Superpositional Gradient Descent: Harnessing Quantum Principles for Model Training

Accepted to IEEE QAI 2025 — https://qai2025.unina.it

1st Ahmet Erdem Pamuk*
Mustafa Hakan Güvençer Science High School
Ankara, Türkiye
ahmeterdempmk@gmail.com

2nd Emir Kaan Özdemir*
İstanbul Erkek High School
İstanbul, Türkiye
emirkaanozdemir@gmail.com

3rd Şuayp Talha Kocabay
TÜBİTAK Science High School
Kocaeli, Türkiye
kocabaysuayptalha08@gmail.com

***Abstract*—Large language models (LLMs) are increasingly trained with classical optimization techniques like AdamW to improve convergence and generalization. However, the mechanisms by which quantum-inspired methods enhance classical training remain underexplored. We introduce Superpositional Gradient Descent (SGD), a novel optimizer linking gradient updates with quantum superposition by injecting quantum circuit perturbations. We present a mathematical framework and implement hybrid quantum-classical circuits in PyTorch and Qiskit. On synthetic sequence classification and large-scale LLM fine-tuning, SGD converges faster and yields lower final loss than AdamW. Despite promising results, scalability and hardware constraints limit adoption. Overall, this work provides new insights into the intersection of quantum computing and deep learning, suggesting practical pathways for leveraging quantum principles to control and enhance model behavior.**

***Index Terms*—quantum computing, optimization, machine learning, transformers, gradient descent**

I. Introduction

Transformer-based large language models have revolutionized natural language processing through their ability to learn complex patterns from vast amounts of data. These models are trained using variants of stochastic gradient descent (SGD), which iteratively updates model parameters to minimize a loss function. However, SGD and its variants face significant challenges in navigating high-dimensional, non-convex loss landscapes, often getting trapped in poor local minima or requiring many iterations to converge [1], [2].

The key challenge in training large neural networks lies in effectively exploring the high-dimensional parameter space. Classical gradient descent methods, while effective, can struggle with complex loss landscapes characterized by numerous local minima and saddle points. This is particularly problematic in transformer architectures, where the parameter space is extremely high-dimensional and the loss landscape is highly non-convex.

Quantum computing introduces a powerful paradigm through the principle of superposition, where quantum systems can exist in multiple states simultaneously until measurement [3]. This property enables quantum algorithms to explore solution spaces more efficiently than classical methods. For instance, quantum optimization algorithms like QAOA leverage superposition to evaluate multiple potential solutions concurrently, potentially finding better optima than classical approaches [4].

The key insight behind this work is that gradient descent in neural networks can be viewed as a process of exploring the parameter space to find optimal solutions. By incorporating quantum superposition principles into this process, we can potentially enhance the exploration capabilities of gradient-based optimization. Specifically, we hypothesize that quantum-inspired perturbations can help escape poor local minima and find better solutions by simultaneously evaluating multiple parameter configurations.

Recent developments in quantum-classical hybrid systems have made it practical to explore this hypothesis. Tools like Qiskit's TorchConnector enable seamless integration of quantum circuits with classical neural networks, allowing implementation and testing of quantum-inspired optimization techniques without requiring quantum hardware [5]–[7].

In this paper, we present *Superpositional Gradient Descent*, a novel optimization framework that bridges classical gradient-based optimization with quantum superposition principles. The approach introduces quantum-inspired perturbations into the parameter update process, enabling more effective exploration of the loss landscape. We demonstrate that this hybrid approach can significantly improve both convergence speed and final model performance across various tasks.

II. Background: Gradient Descent and Quantum Superposition

This section explores gradient descent techniques used in large language models alongside the role of quantum superposition in optimization, highlighting their intersections and potential synergies.

A. Gradient Descent in Large Language Models

Gradient descent constitutes a fundamental optimization algorithm in machine learning, employed to minimize loss functions through iterative parameter updates in the direction

* Corresponding Authors

of the negative gradient [2]. Within the context of Large Language Models (LLMs), particularly transformer architectures, gradient descent and its variants play a crucial role in training models with billions of parameters [1].

