

Here, $\mathbf{W}_g \in \mathbb{R}^{d_{\text{model}}\times 2}$ and $\mathbf{b}_g \in \mathbb{R}^2$ produce token-wise logits for the classical and quantum paths, with $\mathbf{b}_g$ broadcasted across the batch and sequence dimensions.

(2) Softmax routing

$$\mathbf{G} = \text{softmax}\big(\tfrac{\mathbf{Z}}{T}\big) = [\mathbf{G}_{\text{cls}},\, \mathbf{G}_{\text{q}}], \tag{4}$$

where the softmax is applied over the last dimension. $\mathbf{G}_{\text{cls}}$ and $\mathbf{G}_{\text{q}}$ denote the routing weights for the classical and quantum branches, respectively. A smaller temperature T yields sharper routing.

(3) Fusion and residual

$$\mathbf{H}_{\text{fuse}} = \mathbf{G}_{\text{cls}} \odot (\alpha_{\text{cls}}\mathbf{H}_{\text{cls}}) + \mathbf{G}_{\text{q}} \odot (\alpha_{\text{q}}\mathbf{H}_{\text{q}}), \tag{5}$$

where $\odot$ denotes element-wise multiplication (with broadcasting over the feature dimension), and $\alpha_{\text{cls}}, \alpha_{\text{q}} \in \mathbb{R}$ are learnable scaling factors.

Finally, we add the fused output through a residual connection:

$$\mathbf{Y} = \mathbf{H} + \mathbf{H}_{\text{fuse}}. \tag{6}$$

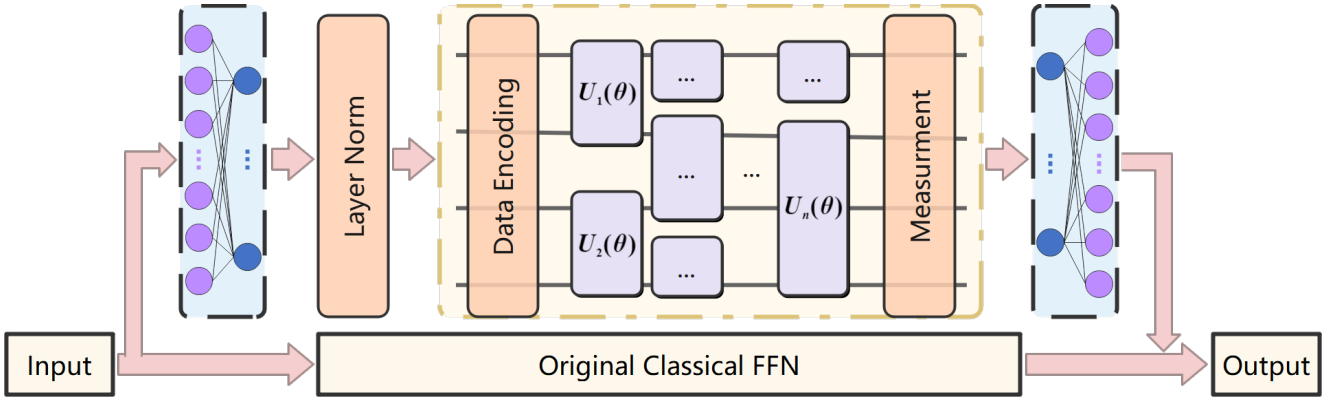

Fig. 1. Quantum adapter architecture within a Transformer block, where a parallel quantum branch is fused with the frozen FFN via a token-wise gate.

c) Two-Phase Gating Schedule (Soft to Hard): We adopt a two-phase regime controlled by training step t and threshold $\tau_{\mathrm{q}} = 5000$:

Phase I (Forced Quantum): if $t < \tau_{\mathrm{q}}$, then

$$\mathbf{G}_{\mathrm{q}} \equiv 1, \quad \mathbf{G}_{\mathrm{cls}} \equiv 0.$$

Phase II (Free Soft Routing): if $t \geq \tau_{\mathrm{q}}$, then

$$\mathbf{G} = \mathrm{softmax}(\mathbf{Z}/T).$$

At initialization, the PQC branch can be difficult to optimize, yielding weak early-stage learning signals, whereas the frozen pretrained FFN is already strong. If token-wise gating is enabled from the start, the gate may quickly collapse to the classical path, assigning near-zero weight to the quantum branch and starving it of gradients. Phase I acts as a warm-up that forces meaningful gradient flow through the PQC, before enabling competitive, temperature-controlled gating in Phase II.

d) Gating Regularization: We employ a regularization loss to stabilize routing:

$$\begin{aligned}\mathcal{L}_{\mathrm{gate}} &= \lambda_{\mathrm{bal}}\, \mathbb{I}[t \geq \tau_{\mathrm{q}}]\, \mathrm{Var}(\mathrm{mean}_{B,S}[\mathbf{G}]) \\ &+\lambda_{\mathrm{ent}}\, \mathrm{mean}_{B,S}\left[-\sum_{k=1}^{2} \mathbf{G}^{(k)} \log\left(\mathbf{G}^{(k)} + \varepsilon\right)\right].\end{aligned} \tag{7}$$

Here, ε is a small constant for numerical stability. In practice, Phase I uses reduced weights ($\lambda_{\mathrm{bal}} + \lambda_{\mathrm{ent}} \approx 10^{-3}$), while Phase II uses full weights (on the order of 10^{-2}), and scales the entropy term by 0.1.

e) Quantum Branch Implementation: The quantum feed-forward mapping f_{q} follows a bottleneck-style "classical-to-quantum-to-classical" design.

(1) Classical-to-Quantum Projection. The input hidden state $\mathbf{H}$ is projected into an n_q-dimensional quantum embedding space:

$$\tilde{\mathbf{H}} = \mathrm{LN}\Big(\mathrm{GELU}\big(\mathbf{H}\mathbf{P}_{\mathrm{in},1}\big)\Big)\mathbf{P}_{\mathrm{in},2}, \tag{8}$$

where $\mathbf{P}_{\mathrm{in},1} \in \mathbb{R}^{d_{\mathrm{model}} \times 16}$ and $\mathbf{P}_{\mathrm{in},2} \in \mathbb{R}^{16 \times n_q}$ are trainable linear projections, and LN denotes LayerNorm [18]. All projections are applied token-wise over the last dimension.

(2) PQC. The projected feature $\tilde{\mathbf{H}}$ is encoded into rotation angles of a quantum circuit $\mathcal{Q}_\Theta$ with L layers and n_q qubits. Each layer applies parameterized single-qubit rotations

$$R_X(\theta_{i,1}^{(\ell)})\, R_Y(\theta_{i,2}^{(\ell)})\, R_Z(\theta_{i,3}^{(\ell)}), \quad i = 1, \ldots, n_q,$$

followed by entangling operations: even layers use chain CNOTs, while odd layers use ring CNOTs and CRZ couplings.

(3) Readout. For each token (b, s), let $\tilde{\mathbf{h}}_{b,s} \in \mathbb{R}^{n_q}$ denote the corresponding slice from $\tilde{\mathbf{H}}$. After the PQC produces the output state $|\psi_{b,s}\rangle = \mathcal{Q}_\Theta(\tilde{\mathbf{h}}_{b,s})$, we read out the expectation value of the Pauli-Z operator on each qubit:

$$z_{b,s,i} = \langle \psi_{b,s}|\, Z_i\, |\psi_{b,s}\rangle\,. \tag{9}$$

Here, $z_{b,s,i} \in [-1, 1]$ is the Pauli-Z expectation on the i-th qubit. Stacking the results across all qubits forms the token-level readout vector $\mathbf{z}_{b,s} = [z_{b,s,1}, \ldots, z_{b,s,n_q}]^\top \in \mathbb{R}^{n_q}$, which serves as the quantum feature representation fed into the classical projection-out module (Eq. 10).

