

Classical neural networks on quantum devices via tensor network disentanglers: A case study in image classification

Borja Aizpurua^{1,2,*}, Sukhbinder Singh³, and Román Orús^{1,4,5}

¹*Multiverse Computing, San Sebastián, Spain*

²*Department of Basic Sciences, Tecnun—University of Navarra, San Sebastián, Spain*

³*Multiverse Computing, Toronto, Ontario, Canada*

⁴*Donostia International Physics Center, San Sebastián, Spain*

⁵*Ikerbasque Foundation for Science, Bilbao, Spain*



(Received 18 September 2025; accepted 12 March 2026; published 8 April 2026)

We address the problem of implementing bottleneck layers from classical pretrained neural networks on a quantum computer, with the goal of exploring intrinsically quantum ansatz for representing large linear layers within hybrid classical-quantum models. Our approach begins with a compression step in which the target linear layer is represented as an effective matrix product operator (MPO) without degrading model performance. The MPO is then further disentangled into a more compact form. This enables a hybrid classical-quantum execution scheme, where the disentangling circuits are deployed on a quantum computer while the remainder of the network—including the disentangled MPO—runs on classical hardware. We introduce two complementary algorithms for MPO disentangling: (1) an *explicitly* disentangling variational method leveraging standard tensor network optimization techniques and (2) an *implicitly* disentangling gradient-descent-based approach. We validate these methods through a proof-of-concept translation of simple classical neural networks for MNIST and CIFAR-10 image classification into a hybrid classical-quantum form.

DOI: [10.1103/fmps-tjwy](https://doi.org/10.1103/fmps-tjwy)

I. INTRODUCTION

The rapid improvement of quantum hardware has intensified the search for useful applications on present-day and near-term quantum devices. In parallel, advances in artificial intelligence have motivated growing interest in leveraging quantum computation to enhance modern machine learning methodologies. Research at this intersection has largely followed two complementary directions [1]. The first develops hybrid classical-quantum algorithms—including the variational quantum eigensolver [2,3], the quantum approximate optimization algorithm [4,5], and quantum variants of classical machine learning (ML) methods such as quantum Boltzmann machines [6] and variational quantum classifiers [7]. The second explores purely quantum machine learning algorithms, such as quantum k -means clustering [8], quantum principal component analysis [9], the quantum linear systems algorithm [10], and quantum circuit Born machines [11].

Both approaches face significant limitations. In hybrid models, quantum circuits are often too shallow to capture meaningful structure [12,13], or too deep to train effectively on noisy hardware [14]. They may also lack the inductive

biases crucial for specific ML tasks, and even when appropriately scaled, training can be hindered by barren plateaus [15,16]. Purely quantum methods encounter similar scalability challenges, with current demonstrations restricted to small problem sizes.

In this work, we investigate how quantum computation can be integrated into practical, large-scale classical ML workflows. Specifically, we focus on embedding a bottleneck segment of a classically pretrained deep neural network into a quantum circuit, with the aim of improving performance or efficiency. Modern deep networks concentrate much of their parameter count in large linear layers, each represented by a weight matrix. Concretely, given a network $\mathcal{N}$, we target a large bottleneck linear layer represented by matrix W with dimension 2^ℓ , $\ell \in \mathbb{N}$, and implement it on a quantum processor by means of quantum circuits for ℓ qubits.

Inference proceeds in a hybrid fashion: Classical data flow through the neural network until reaching the designated quantum layer. The output activations of the preceding classical layer are encoded into a quantum state—via amplitude encoding or by approximating them as a matrix product state (MPS) in canonical form, which is prepared on the quantum computer [17]—and processed by the pretrained quantum circuit (Fig. 1). The circuit's outputs are measured and converted back into classical activations through measurement and classical postprocessing (full state tomography in our proof-of-concept simulations), and passed to the remainder of the classical network $\mathcal{N}$ for standard inference.

The central challenge in our approach is to translate the weight matrix W into a quantum circuit of manageable

*Contact author: borja.aizpurua@multiversecomputing.com

Published by the American Physical Society under the terms of the [Creative Commons Attribution 4.0 International](https://creativecommons.org/licenses/by/4.0/) license. Further distribution of this work must maintain attribution to the author(s) and the published article's title, journal citation, and DOI.

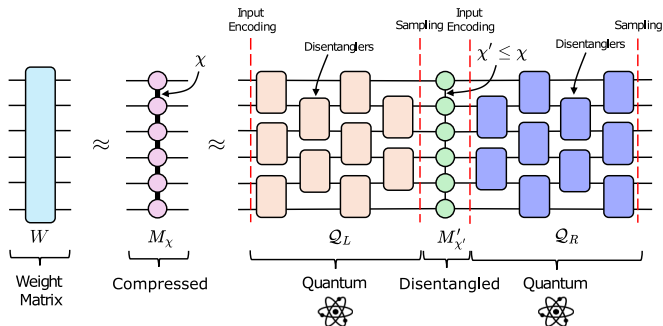


FIG. 1. Schematic of the hybrid quantum-classical model architecture. A classical image is encoded into a quantum state and transformed via a disentangling circuit Q_L . A compressed MPO M' , acts on this state, followed by another quantum circuit Q_R . Both Q_L and Q_R are implemented as variational quantum circuits composed of two-qubit gates. The final state is sampled or reconstructed to yield a classical vector, completing the inference pipeline.

complexity. A straightforward method involves selecting a circuit ansatz and optimizing it—via classical variational or gradient-descent techniques—to approximate W . However, as discussed below, this optimization can become prohibitively expensive, with costs scaling exponentially in circuit depth, especially when W is represented as a dense matrix.

To address this, we propose a two-step strategy to better manage computational overhead. First, we perform a weight-compression step by approximating W as a *matrix product operator* (MPO) M_χ , where χ denotes the maximum bond dimension in the tensor network. Crucially, this replacement preserves the network’s performance within acceptable bounds. MPO-based compression of linear layers has been demonstrated as an effective model compression technique [18,19], including applications in contemporary large language models [20].

Having obtained the optimized MPO M approximating W , we then apply a disentangling step: We approximate M by quantum circuits such that

$$M \approx Q_L M' Q_R,$$

where M' is a more compact MPO with reduced bond dimension $\chi' < \chi$, and Q_L, Q_R are quantum circuits. We envision these circuits, transpiled to hardware-level implementations, running on a quantum processor, while the compressed MPO M' remains integrated within the classical neural network.