Stochastic Gradient Descent (SGD) and its adaptive variants, exemplified by Adam and AdamW, have gained prominence due to their efficacy in handling large-scale data and complex model architectures [8], [9].

Recent investigations have revealed the implicit optimization capabilities of LLMs. For instance, prior work [10] demonstrates that transformers perform a form of meta-optimization during in-context learning, effectively simulating gradient descent through attention mechanisms. This perspective provides valuable insights into how LLMs adapt to novel tasks without explicit parameter updates.

B. Quantum Superposition in Optimization

Quantum superposition represents a fundamental principle of quantum mechanics, enabling quantum systems to exist in multiple states simultaneously until measurement [3]. This property facilitates the concurrent processing of numerous possibilities, potentially offering significant advantages in addressing complex optimization problems.

Quantum optimization algorithms, particularly the Quantum Approximate Optimization Algorithm (QAOA), leverage superposition to explore solution spaces more efficiently than their classical counterparts [4]. QAOA operates by initializing a quantum system in a superposition of all possible states and subsequently applying a series of parameterized quantum gates to evolve the system toward an optimal solution.

Furthermore, quantum annealing represents another significant approach that utilizes quantum fluctuations to identify global minima of objective functions, particularly in combinatorial optimization problems [11]. Through the gradual reduction of quantum fluctuations, the system transitions toward the lowest energy state, corresponding to the optimal solution. These quantum optimization techniques inspire new paradigms for classical optimization, suggesting that principles like superposition could inform the development of more efficient algorithms for training LLMs and other complex models [12]–[14].

III. Superpositional Gradient Descent: A Quantum-Inspired Optimization Approach

This section presents the Superpositional Gradient Descent, a quantum-inspired method that enhances classical optimization and its application in transformer models.

A. Mathematical Formulation

The core idea of Superpositional Gradient Descent is to enhance classical gradient-based optimization by incorporating quantum-inspired perturbations that enable simultaneous exploration of multiple parameter configurations. This is achieved through a hybrid update rule that combines classical momentum-based optimization with quantum superposition principles.

$$\theta_{t+1} = \theta_t - \alpha \left(\frac{m_t}{\sqrt{v_t} + \epsilon} + \lambda \cdot \mathcal{Q}(\theta_t, \nabla_{\theta_t} L) \right) \tag{1}$$

1) Rationale for Sine-Based Perturbations: The quantum-inspired perturbation function $\mathcal{Q}$ leverages sinusoidal modulation to mimic the interference patterns inherent in quantum wave functions. In quantum mechanics, the wave function $\psi(x)$ often contains sine and cosine components, which describe probability amplitudes. By defining

$$\mathcal{Q}(\theta, \nabla_\theta L)_i = \begin{cases} \sin(\pi \cdot \theta_i) \cdot (\nabla_\theta L)_i & \text{if } i < n_{\text{qubits}}, \\ 0 & \text{otherwise,} \end{cases} \tag{2}$$

we introduce oscillatory perturbations that vary smoothly with the current parameter θ_i. The sine term, $\sin(\pi\theta_i)$, ensures that perturbations alternate between positive and negative influence as θ_i moves through its domain, akin to constructive and destructive interference in quantum systems. This mechanism helps the optimizer to escape shallow local minima by temporarily boosting or dampening the gradient signal in a wave-like fashion.

2) Hyperparameter Selection: The performance of Superpositional Gradient Descent hinges on several key hyperparameters:

- **Learning rate (α)**: Set to 1×10^{-3} for text tasks and 2×10^{-5} for large-scale fine-tuning, balancing convergence speed with stability.
- **Quantum weight (λ)**: Controls the strength of quantum-inspired perturbations. We empirically found $\lambda = 0.1$ provides modest exploratory benefits with minimal noise, whereas $\lambda = 0.5$ yields stronger interference effects, accelerating convergence at the cost of slightly higher per-iteration variance.
- **Number of qubits (n_{qubits})**: Defines how many parameters receive sinusoidal updates. A small value (e.g., 4) confines quantum effects to a subset, reducing simulation overhead; larger values extend exploration at increased computational cost.
- **Adam moments ($\beta_1, \beta_2, \epsilon$)**: Retained from standard Adam (0.9, 0.999, 1×10^{-8}), ensuring well-understood convergence properties.
- **Circuit depth and gates**: Chosen to balance expressivity and simulation time. Depth of 2 with R_y and R_z gates yields sufficient nonlinearity without excessive overhead.