(4) Quantum-to-Classical Projection and Residual. The quantum output is mapped back to the model dimension:

$$\hat{\mathbf{H}} = \mathrm{GELU}(\mathbf{z}\mathbf{P}_{\mathrm{out},1})\mathbf{P}_{\mathrm{out},2}, \tag{10}$$

where $\mathbf{P}_{\mathrm{out},1} \in \mathbb{R}^{n_q \times 16}$ and $\mathbf{P}_{\mathrm{out},2} \in \mathbb{R}^{16 \times d_{\mathrm{model}}}$. The final quantum-enhanced feed-forward mapping is

$$f_{\mathrm{q}}(\mathbf{H}; \Omega) = \mathbf{H} + \beta\, \hat{\mathbf{H}}, \tag{11}$$

where $\Omega = \{\mathbf{P}_{\mathrm{in},*}, \mathbf{P}_{\mathrm{out},*}, \Theta\}$ collects all trainable parameters and β is a learnable scaling factor.

AngleEmbedding projects $\tilde{\mathbf{H}}$ to rotation angles. To stabilize training, we apply LayerNorm and GELU on the classical side and augment the quantum block with the residual connection in Eq. (11).

IV. Experiments

In this section, we present an empirical evaluation of the proposed QEA to validate its effectiveness as an ultra-low-parameter fine-tuning mechanism. We evaluate the performance of QEA by conducting benchmarking across various model scales, on both small-scale and large-scale language modeling tasks. Furthermore, we perform extensive ablation

TABLE I
COMPARISON OF QEA AND PEFT BASELINES ACROSS DIFFERENT MODELS AND DATASETS. WE REPORT TRAINABLE PARAMETER RATIO (%) AND PERPLEXITY.

Model (Dataset)	Method	# Params (%)	PPL
LLaMA-7B (WikiText-2)	Pretrained Model	0%	18.73
	QEA (Ours)	**0.068%**	**8.38**
	LoRA	0.286%	11.51
	DoRA	0.305%	11.60
	AdaLoRA	0.214%	11.48
	OFT	0.605%	11.56
LLaMA-7B (WikiText-103)	Pretrained Model	0%	20.10
	QEA (Ours)	**0.068%**	**9.10**
	LoRA	0.286%	11.77
	DoRA	0.305%	11.79
	AdaLoRA	0.214%	12.20
	OFT	0.605%	12.21
LLaMA-3.2-1B (WikiText-2)	Pretrained Model	0%	37.99
	QEA (Ours)	**0.066%**	**15.28**
	LoRA	0.281%	15.36
	DoRA	0.319%	15.29
	AdaLoRA	0.211%	15.53
	OFT	1.175%	15.73
LLaMA-3.2-1B (WikiText-103)	Pretrained Model	0%	39.51
	QEA (Ours)	**0.066%**	**15.88**
	LoRA	0.281%	15.96
	DoRA	0.319%	15.96
	AdaLoRA	0.211%	16.23
	OFT	1.175%	16.56

studies to analyze the experimental results under different settings. Finally, we provide a detailed discussion on the implications of current quantum hardware limitations and the model's robustness against simulated quantum noise.

A. Setup

a) Datasets and Baseline Methods: We evaluate the proposed QEA on two representative large-scale models: LLaMA-7B [19] and LLaMA-3.2-1B [20]. We use WikiText-2 [21] and WikiText-103 [21] as benchmark datasets, representing small-scale and large-scale language modeling, respectively. For efficiency, all experiments on WikiText-103 use a 5% subset of the training data, while WikiText-2 uses the full corpus. QEA is compared against several representative parameter-efficient fine-tuning methods, including LoRA [3], DoRA [4], AdaLoRA [22], and OFT [6].

b) Experimental Settings: All quantum modules are simulated using the PennyLane framework [23] on classical hardware with identical training configurations across methods. Table II summarizes the key hyperparameters used in our QEA experiments. All experiments were performed on a single NVIDIA A6000 GPU with identical training configurations for fair comparison.