We emphasize that the primary aim of this paper is not to demonstrate a practical quantum advantage for inference on deep learning models, but to initiate the development of concrete algorithms realizing this hybrid classical-quantum framework—one that could eventually enable scaling regimes beyond what is accessible with purely classical tensor network representations. Here, we demonstrate a compression technique in which classical parameters in a pretrained model are effectively and efficiently replaced—via a disentangling procedure—with parameters encoded in a quantum circuit. This, in turn, suggests that incorporating quantum circuits into a classical model can enhance its expressivity.

Accordingly, we do not claim quantum advantage in computational cost or inference speed, but rather study

expressivity as a distinct scaling resource enabled by intrinsically quantum circuit ansatz.

Recently, we outlined this proposal in the context of large language models [21]. Here, we focus on a concrete proof-of-concept study involving image classification on the MNIST [22,23] and CIFAR-10 [24] datasets, using elementary neural architectures such as multilayer perceptrons and their MPO-tensorized counterparts. Even in this simplified setting, we identify significant challenges *en route* to practical quantum advantage—mainly, the rapid growth in circuit depth after transpilation to hardware—and propose several avenues for improving the baseline methods presented here.

II. MPO COMPRESSION OF THE WEIGHT MATRIX

The first step is to compress the weight matrix W into an MPO M . While other tensor network architectures are possible, we focus on MPOs due to their demonstrated effectiveness in compressing large neural networks [18,19], including large language models [20]. It is important to note that MPO M is not simply a low-rank approximation of W . Empirically, typical pretrained weight matrices exhibit near-maximal MPO bond dimensions, as expected: Gradient-descent optimization—the standard training procedure—does not inherently bias weights toward low-rank subspaces unless such bias is explicitly enforced by the architecture or training algorithm.

Indeed, we find that naively replacing the weight matrix with a low-rank MPO approximation significantly degrades model performance. Instead, after substituting W with a low-rank MPO approximation, the updated model must be healed—namely, retrained or fine-tuned—to recover its original accuracy. Once the training loss converges and the model is sufficiently healed, we obtain the optimal MPO M corresponding to the original weight W . With M fixed, we can then proceed to the disentangling step. However, before doing so, it is necessary to select an ansatz for the disentangling circuits.

III. FIXING A DISENTANGLING CIRCUIT ANSATZ

Several design choices must be made to fix the structure of the disentangling circuits before gate optimization. Ideally, we seek to completely disentangle the MPO, achieving $\chi' = 1$. In general, this corresponds to a disentangling quantum channel (superoperator) Φ defined via $\Phi(MM^\dagger) = \bigotimes_i M_i^\circ$, where $\{M_i^\circ\}$ are positive matrices forming a completely disentangled MPO M° (or possibly its square root).

From the Kraus decomposition [25]

$$\Phi(\cdot) = \sum_k U_k(\cdot)U_k^\dagger, \quad (1)$$

we recover a set of disentangling unitaries $\{U_k\}$. Each U_k can be approximated by a quantum circuit $\mathcal{U}_k$, yielding an ensemble of disentangling circuits. In this work, we restrict attention to a single quantum circuit in the Kraus ensemble by fixing $k = 1$ and setting $Q_L = \mathcal{U}_1$. We further allow for a distinct adjoint circuit $Q_R \neq Q_L^\dagger$.

Following these choices, the geometry, sparsity, and gate set of the disentangling circuits must be specified. We primarily adopt a two-body brickwall architecture, possibly

interspersed with layers of single-qubit gates. (Some of our experiments to draw comparisons extend to higher-body gates, up to six-body interactions.)

For simplicity and numerical stability, we restrict ourselves to real orthogonal disentanglers in this work. This restriction is a deliberate modeling choice rather than a technical limitation: While modern automatic differentiation frameworks support complex-valued parameters and extending our approach to complex unitaries is straightforward, real-valued circuits define a smaller variational class that trades expressivity for potentially improving numerical stability and reducing contraction cost. For instance, recent results show that real-valued tensor networks composed predominantly of non-negative entries can be contracted much more efficiently than generic real-valued tensor networks, demonstrating that the tensor data type can significantly affect contraction cost [26]. We expect complex-valued disentanglers to increase expressivity in general, but leave a systematic comparison between real and complex circuits to future work.

A key practical consideration in choosing the circuit ansatz is minimizing the size and complexity of the *physical* circuits resulting from transpilation to target quantum hardware. Shallow logical circuits composed of two-body random unitaries often transpile into very deep physical circuits with potentially nonlocal gate placements, thereby increasing execution costs on quantum devices. To mitigate this, we fix some gates in the disentangling circuits to elementary gates native to the hardware. Specifically, in this work, we fix some or all two-body gates to cnot gates. This approach potentially sacrifices some disentangling power but significantly reduces transpilation complexity.

As an extreme choice, we fix all two-body disentanglers to CNOTS arranged in a brickwall pattern and optimize only the single-qubit gates to maximize disentangling. Remarkably, as we demonstrate, this constrained ansatz still achieves strong model performance, indicating that variational single-qubit gates applied on a frozen scaffold of CNOTS provide substantial disentangling capability while lowering transpilation overhead.

IV. EXPLICIT DISENTANGLERS VIA VARIATIONAL OPTIMIZATION

One approach to find the disentanglers by maximizing the overlap

$$\text{Tr}(M_\chi(Q_L M'_\chi Q_R)) / (\|M_\chi\| \|M'_\chi\|), \quad (2)$$

similar to technique used to determine disentanglers in the multiscale entanglement renormalization ansatz [27,28].

All disentangling gates are initialized randomly and then iteratively optimized. At each iteration, we compute the environment tensor $\mathcal{E}_g$ for each gate g . Concretely, assuming g belongs to the left circuit Q_L , the environment is given by

$$\mathcal{E}_g \equiv \text{Tr}(M_\chi(Q_L^{g \rightarrow \text{hole}} M'_\chi Q_R)) / (\|M_\chi\| \|M'_\chi\|), \quad (3)$$

where $Q_L^{g \rightarrow \text{hole}}$ denotes the circuit obtained by replacing gate g with an empty hole (i.e., a slot inside the quantum circuit with open indices, where g plugs in), as illustrated in Fig. 2.