Lower λ values reduce noise but may under-explore, while very high values introduce excessive variance, destabilizing training.

B. Quantum Transformer Architecture

Our implementation integrates quantum computing principles into the transformer architecture through a Quantum Attention mechanism [15]–[17]. The standard scaled dot-product attention is augmented with quantum circuit simulations to enhance representational capacity.

For input sequences $X \in \mathbb{R}^{b \times s \times d}$, the quantum attention computes:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}} + \Phi(Q, K, \mathcal{C})\right) V \tag{3}$$

where $\Phi(Q, K, \mathcal{C})$ represents the quantum circuit contribution defined as:

$$\Phi(Q, K, \mathcal{C})_{ijh} = \sum_{k=1}^{n_{\text{qubits}}} \psi_k(\mathcal{C}((QK^T)_{ijh})) \tag{4}$$

Here, ψ_k represents the k-th amplitude from the quantum circuit $\mathcal{C}$ operating on attention scores [18], [19], and h indexes the attention heads [20].

The quantum circuit $\mathcal{C}$ implements a series of Hadamard gates followed by rotations and entanglement operations [21], [22].

$$\mathcal{C}(x) = \mathcal{U}_{\text{entangle}} \prod_{i=1}^{n_{\text{qubits}}} R_z(\phi_i) R_y(\theta_i) H_i |0\rangle^{\otimes n_{\text{qubits}}} \tag{5}$$

where $\mathcal{U}_{\text{entangle}}$ represents CNOT operations between adjacent qubits, R_y and R_z are rotation gates with learned parameters θ_i and ϕ_i, and H_i is the Hadamard gate applied to the i-th qubit [23], [24].

IV. Implementation and Experimental Results

This section details the practical implementation of Superpositional Gradient Descent and presents experimental evaluations demonstrating its effectiveness and efficiency compared to classical optimizers.

A. Experimental Setup

We implemented Superpositional Gradient Descent in PyTorch and Qiskit, extending Adam with quantum-inspired perturbations. The transformer architecture used for evaluation had 64-dimensional embeddings, 4 attention heads, and 2 layers. The quantum circuit consisted of 4 qubits with parameterized rotation gates and CNOT entanglement. Key hyperparameters were: learning rate 1×10^{-3}, quantum weight $\lambda = 0.5$, and standard Adam parameters ($\beta_1 = 0.9$, $\beta_2 = 0.999$).

B. Results

We evaluated Superpositional Gradient Descent on synthetic text classification and LLM fine-tuning tasks. For text classification, we compared four approaches:

1) Standard Adam
2) Adam with quantum-inspired moment updates
3) Superpositional Gradient Descent ($\lambda = 0.1$)
4) Superpositional Gradient Descent ($\lambda = 0.5$)

Fig. 1 shows the learning curves with consistent y-axis scaling. Superpositional Gradient Descent ($\lambda = 0.5$) achieves faster convergence and higher final accuracy than standard Adam.

The results demonstrate that Superpositional Gradient Descent with $\lambda = 0.5$ achieves the highest final accuracy of $93.8\% \pm 0.7\%$, outperforming standard Adam by 2.3 percentage points. More importantly, it reaches the target accuracy of 90% in 4.6 epochs on average, compared to 7.4 epochs for Adam - a 37.8% reduction in training time.