Unless specified, we use a two-phase routing schedule. Gate regularizers follow Eq. (7) with small weights during warm-up and larger weights afterward. The trainable parameter count scales linearly with the model dimension and the number of qubits, plus $O(L\, n_q)$ PQC parameters. The backbone is frozen and the plug-in nature is compatible with other PEFT components [1], [3].

TABLE II
HYPERPARAMETER SETTINGS USED IN OUR EXPERIMENTS.

Hyperparameter	Value	Hyperparameter	Value
Qubits	4–8	Batch size	1
Learning rate	7×10^{-4}	Sequence length	128
Seed	42	Optimizer	AdamW
Scheduler	Cosine	Warmup ratio	0.03

TABLE III
COMPARISON BETWEEN LORA AND QEA WHEN APPLIED ONLY TO FFN.

Method	# Params (%)	PPL
LoRA (FFN only)	0.170%	12.00
QEA (FFN only)	0.068%	**9.10**

We report PPL and the proportion of trainable parameters. For the smaller model LLaMA-3.2-1B, we use 4 qubits; for LLaMA-7B, we use 8 qubits. For comparability, we set the adaptation rank of all rank-based PEFT baselines to match the number of qubits n_q used in QEA.

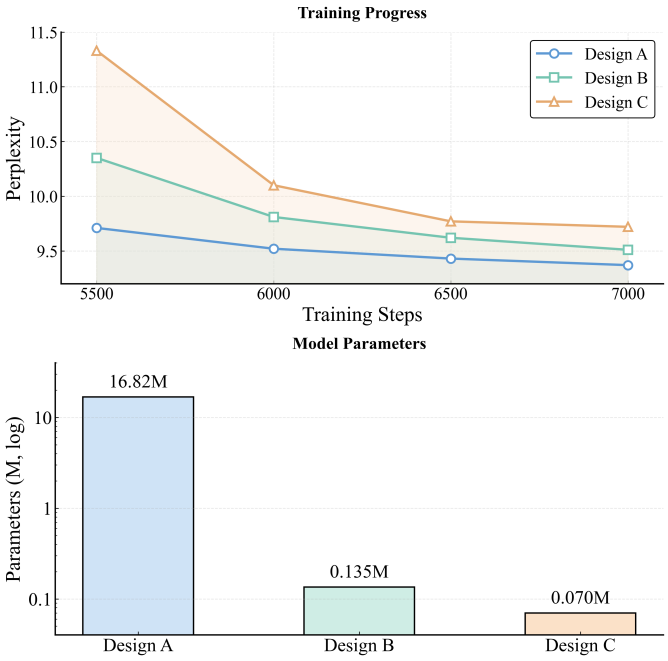

Fig. 2. Ablation study on three classical-to-quantum projection designs in QEA (Designs A–C). Top: training progress (PPL vs. training steps). Bottom: corresponding trainable parameter counts.

B. Main Results

Table I summarizes results on LLaMA-7B and LLaMA-3.2-1B over WikiText-2/103. Under an ultra-small trainable budget, QEA achieves lower or comparable PPL to strong PEFT baselines while keeping the backbone frozen.

TABLE IV
TRAINING AND INFERENCE THROUGHPUT COMPARISON UNDER A UNIFIED SETTING WITH BATCH SIZE = 1 ON A SINGLE NVIDIA A6000 GPU.

Method	Training Throughput (batch/s) ↑	Inference Throughput (batch/s) ↑
LoRA	0.56	**8.4**
DoRA	0.20	2.5
AdaLoRA	0.53	8.3
OFT	0.50	7.9
QEA (Ours)	**1.11**	2.4

TABLE V
TRAINING AND INFERENCE THROUGHPUT UNDER DIFFERENT NUMBERS OF QUBITS. HIGHER IS BETTER FOR BOTH COLUMNS.