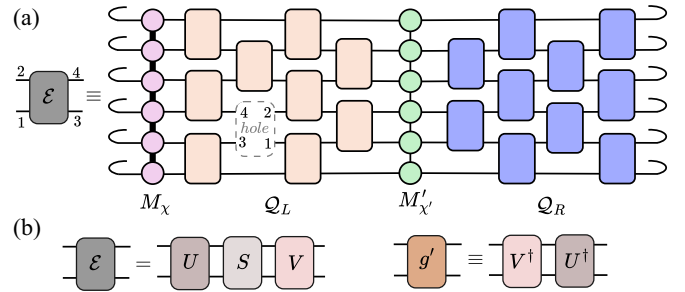


FIG. 2. Variational optimization of disentanglers. (a) The environment tensor $\mathcal{E}_g$, Eq. (3), corresponding to a gate g being optimized is obtained by removing that tensor from the circuit, leaving a hole with open indices, and contracting the remaining tensors that appear in Eq. (3). (b) The singular value decomposition of $\mathcal{E}_g$ is performed, $\mathcal{E}_g = U S V$. The updated gate is obtained as $g' = V^\dagger U^\dagger$.

We note that the environment tensor $\mathcal{E}_g$ plays the role of the local linear functional governing the variation of the overlap objective with respect to the entries of the gate g . In this sense, $\mathcal{E}_g$ is closely related to the gradient of the cost function with respect to g , and for the circuit sizes considered in our proof-of-concept experiments, it could equivalently be obtained via automatic differentiation (AD) through the full contraction graph. However, the use of AD in this context does not circumvent the dominant computational cost of contracting the full tensor network defining the overlap objective (as discussed below). Nonetheless, we also explored gradient-based explicit disentangling and found that environment-based variational updates converge faster and more robustly, as they exploit the local structure of the optimization problem rather than relying on repeated full-network gradient evaluations.

Computing the environments $\mathcal{E}_g$ requires contracting the tensor network formed by the circuits and the MPOs, equating to the trace on the right-hand side of Eq. (3). The gate g is then updated via singular value decomposition (SVD) of its environment, $\mathcal{E}_g = U S V$, by setting

$$g' \equiv V^\dagger U^\dagger. \quad (4)$$

In practice, we find that large weight matrices from pre-trained networks often translate into very deep circuits under this approach. Consequently, the cost of computing the exact environment tensors $\{\mathcal{E}_g\}$ grows rapidly with circuit depth, making the variational optimization computationally challenging for large circuits.

For the shallow circuits used in the numerical experiments presented here (in particular, MPOs defined over six qubits and modest circuit depth), exact contraction of the environment tensors is fully feasible and does not pose a computational bottleneck. Our discussion of approximate contraction strategies is included to clarify how the method would scale to deeper disentangling circuits, where contraction cost is governed primarily by circuit depth and entangling structure rather than the number of qubits alone.

However, for large circuits, these environments can be approximated using standard tensor network techniques such as boundary MPS contractions or renormalization group methods [29,30]. However, the efficiency of these approaches

depends heavily on the circuit's structure, sparsity, and entangling power.

We note that a similar variational procedure could be applied directly to approximate the dense weight matrix W using quantum circuits. For example, one could first decompose W via a QR decomposition, $W = QR$, where Q is unitary, or via a singular value decomposition, $W = USV$, where U and V are unitary matrices. The variational optimization would then target only the unitary components Q or U, V , while the nonunitary components remain in the classical network.

Our motivation for not adopting this direct approach is that we expect the disentangling circuits for MPOs with small bond dimension—those typically obtained after the weight-compression step—to be substantially less entangling and more structured than those derived directly from dense matrices. Consequently, by first compressing the dense weight into an MPO, we anticipate a significant reduction in the overall computational cost.

A. Numerical results

To practically demonstrate the variational optimization of disentanglers, we begin with a simple proof-of-principle experiment. In this setting, we skip the MPO compression stage—necessary when working with pretrained dense networks—and instead train a compact tensor network neural network (TNN) from scratch for MNIST classification. Our lightweight model, which attains a test accuracy of 92.6%, serves as a convenient proxy for the compressed network in our classical-quantum workflow, allowing us to focus directly on the disentangler optimization step.

For this experiment, we downsampled the original MNIST images from 28×28 pixels to 8×8 , resulting in input vectors with 64 components. The TNN architecture comprised (1) a single MPO layer consisting of six tensors, each with input and output dimension 2 (corresponding to six qubit wires, since $2^6 = 64$) and all bond dimensions set to 2; (2) a ReLU (rectified linear unit) activation layer; and (3) a dense output layer of dimension 64×10 . This architecture is depicted in Fig. 3.

Next, we applied the variational disentangling procedure to the trained MPO layer. For this demonstration, we fixed¹ the compressed MPO M'_χ as follows: We first brought the input MPO M_χ into canonical form and then truncated all but the largest singular value at each bond, resulting in a compressed bond dimension $\chi' = 1$. This rank-1 MPO served as the target for the disentangling optimization, in which we maximized the overlap (2) to best approximate M' .

We considered disentangling circuits with brickwall geometry composed exclusively of n -body gates, where $n = 2, 3, 4, 5, 6$. In Table I, we list the overlap between the target, completely disentangled ($\chi' = 1$) MPO, and the input MPO after applying only the first layer of disentanglers on each side. We see that increasing the gate size consistently boosts the

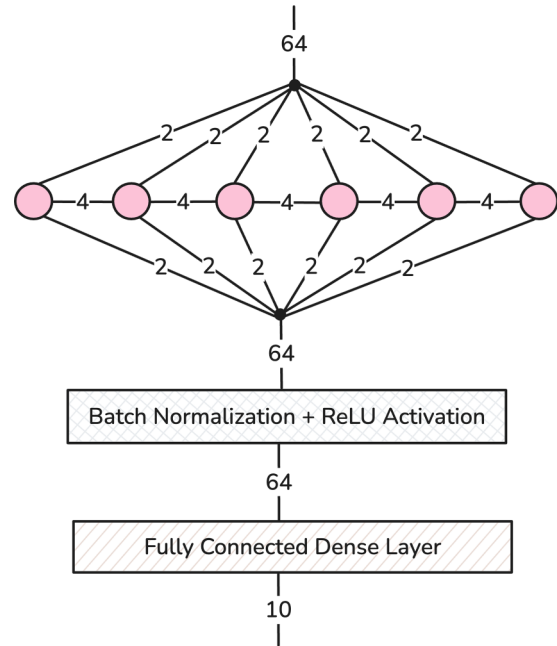


FIG. 3. A simple classical TNN used in the numerical experiments for variational disentanglers. The network begins with an MPO layer composed of six tensors (shaded circles), with all index dimensions indicated. Black dots represent `reshape()` operations: The top dot reshapes an input vector of size 64 into a six-index tensor (each index of dimension 2), while the bottom dot reshapes a six-index tensor back into a size-64 vector. The second layer applies standard batch normalization followed by a ReLU activation. The final layer is a fully connected dense layer represented by a 64×10 matrix.