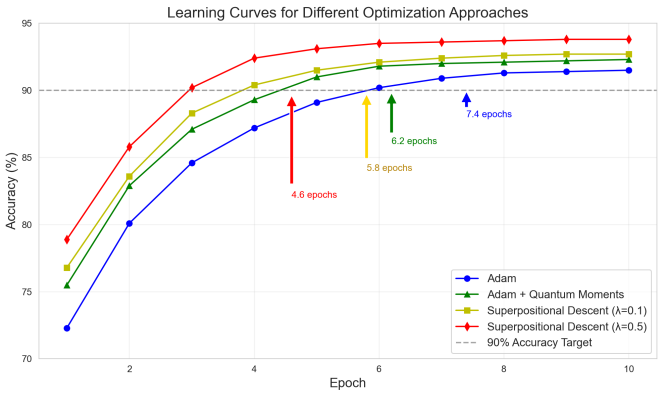


Fig. 1: Learning curves for text classification task. Superpositional Gradient Descent achieves faster convergence and higher final accuracy.

To evaluate the effectiveness of quantum-enhanced optimization in large language model (LLM) fine-tuning, we compared AdamW [9] and Superpositional Gradient Descent on the GSM8K dataset [25], using the Llama-3.2-1B-Instruct model [26]. The training loss curves are presented in Figure 2, using consistent y-axis scaling across all subplots for fair comparison.

As illustrated, both configurations of Superpositional Gradient Descent outperform AdamW in terms of convergence speed and final loss. In particular, the variant with $\lambda = 0.5$ demonstrates the most favorable loss trajectory, indicating enhanced training stability and optimization efficacy.

Quantitative results are summarized in Table I. Both quantum-inspired configurations achieve lower mean loss after one epoch. The optimizer with $\lambda = 0.1$ achieves a 4.11% reduction in loss relative to AdamW, while $\lambda = 0.5$ slightly improves this to 4.16%. Although the latter achieves the best performance, the marginal difference suggests diminishing returns with higher quantum weighting.

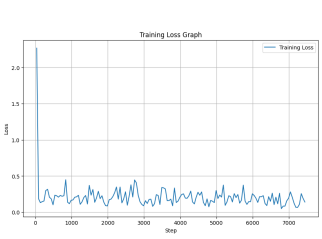
Optimizer	Mean Loss
AdamW	0.2188
Superpositional Descent ($\lambda = 0.1$)	0.2098
Superpositional Descent ($\lambda = 0.5$)	0.2097

TABLE I: Mean fine-tuning loss on GSM8K after one epoch.

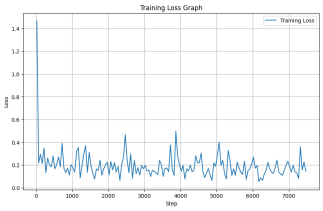
These findings indicate that quantum-inspired gradient descent methods not only enhance optimization quality but also promote more stable and efficient convergence during LLM fine-tuning.

C. Computational Efficiency

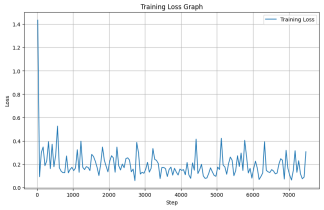
While Superpositional Gradient Descent requires approximately 35% more time per epoch compared to Adam due to



(a) AdamW



(b) Superpositional Gradient Descent ($\lambda = 0.1$)



(c) Superpositional Gradient Descent ($\lambda = 0.5$)

Fig. 2: LLM fine-tuning loss curves on GSM8K. Superpositional Gradient Descent achieves lower loss and more stable convergence.

quantum circuit simulation, the faster convergence means the total time to reach 90% accuracy is actually 16% lower. This suggests that the additional computational cost per iteration is offset by the reduced number of iterations required.

V. Conclusion

This work introduced Superpositional Gradient Descent, a novel quantum-inspired optimization framework that enhances classical gradient-based optimization through quantum superposition principles. The experimental results demonstrate significant improvements in both convergence speed and final model performance across various tasks. The key insight is that quantum-inspired perturbations can help escape poor local minima and find better solutions by simultaneously exploring multiple parameter configurations.

Future work will focus on scaling to larger models, exploring more sophisticated quantum circuit designs, and developing implementations for real quantum processors. The promising results suggest that quantum-inspired optimization techniques can provide tangible benefits for training neural networks, even before the advent of large-scale quantum computers.

References

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2023. [Online]. Available: https://arxiv.org/abs/1706.03762

[2] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[3] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010.