Qubit Number	Training Throughput (batch/s) ↑	Inference Throughput (batch/s) ↑
2	**2.5**	**4.8**
4	1.6	3.3
6	1.1	2.4
8	0.8	1.9

a) LLaMA-7B: On WikiText-2, QEA attains 8.38 PPL with only 0.068% trainable parameters, outperforming LoRA and other baselines. On WikiText-103, QEA reaches 9.10 PPL at the same budget, again beating other baselines. Overall, QEA reduces the trainable ratio by roughly $4\times$ versus LoRA while improving accuracy.

b) LLaMA-3.2-1B: With only 0.066% trainable parameters, QEA achieves perplexities of 15.28 on WikiText-2 and 15.88 on WikiText-103, matching or slightly surpassing LoRA at 15.36 and 15.96 with 0.281% trainable parameters and DoRA at 15.29 and 15.96 with 0.319% trainable parameters. These results show that the performance advantage of QEA remains evident at smaller model scales, where the parameter budget is especially constrained.

c) Training and Inference Efficiency: To assess the computational efficiency of QEA, we measure both training and inference throughput and compare it against representative PEFT baselines. Table IV reports the results. QEA achieves the highest training throughput among all compared methods, reaching 1.11 batches/s. This improvement is primarily attributed to the lightweight nature of the quantum-enhanced adapter and the reduced number of trainable parameters. In terms of inference, QEA attains a throughput of 2.4 batches/s, which is comparable to DoRA, although lower than LoRA, AdaLoRA, and OFT.

We further note that current experiments simulate quantum circuits on classical hardware. As quantum computing hardware continues to advance, with improvements in gate fidelity and execution speed, the runtime efficiency of quantum-enhanced modules is expected to improve further. Therefore, the present results should be interpreted as an early exploration of the training and inference efficiency of quantum-enhanced parameter-efficient fine-tuning.

C. Comparison with LoRA (FFN Only)

We conduct a fair comparison between LoRA and QEA by restricting both methods to the FFN sublayer within LLaMA-7B on WikiText-103. As shown in Table III, QEA achieves lower PPL with significantly fewer trainable parameters, demonstrating the efficiency of quantum enhancement under a comparable setting.

D. Ablation Studies

We perform ablation studies on the internal design of QEA, including different classical–quantum projection strategies and the number of qubits n_q.

a) Classical–Quantum Projection Design: For the classical-to-quantum projection, we evaluate three projection designs on LLaMA-7B, denoted as Design A, Design B, and Design C in Fig. 2. Specifically, Design A projects from 4096 to 2048 with an activation function and LayerNorm, followed by a further projection to 8 for an 8-qubit configuration; Design B projects from 4096 to 16, followed by activation and LayerNorm, and then to 8; and Design C uses a direct linear mapping from 4096 to 8. As shown in Fig. 2, Design A achieves the best PPL but introduces substantially more parameters, whereas Design C is the most lightweight but yields worse performance. Design B offers the best trade-off between effectiveness and parameter efficiency, and is therefore adopted as the default configuration in QEA.

b) Effect of the Number of Qubits: We further investigate the impact of the number of qubits on both model performance and runtime behavior. We conduct ablation studies on LLaMA-7B with the WikiText-103 dataset, varying the number of qubits while keeping all other settings fixed. As shown in Fig. 3, increasing the number of qubits does not monotonically improve PPL. In particular, using six qubits yields the best performance on this task, indicating that an appropriate qubit size provides a better balance between expressiveness and optimization stability.

Beyond performance, we also measure the training and inference speed under different qubit configurations. Although classical simulation of parameterized quantum circuits is known to exhibit exponential worst-case complexity with respect to the number of qubits, we deliberately restrict QEA to very small qubit counts and shallow circuits. As a result, the empirical runtime grows moderately as the number of qubits increases, rather than exhibiting prohibitive scaling. Table V summarizes the training and inference speed under different qubit settings.