overlap—from 62% for two-qubit gates to 96% for six-qubit gates—demonstrating that larger gates can suppress entanglement more effectively in a single step. This benefit, however, comes at a steep cost: The total gate count grows rapidly, and the number of hardware-compiled gates (evaluated using the FakeMontreal-v2 backend) increases by more than two orders of magnitude when moving from $n = 2$ to $n = 6$. This highlights a clear trade-off between circuit efficiency and entanglement suppression.

Next, we investigated the impact of increasing circuit depth. As shown in Fig. 4 (top left), deeper circuits yield progressively more faithful reconstructions of the target MPO at large depth. We found an anomalous dip in overlap be-

TABLE I. Effect of unitary size on MPO overlap and gate complexity. Hardware-compileable gate counts (FakeMontreal backend) are higher for larger unitaries.

Gate size	Gate count (1Q/2Q)	RealHW gates (1Q/2Q)	Overlap (%)
$2qU$	8/3	32/3	62
$3qU$	32/19	174/44	72
$4qU$	164/100	1062/249	81
$5qU$	693/442	4481/1125	93
$6qU$	2893/1868	18 481/4799	96

¹More generally, one could jointly optimize both the disentanglers and the compressed MPO tensors. The latter can be updated in a similar fashion, for example, via an SVD of the MPO tensor environments appearing inside the trace in the overlap expression (2).

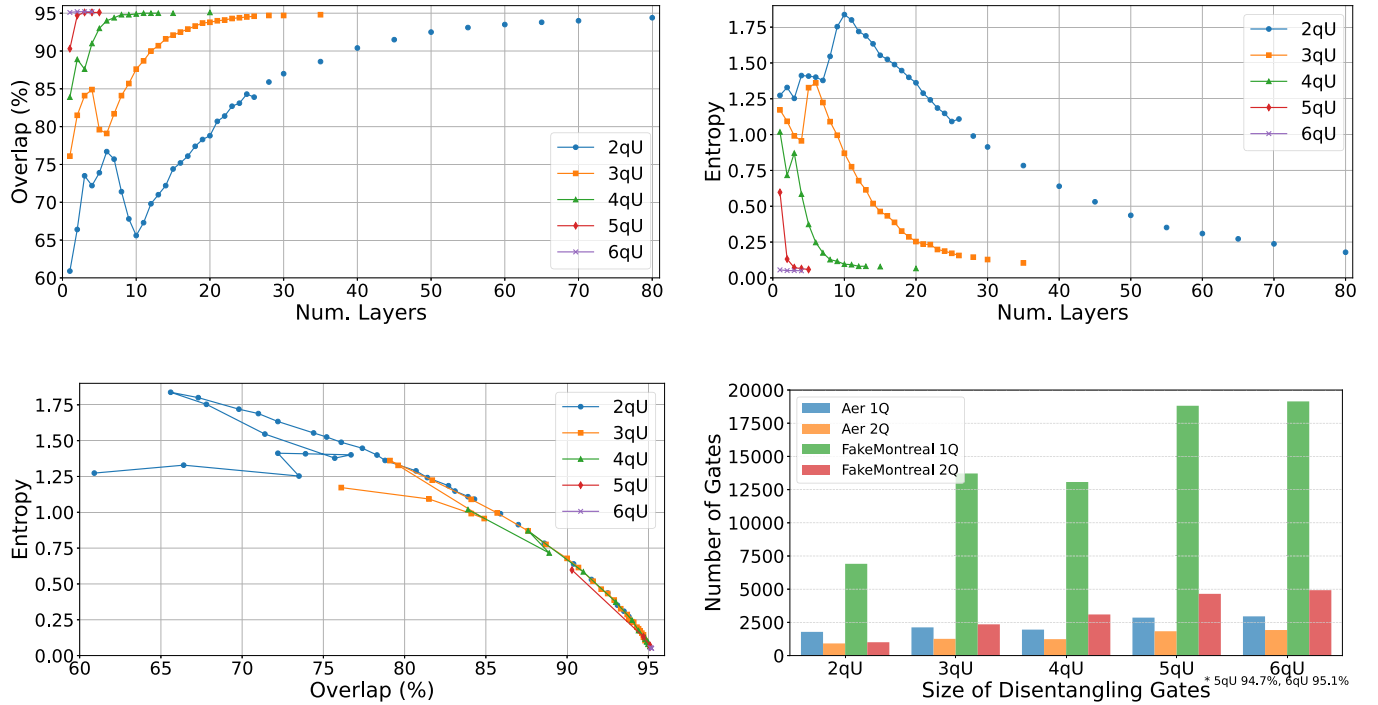


FIG. 4. (Top left) Overlap, Eq. (2), between disentangled MPO and the rank-1 target, as a function of the number of disentangling layers. Deeper circuits improve disentangling. (Top right) Average bond entanglement entropy S_{avg} , Eq. (5), of the MPO vs number of disentangling layers. (Bottom left) Relationship between MPO overlap and entanglement entropy. Higher fidelity correlates with reduced entropy. (Bottom right) Total gates required to reach $\sim 94\%$ overlap. Two-qubit unitary (2qU) circuits minimize hardware overhead, making them favorable for NISQ platforms.

tween beyond five and ten layers, which diminishes as the gates become more nonlocal. At present, we do not have an explanation for this behavior.