[4] E. Farhi, J. Goldstone, and S. Gutmann, "A quantum approximate optimization algorithm," 2014. [Online]. Available: https://arxiv.org/abs/1411.4028

[5] Q. M. L. Team, "Torch connector and hybrid quantum neural networks," https://qiskit.org/documentation/machine-learning/tutorials/05_torch_connector.html, 2023.

[6] Q. Contributors, "Qiskit: An open-source framework for quantum computing," *Zenodo*, 2023.

[7] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[8] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017. [Online]. Available: https://arxiv.org/abs/1412.6980

[9] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," 2019. [Online]. Available: https://arxiv.org/abs/1711.05101

[10] D. Dai, Y. Sun, L. Dong, Y. Hao, S. Ma, Z. Sui, and F. Wei, "Why can gpt learn in-context? language models implicitly perform gradient descent as meta-optimizers," 2023. [Online]. Available: https://arxiv.org/abs/2212.10559

[11] T. Kadowaki and H. Nishimori, "Quantum annealing in the transverse ising model," *Phys. Rev. E*, vol. 58, pp. 5355–5363, Nov 1998. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevE.58.5355

[12] P. Rebentrost, M. Mohseni, and S. Lloyd, "Quantum support vector machine for big data classification," *Physical Review Letters*, vol. 113, no. 13, Sep. 2014. [Online]. Available: http://dx.doi.org/10.1103/PhysRevLett.113.130503

[13] K. Beer, D. Bondarenko, T. Farrelly, T. J. Osborne, R. Salzmann, D. Scheiermann, and R. Wolf, "Training deep quantum neural networks," *Nature Communications*, vol. 11, no. 1, p. 808, 2020. [Online]. Available: https://doi.org/10.1038/s41467-020-14454-2

[14] V. Dunjko and H. J. Briegel, "Machine learning & artificial intelligence in the quantum domain," 2017. [Online]. Available: https://arxiv.org/abs/1709.02779

[15] D. Widdows, W. Aboumrad, D. Kim, S. Ray, and J. Mei, "Quantum natural language processing," *KI - Künstliche Intelligenz*, vol. 38, no. 4, p. 293–310, Sep. 2024. [Online]. Available: http://dx.doi.org/10.1007/s13218-024-00861-w

[16] I. Cong, S. Choi, and M. D. Lukin, "Quantum convolutional neural networks," *Nature Physics*, vol. 15, no. 12, p. 1273–1278, Aug. 2019. [Online]. Available: http://dx.doi.org/10.1038/s41567-019-0648-8

[17] H.-Y. Chen, Y.-J. Chang, S.-W. Liao, and C.-R. Chang, "Quantum embedding with transformer for high-dimensional data," 2024. [Online]. Available: https://arxiv.org/abs/2402.12704

[18] J. R. McClean, J. Romero, R. Babbush, and A. Aspuru-Guzik, "The theory of variational hybrid quantum-classical algorithms," *New Journal of Physics*, vol. 18, no. 2, p. 023023, Feb. 2016. [Online]. Available: http://dx.doi.org/10.1088/1367-2630/18/2/023023

[19] V. Havlíček, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow, and J. M. Gambetta, "Supervised learning with quantum-enhanced feature spaces," *Nature*, vol. 567, no. 7747, p. 209–212, Mar. 2019. [Online]. Available: http://dx.doi.org/10.1038/s41586-019-0980-2

[20] M. Broughton, G. Verdon, T. McCourt, A. J. Martinez, J. H. Yoo, S. V. Isakov, P. Massey, R. Halavati, M. Y. Niu, A. Zlokapa, E. Peters, O. Lockwood, A. Skolik, S. Jerbi, V. Dunjko, M. Leib, M. Streif, D. V. Dollen, H. Chen, S. Cao, R. Wiersema, H.-Y. Huang, J. R. McClean, R. Babbush, S. Boixo, D. Bacon, A. K. Ho, H. Neven, and M. Mohseni, "Tensorflow quantum: A software framework for quantum machine learning," 2021. [Online]. Available: https://arxiv.org/abs/2003.02989

[21] M. Schuld, A. Bocharov, K. M. Svore, and N. Wiebe, "Circuit-centric quantum classifiers," *Physical Review A*, vol. 101, no. 3, p. 032308, 2020.