E. Discussion on Quantum Hardware and Noise

Current noisy intermediate-scale quantum (NISQ) hardware is known to suffer from limited gate fidelity and relatively slow operations, which pose challenges for direct deployment of quantum-enhanced models. For this reason, our primary experiments are conducted on classical quantum simulators to ensure stability and reproducibility.

To approximate realistic hardware conditions, we additionally evaluate QEA by injecting depolarizing, amplitude damping, and phase damping noise into the simulator. Specifically,

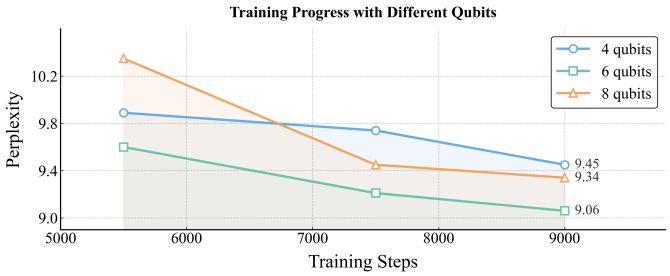

Fig. 3. Ablation study on the number of qubits in QEA. Results on LLaMA-7B with WikiText-103 show that six qubits achieve the best perplexity, while increasing qubits beyond this point leads to diminishing returns.

on LLaMA-7B with WikiText-2, QEA attains a PPL of 9.03 under the noisy setting, compared to 8.38 in the noiseless simulation. Although performance degrades moderately, QEA remains substantially better than classical PEFT baselines, indicating a degree of robustness to quantum noise.

This observation is consistent with prior findings in quantum learning, where moderate noise can sometimes act as an implicit regularizer [24], and with analogous effects reported in classical machine learning, where mild noise has been shown to benefit model robustness and generalization [25].

We emphasize that this work does not claim immediate practical deployment on current NISQ devices. Rather, our goal is to explore PQC-based parameterization as a highly expressive, ultra-low-parameter adaptation mechanism, while keeping the method simulator-friendly and reproducible on classical hardware. As quantum hardware continues to evolve, improvements in cost, fidelity, and gate speed may further enhance the practicality of such hybrid approaches.

V. Conclusion

We introduce QEA, a quantum-enhanced adapter that augments frozen Transformers with a lightweight, parallel PQC branch and token-wise gated fusion. This design encodes hidden states through a compact classical–quantum–classical bottleneck, enhancing expressiveness per parameter while preserving efficiency. QEA remains simulator-friendly and hardware-agnostic, offering a practical bridge between quantum computation and parameter-efficient adaptation. Future work may explore richer circuit structures and deployment on real quantum hardware.

References

[1] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. de Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly, "Parameter-efficient transfer learning for NLP," in *ICML*, ser. Proceedings of Machine Learning Research, vol. 97. PMLR, 2019, pp. 2790–2799.

[2] X. L. Li and P. Liang, "Prefix-tuning: Optimizing continuous prompts for generation," in *ACL/IJCNLP (1)*. Association for Computational Linguistics, 2021, pp. 4582–4597.

[3] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models," in *ICLR*. OpenReview.net, 2022.

[4] S. Liu, C. Wang, H. Yin, P. Molchanov, Y. F. Wang, K. Cheng, and M. Chen, "Dora: Weight-decomposed low-rank adaptation," in *ICML*. OpenReview.net, 2024.

[5] D. J. Kopiczko, T. Blankevoort, and Y. M. Asano, "Vera: Vector-based random matrix adaptation," in *ICLR*. OpenReview.net, 2024.

[6] Z. Qiu, W. Liu, H. Feng, Y. Xue, Y. Feng, Z. Liu, D. Zhang, A. Weller, and B. Schölkopf, "Controlling text-to-image diffusion by orthogonal finetuning," in *NeurIPS*, 2023.

[7] Z. Chen, R. Dangovski, C. Loh, O. Dugan, D. Luo, and M. Soljacic, "Quanta: Efficient high-rank fine-tuning of llms with quantum-informed tensor adaptation," in *NeurIPS*, 2024.