We also quantified compression quality by computing the average bond entanglement entropy of the disentangled MPO. Suppose the MPO has b bond indices. For each bond $k = 1, 2, \dots, b$, we reshape the MPO into a matrix M_k by grouping all open indices to the left of bond k into a single input index and all open indices to the right into a single output index. Let λ_k denote the diagonal matrix of singular values of M_k . The bond entanglement entropy is then given by

$$S_k = -\text{Tr}(\hat{\lambda}_k^2 \log \hat{\lambda}_k^2), \quad S_{\text{avg}} \equiv \frac{1}{b} \sum_{k=1}^b S_k, \quad (5)$$

where $\hat{\lambda}_k = \lambda_k / \sqrt{\text{Tr}(\lambda_k^2)}$ are the normalized singular values. All logarithms are base 2 unless otherwise stated. Interpreting the MPO as a pure $2(b+1)$ -body quantum state, S_k quantifies the entanglement across the bipartition defined by bond k . The maximum value of S_{avg} is

$$S_{\text{avg}}^{\text{max}} = \frac{1}{b} \sum_{k=1}^b \log \chi_k, \quad (6)$$

where $\chi_1, \chi_2, \dots, \chi_b$ are the MPO bond dimensions. This follows from the fact that the maximum possible Schmidt rank for each MPO bond is equal to its corresponding bond dimension.

Figure 4 (top right) shows a consistent decrease in entropy with increasing disentangling circuit depth, indicating effective disentangling. This is further corroborated in Fig. 4

(bottom left), where overlap fidelity is inversely correlated with the final entropy—reinforcing the interpretation that the circuits genuinely disentangle the MPO.

From a practical deployment standpoint, disentanglers must be both expressive and hardware efficient. Figure 4 (bottom right) compares the number of quantum gates needed to achieve approximately 94% MPO overlap across different unitary sizes. While five- and six-qubit gate circuits achieve high accuracy, they incur a prohibitive gate cost. Two-qubit gates are the most efficient, striking a balance between expressivity and NISQ suitability. Strategies for improving the corresponding overlaps in these cases are discussed in Sec. VI.

TABLE II. Classification test accuracy for different models with and without variationally optimized disentanglers. Below, classical disentanglers correspond to unitary gates that are run as a custom layer inside a classical neural network (custom neural networks layers are discussed in Sec. V), while quantum disentanglers are run as simulated quantum circuits via Qiskit on the noise-free AER backend.

Model	Accuracy (%)
Original TNN (Fig. 3)	92.6 (baseline)
Disentangled MPO + gates (classical)	92.6
Disentangled MPO + gates (quantum)	92.4
Disentangled MPO + gates + six-body unitary (classical)	94.7 (+2.1)
Disentangled MPO + gates + six-qubit unitary (quantum)	93.6 (+1.2)

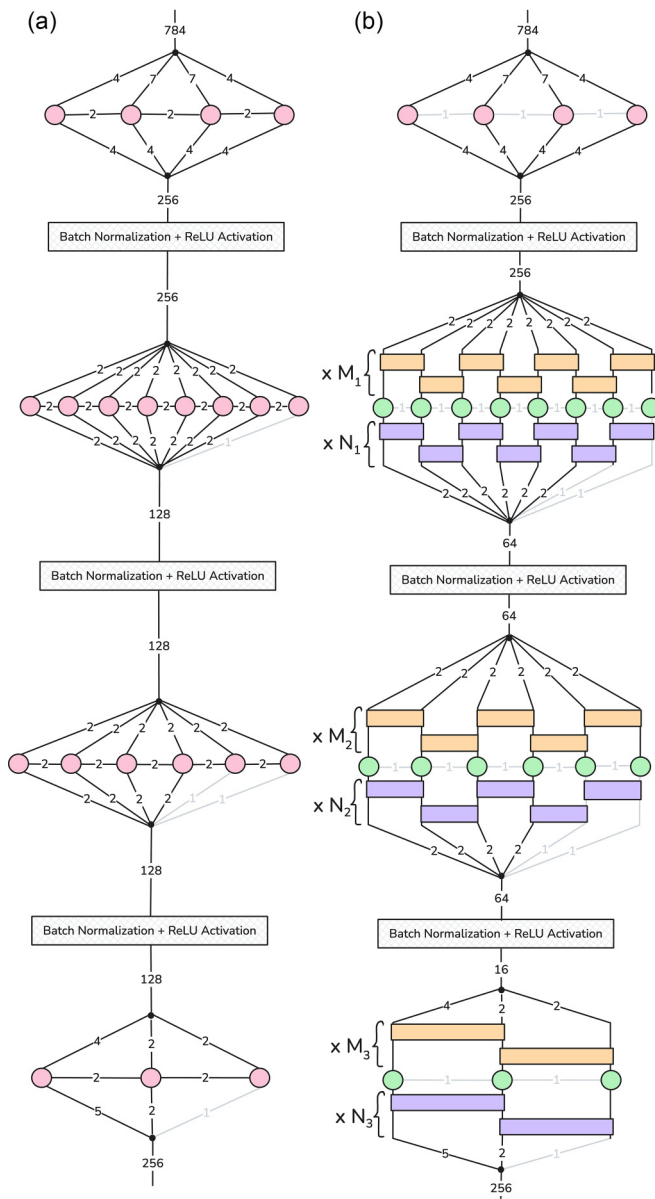


FIG. 5. Neural network architectures used in the MNIST experiments. Trivial indices having dimension equal to 1 are shown in light gray. (a) The original TNN trained for MNIST classification, comprising four MPO layers (pink) interleaved with batch normalization and ReLU activation functions. (b) The schematic of the hybrid classical-quantum neural network, featuring completely disentangled MPOs (green) augmented with multiple layers of disentanglers.

We also benchmarked the test accuracy of the various models that we have considered above. The results are listed in Table II. Disentangled MPOs constructed from six-qubit gates maintain baseline accuracy despite compression. In these simulations, $6qU$ disentanglers were chosen for convenience; this choice affects the residual entanglement structure but not the classification accuracy, as disentanglers in this first stage only reduce entanglement and are not yet optimized to improve accuracy.

We note that adding trainable six-qubit variational gates before and after the MPO further improves accuracy to 94.7%, while their quantum circuit counterparts achieve 93.6%. This

TABLE III. Classification test accuracy of different models with and without gradient-descent optimized implicit disentangles for MNIST classification. The disentangled TNN below is the one depicted in Fig. 5(b) but without any disentangles (setting $M_i = N_i = 0$, for all i). One-body gates are not shown in the figures. Best model is highlighted in bold font.