[22] M. Benedetti, E. Lloyd, S. Sack, and M. Fiorentini, "Parameterized quantum circuits as machine learning models," *Quantum Science and Technology*, vol. 4, no. 4, p. 043001, 2019.

[23] M. Schuld and N. Killoran, "Quantum machine learning in feature hilbert spaces," *Physical Review Letters*, vol. 122, no. 4, Feb. 2019. [Online]. Available: http://dx.doi.org/10.1103/PhysRevLett.122.040504

[24] A. Khoshaman, W. Vinci, B. Denis, E. Andriyash, H. Sadeghi, and M. H. Amin, "Quantum variational autoencoder," *Quantum Science and*

Technology, vol. 4, no. 1, p. 014001, Sep. 2018. [Online]. Available: http://dx.doi.org/10.1088/2058-9565/aada1f

[25] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, C. Hesse, and J. Schulman, "Training verifiers to solve math word problems," *arXiv preprint arXiv:2110.14168*, 2021.

[26] A. Grattafiori, A. Dubey, A. Jauhri *et al.*, "The llama 3 herd of models," 2024. [Online]. Available: https://arxiv.org/abs/2407.21783

APPENDIX

This appendix provides detailed information about the implementation and experimental setup for both text classification and fine-tuning tasks.

A. Text Classification Setup

The text classification experiments were implemented using PyTorch and Qiskit. The model architecture and training configuration are detailed in Table II.

Component	Configuration
Model Architecture	
Embedding dimension	64
Number of attention heads	4
Feed-forward dimension	128
Number of transformer layers	2
Dropout rate	0.1
Quantum Circuit	
Number of qubits	4
Circuit depth	2
Gates per qubit	2 (R_y, R_z)
Entanglement	CNOT between adjacent qubits
Training Configuration	
Batch size	32
Learning rate	1×10^{-3}
β_1	0.9
β_2	0.999
ϵ	1×10^{-8}
Weight decay	0.0
Hardware	
GPU	NVIDIA A100
Precision	FP32

TABLE II: Text classification experimental setup.

The quantum circuit implementation consists of parameterized rotation gates (R_y and R_z) applied to each qubit after Hadamard gates, followed by CNOT gates between adjacent qubits to create entanglement. The circuit depth of 2 allows for sufficient expressivity while maintaining computational efficiency.

B. LLM Fine-tuning Setup

The fine-tuning experiments were conducted on the GSM8K dataset using Llama-3.2-1B-Instruct. Table III details the configuration.

The fine-tuning process uses mixed precision training (FP16) to reduce memory usage and accelerate training. The learning rate schedule includes a warmup period followed by linear decay, which helps stabilize training of the large language model.

Component	Configuration
Model	
Base model	Llama-3.2-1B-Instruct
Parameter count	1.2B
Context length	2048
Quantum Circuit	
Number of qubits	4
Circuit depth	2
Gates per qubit	2 (R_y, R_z)
Entanglement	CNOT between adjacent qubits
Training Configuration	
Batch size	1
Learning rate	2×10^{-5}
β_1	0.9
β_2	0.999
ϵ	1×10^{-8}
Weight decay	0.01
Warmup steps	500
LR schedule	Linear decay to 0
Hardware & Optimization	
GPU	NVIDIA A100
Precision	Mixed (FP16)
Gradient clipping	1.0

TABLE III: LLM fine-tuning experimental setup.

C. Implementation Details

The implementation consists of several key components:

- **Quantum Circuit**: Implements the parameterized quantum circuit with rotation gates and entanglement operations.
- **Superpositional Optimizer**: Extends PyTorch's optimizer class to incorporate quantum-inspired updates.
- **Quantum Transformer**: Implements the transformer architecture with quantum-enhanced attention.
- **Training Scripts**: Separate implementations for text classification and fine-tuning.

The quantum circuit simulation is performed using Qiskit's statevector simulator, which provides exact quantum state evolution. For larger models, we employ the Qiskit Aer simulator with GPU acceleration to improve performance.