[8] T. Koike-Akino, F. Tonin, Y. Wu, F. Z. Wu, L. N. Candogan, and V. Cevher, "Quantum-peft: Ultra parameter-efficient fine-tuning," in *ICLR*. OpenReview.net, 2025.

[9] C. Liu, C. H. Yang, H. Goan, and M. Hsieh, "A quantum circuit-based compression perspective for parameter-efficient learning," in *ICLR*. OpenReview.net, 2025.

[10] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," in *NAACL-HLT (1)*. Association for Computational Linguistics, 2019, pp. 4171–4186.

[11] T. Jiang, S. Huang, S. Luo, Z. Zhang, H. Huang, F. Wei, W. Deng, F. Sun, Q. Zhang, D. Wang, and F. Zhuang, "Mora: High-rank updating for parameter-efficient fine-tuning," *CoRR*, vol. abs/2405.12130, 2024.

[12] H. He, P. Ye, Y. Ren, Y. Yuan, and L. Chen, "Gora: Gradient-driven adaptive low rank adaptation," *CoRR*, vol. abs/2502.12171, 2025.

[13] S. B. Dharavath, T. Dam, S. Chakraborty, P. Roy, and A. Maiti, "Quantum inverse contextual vision transformers (Q-ICVT): A new frontier in 3d object detection for avs," in *CIKM*. ACM, 2024, pp. 3724–3729.

[14] Y. Chen, H. Lan, B. Gwee, Q. Cao, S. G. Razul, and Z. Lin, "DQSA: dynamic quantized self-attention for multi-task encrypted network traffic classification," *IEEE Trans. Inf. Forensics Secur.*, vol. 21, pp. 304–318, 2026.

[15] J. Pfeiffer, A. Kamath, A. Rücklé, K. Cho, and I. Gurevych, "Adapterfusion: Non-destructive task composition for transfer learning," in *EACL*. Association for Computational Linguistics, 2021, pp. 487–503.

[16] T. Zhu, X. Qu, D. Dong, J. Ruan, J. Tong, C. He, and Y. Cheng, "Llama-moe: Building mixture-of-experts from llama with continual pre-training," in *EMNLP*. Association for Computational Linguistics, 2024, pp. 15 913–15 923.

[17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[18] R. Xiong, Y. Yang, D. He, K. Zheng, S. Zheng, C. Xing, H. Zhang, Y. Lan, L. Wang, and T. Liu, "On layer normalization in the transformer architecture," in *ICML*, ser. Proceedings of Machine Learning Research, vol. 119. PMLR, 2020, pp. 10 524–10 533.

[19] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample, "Llama: Open and efficient foundation language models," *CoRR*, vol. abs/2302.13971, 2023.

[20] L. Team, "The llama 3 herd of models," *CoRR*, vol. abs/2407.21783, 2024.

[21] S. Merity, C. Xiong, J. Bradbury, and R. Socher, "Pointer sentinel mixture models," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. [Online]. Available: https://openreview.net/forum?id=Byj72udxe

[22] Q. Zhang, M. Chen, A. Bukharin, P. He, Y. Cheng, W. Chen, and T. Zhao, "Adaptive budget allocation for parameter-efficient fine-tuning," in *ICLR*. OpenReview.net, 2023.

[23] V. Bergholm, J. A. Izaac, M. Schuld, C. Gogolin, and N. Killoran, "Pennylane: Automatic differentiation of hybrid quantum-classical computations," *CoRR*, vol. abs/1811.04968, 2018.

[24] L. Domingo, G. Carlo, and F. Borondo, "Taking advantage of noise in quantum reservoir computing," *Scientific Reports*, vol. 13, no. 1, p. 8790, 2023.

[25] C. Wu, F. Wu, T. Qi, and Y. Huang, "Noisytune: A little noise can help you finetune pretrained language models better," in *ACL (2)*. Association for Computational Linguistics, 2022, pp. 680–685.