Model	Params	Accuracy (%)
Original TNN [Fig. 5(a)]	1008	94.47 (baseline)
Disentangled TNN	854	90.31
Disentangled TNN + only CNOTS	854	91.58
Disentangled TNN + CNOTS + one-body gates	876	92.78
Disentangled TNN + CNOTS + two-body gates	963	93.88
Disentangled TNN + CNOTS + one- and two-body gates	930	93.51
Disentangled TNN + CNOTS + one- and two-body gates	1029	94.66

points to another opportunity for leveraging quantum computing to boost the model’s accuracy—by adding layers to the network that run on the quantum. Although the unitary blocks are mathematically identical in both cases, the quantum-simulated pipeline requires explicit state preparation and tomography of the output. These steps introduce small reconstruction errors even on the noise-free AER backend, which we believe account for the observed $\sim 1\%$ drop in accuracy compared to the purely classical implementation.

The Qiskit quantum simulations summarized in the table were performed as follows (boldface denotes Qiskit modules and routines):

- (1) Normalize and encode the input data using `Amplitude` or `MPS Encoding`.
- (2) Add unitaries to the quantum circuit via the `Operator` and `Unitary` methods.
- (3) Prepare a `StateTomography` experiment using `qiskit_experiments.library`.
- (4) Extract the state vector from the resulting density matrix.
- (5) Apply the initial normalization and contract with the reduced MPO.
- (6) Repeat the procedure with unitaries applied to the other side of the MPO.

Finally, we highlight two empirical observations that warrant further investigation but lie beyond the scope of this work. First, the disentangling algorithm described above converged more rapidly and achieved higher overlaps than a gradient-descent optimization of unitaries aimed at minimizing the loss in Eq. (2)². Second, disentangling was consistently more

²This differs from our second approach for finding disentanglers, described in Sec. V. Here, gradient descent was used to optimize unitaries that *explicitly* disentangle the MPO by maximizing the overlap (2), as in the variational procedure, whereas Sec. V discusses a gradient-descent approach for determining *implicitly* disentangling gates.

TABLE IV. Details of the circuit layers used in the MNIST models listed in Table III. Here, 1b and 2b denote trainable one-body and two-body gates, respectively, while cnot gates remain fixed during training. The values M_i and N_i indicate the number of layers in the circuits preceding and following the MPO layers in the model [see Fig. 5(b)].

Model	No. of Layers	Gate types
Only CNOTS	$M_1 = 1; N_1 = 2$ $M_2 = 6; N_2 = 2$ $M_3 = N_3 = 1$	All CNOTS All CNOTS All CNOTS
CNOTS + 1b	$M_1 = 1; N_1 = 2$ $M_2 = 8; N_2 = 1$ $M_3 = N_3 = 1$	All CNOTS 1b + CNOTS + 1b + 5 × CNOTS; CNOTS 1b; CNOTS
CNOTS + 2b	$M_1 = 1; N_1 = 2$ $M_2 = 6; N_2 = 2$ $M_3 = N_3 = 1$	CNOTS; CNOTS + 2b Odd layers = CNOTS; even layers = 2b Odd layers = CNOTS; even layers = 2b
CNOTS + 1b and 2b	$M_1 = 1; N_1 = 2$ $M_2 = 6; N_2 = 2$ $M_3 = N_3 = 1$	CNOTS; CNOTS + 2b Odd layers = CNOTS; even layers = 2b CNOTS + 2b + CNOTS + 2b

effective and efficient for MPOs composed of positive-valued tensors (all entries positive) than for MPOs with mixed-sign entries. This may be related to recent results indicating that the computational difficulty of contracting a tensor network is influenced by the sign structure of its entries [26,31].

V. IMPLICIT DISENTANGLERS VIA GRADIENT DESCENT

The disentanglers can also be obtained via gradient descent during the “healing” phase of the model, which takes place in the compression step. In this approach, we replace the dense weight inside the network $\mathcal{N}$ with the product $Q_L M'_{\chi'} Q_R$ and jointly optimize the disentangling gates in the unitary circuits Q_L and Q_R together with the “disentangled” MPO $M'_{\chi'}$.

The optimization here minimizes the model’s loss—defined here as the standard cross-entropy loss for image classification—rather than explicitly disentangling the MPOs by maximizing the overlap (2). In other words, the gates are tuned to improve the overall model performance (or equivalently, maximize its training accuracy), not to directly suppress entanglement. We nevertheless refer to them as “disentanglers” since, together with the compact MPO—typically exhibiting an average bond entanglement entropy (5) lower than both the original MPO and the maximum entropy (6) allowed by the compressed bond dimensions—they provide an accurate replacement for the original MPOs. Notably, even one-body unitaries can serve effectively as implicit disentanglers when embedded inside a scaffold of fixed cnot gates. In practice, we find that this replacement does not usually degrade the model accuracy without substantially increasing the model size. In this implicit setting, the trade-off of interest is therefore between (1) increasing the MPO bond dimension to recover accuracy and (2) keeping the MPO bond dimension fixed while increasing circuit depth to recover accuracy.

To perform this optimization, we implemented a custom PyTorch module representing an abstract layer composed of the quantum circuit Q_L , a compressed MPO $M'_{\chi'}$, and the quantum circuit Q_R . Specifically, we derived a custom class from PyTorch’s `nn.Module` base class to store lists of unitary gates for Q_L and Q_R , along with a list of tensors forming the

MPO $M'_{\chi'}$. We then overrode the `forward()` method to define the data flow through this custom layer.³

Each unitary gate g was parametrized as $g = e^t$, where t is a randomly initialized lower-triangular matrix. As noted earlier, we restricted our focus to real orthogonal matrices (for implementation convenience and does not limit the generality of the framework), which is enforced by constraining t to be real and lower triangular.⁴

The `forward()` method of our custom module multiplies the input batch of activations, represented as a matrix X , with the circuits and MPO to return the output activations Y obtained via a sequence of matrix multiplications,

$$Y = X Q_L M'_{\chi'} Q_R. \quad (7)$$

Here, by slight abuse of notation, $Q_{L,R}$ also denotes the matrix obtained by contracting all gates in the corresponding circuit. In practice, the total multiplication (7) was performed using PyTorch’s `einsum()` function, following an optimal contraction order determined by the built-in `opt_einsum()` utility.

As in the variational optimization of the disentangler, for very deep circuits the cost of the tensor network contraction (7) can be significantly reduced by using established contraction schemes [29,30]. When the input activations X are represented as a large matrix, they can first be decomposed into a low-rank tensor network prior to performing the contraction (7). However, because the circuits in our experiments were relatively shallow, the direct `einsum()`-based contraction was sufficient.

During training of our custom models with automatic differentiation in PyTorch, we observed vanishing gradients, especially as the depth of the circuits Q_L and Q_R increased. We were able to substantially improve model performance by inserting nonlinearities between gates. This effectively segments the circuits Q_L and Q_R into subcircuits that must

³See, for example, the simple PyTorch tutorial [32].

⁴This parametrization also constrains the optimization to orthogonal matrices with determinant equal to 1; however, this restriction did not seem to adversely impact the model accuracy.

TABLE V. Classification test accuracy of different models with and without gradient-descent optimized implicit disentanglers for CIFAR-10 classification. The disentangled TNN below is the one depicted in Fig. 6(b) but without any disentanglers (setting $M_i = N_i = 0$, for all i). One-body gates are not shown in the figures. Best disentangled model highlighted in bold font.

Model	Params	Accuracy (%)
Original TNN [Fig. 6(a)]	36.7k	61.29 (baseline)
Disentangled TNN	24.2k	58.04
Disentangled TNN + only CNOTS	24.2k	58.59
Disentangled TNN + CNOTS + one-body gates	25.5k	59.19
Disentangled TNN + CNOTS + two-body gates	24.3k	59.90
Disentangled TNN + CNOTS + one- and two-body gates	25.3k	60.74

be installed separately on the quantum computer, while the nonlinear layers remain on the classical computer.

This presents the principal challenge of training implicitly disentangling circuits via gradient descent: Deep disentangling circuits may need to be partitioned into shallower subcircuits, potentially diluting the quantum computational advantage due to the overhead of repeatedly preparing activations on the quantum device, performing measurements, and reconstructing output activations via tomography.

A. Numerical results

MNIST classification. The results of our experiments are listed in Table III. We first trained the TNN depicted in Fig. 5(a) for MNIST classification obtaining a baseline accuracy of 94.47%. (We once again bypassed the MPO compression step as we could easily train a tensorized version of the network.) We then consider a target disentangled TNN with the bond dimensions depicted in Fig. 5(b) without any gates, namely, setting number of layers $M_i = N_i = 0$, for all i . After now introducing some one-body gates (not shown in the figure) and two-body gates, we perform the gradient-descent optimization described above. We also fixed some of the gates to be cnot gates. The details of the circuit layers inside our MNIST models are listed in Table IV.

Remarkably, including only fixed cnot gates (kept constant during optimization) led to a slight increase in model accuracy. Introducing trainable one-body and two-body gates separately yielded further improvements, and combining both one-body and two-body trainable gates allowed us to match the baseline accuracy.

CIFAR-10 classification. The results of the CIFAR-10 experiments are presented in Table V. We began by training a TNN for CIFAR-10 classification with the architecture shown in Fig. 6(a). The corresponding disentangled TNN, depicted in Fig. 6(b), was obtained by setting all $M_i = N_i = 0$. Unlike the MNIST case, we did not fully disentangle the MPOs; instead, we set all MPO bond dimensions to be larger than one but smaller than those in the initial TNN. Once again, adding only cnot disentanglers resulted in a slight improvement in accuracy. Adding one-body and two-body disentanglers also

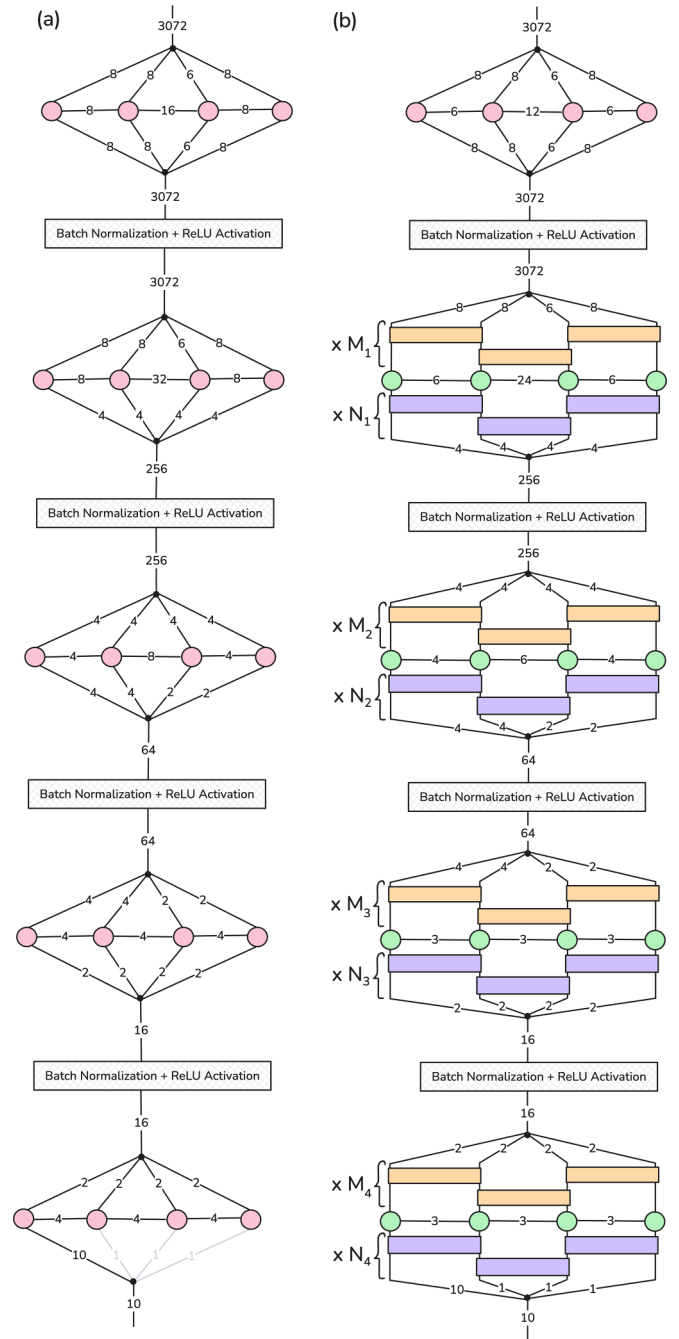


FIG. 6. Neural network architectures used in the CIFAR experiments. (a) The original TNN trained for CIFAR-10 classification, comprising four MPO layers (pink) interleaved with batch normalization and ReLU activation functions. (b) The schematic of the hybrid classical-quantum neural network, featuring disentangled MPOs (green) augmented with multiple layers of disentanglers.

improved accuracy. The details of the circuit layers inside our CIFAR models are listed in Table VI.

The best-performing model was, once again, obtained using a combination of one-body and two-body gates. This model does not yet match the baseline accuracy—owing to its substantially fewer parameters (25.3k) compared to the baseline (36.7k).

TABLE VI. Details of the circuit layers used in the CIFAR-10 models listed in Table V. Here, 1b and 2b denote trainable one-body and two-body gates, respectively, while CNOT gates remain fixed during training. The values M_i and N_i indicate the number of layers in the circuits preceding and following the MPO layers in the model [see Fig. 6(b)].

Model	No. of Layers	Gate types
Only CNOTS	$M_1 = N_1 = 1$	All CNOTS
	$M_2 = 3; N_2 = 4$	All CNOTS
	$M_3 = 4; N_3 = 3$	All CNOTS
	$M_4 = N_4 = 1$	All CNOTS
CNOTS + 1b	$M_1 = N_1 = 2$	All CNOTS
		Odd layers = 1b; even layers = CNOTS;
	$M_2 = 6; N_2 = 8$	remaining CNOTS
	$M_3 = 8; N_3 = 6$	CNOTS
CNOTS + 2b	$M_4 = N_4 = 2$	All CNOTS
	$M_1 = N_1 = 1$	CNOTS; 2b
	$M_2 = 3; N_2 = 4$	Odd layers = CNOTS; even layers = 2b
	$M_3 = 4; N_3 = 3$	Odd layers = CNOTS; even layers = 2b
CNOTS + 1b + 2b	$M_4 = N_4 = 1$	2b + CNOTS
	$M_1 = N_1 = 2$	CNOTS
		1b + CNOTS + 1b + 2b + CNOTS + 2b + 1b + CNOTS + 1b; CNOTS + 1b + CNOTS + 1b + CNOTS + 2b + CNOTS + 2b + CNOTS + 1b + CNOTS + 1b +
	$M_2 = 9; N_2 = 12$	CNOTS
	$M_3 = 8; N_3 = 6$	All CNOTS
	$M_4 = N_4 = 2$	All CNOTS

We emphasize that the modest parameter reduction in this CIFAR-10 proof-of-concept ($36.7k \rightarrow 25.3k$) is not, by itself, a motivation for deploying quantum hardware. Rather, it illustrates the primary mechanism studied in this work: Quantum disentangling circuits can compensate for a reduction in MPO bond dimension, thereby preserving accuracy even when the classical tensor network is deliberately constrained to a smaller bond dimension.

At scale, the relevant constraint is not the absolute number of parameters in these small models, but the classical cost of increasing the MPO bond dimension. Increasing χ improves the expressivity of an MPO layer, but it also increases the cost of tensor network contractions and the memory footprint. In contrast, surrounding a fixed-bond-dimension MPO M'_χ with quantum circuits Q_L, Q_R provides an alternative way to recover lost performance without increasing the classical tensor network complexity.

Beyond this compression role, our results also suggest a secondary implication: Circuit layers may serve as an additional expressive resource. In some settings (e.g., the MNIST results in Table III), adding trainable quantum gates allows the hybrid model to match or even exceed the original baseline accuracy, indicating that circuit depth can enlarge the effective function class represented by $Q_L M'_\chi Q_R$. While this expressive enhancement is not uniformly observed across

all experiments, the disentangling framework developed here provides a concrete setting in which such effects can be systematically explored.

We emphasize that batch normalization and ReLU activations are essential in this setting. In particular, when disentangling circuits became too deep, we observed vanishing gradients that led to poor training and higher loss. We substantially improved performance by inserting additional nonlinearities and normalization layers inside the disentangling circuits, beyond those already present in the original TNN. This behavior is reminiscent of the training instabilities reported in deep linear networks—such as plateaus and vanishing gradients with increasing depth [33]—where the absence of nonlinearities fundamentally compromises trainability. With these extra nonlinear layers, our hybrid tensor network models retained stable training and achieved the accuracies reported in Tables III and V.

VI. DISCUSSION

The most powerful deep learning models in use today are entirely classical, yet a significant portion of their computational and memory costs is concentrated in a few large linear layers. If even a small fraction of this workload could be offloaded to a quantum computer—without sacrificing

accuracy—it could offer a tangible pathway toward near-term quantum utility. In this work, we have explored this possibility by targeting bottleneck layers of a pretrained neural network, compressing and translating them into quantum circuits, and embedding them within a hybrid classical-quantum inference pipeline.

Our strategy involves first compressing the dense weight matrix W into a matrix product operator M_χ , ensuring no significant degradation in model accuracy, and subsequently disentangling M_χ into a more compact MPO $M_{\chi'}$, with reduced bond dimension $\chi' < \chi$. In this approximate representation, the disentangling circuits are installed and executed on a quantum processor, while the remainder of the network—including the compressed MPO $M_{\chi'}$ —runs on classical hardware. We expect that this two-step approach yields simpler quantum circuits compared to directly approximating W with a unitary circuit.

The principal challenge to realizing quantum advantage in this framework lies in reducing the size and complexity of the disentangling circuits. While the intermediate MPO representation provides greater control and simplification, a comprehensive exploration of circuit ansatz is needed, including systematic investigation of circuit geometry and gate selection. Notably, the potential of one-body variational gates combined with fixed, nonvariational two-body gates remains underexplored. Carefully designed circuit ansatz leveraging such gate arrangements could drastically reduce the complexity of hardware-transpiled circuits. Alternatively, one might fix the gate set and optimize solely over circuit geometry—for instance, assessing the extent to which an MPO can be disentangled using only cnot gates. It would also be interesting to explore disentangling maps composed of more than one Kraus circuit built, corresponding to $k > 1$ in Eq. (1).

In terms of computational overhead, the hybrid inference pipeline includes several costs beyond those of a purely

classical execution: (1) state preparation to encode classical activations, (2) execution of the disentangling circuits, and (3) measurement and classical postprocessing to recover output activations. Full state tomography is not expected to scale, and in realistic settings one would estimate only task-relevant observables or use sampling-based reconstruction and structured encodings (e.g., MPS-based state preparation). Therefore, as a proof of concept studied here, these quantum input/output costs dominate and preclude any practical quantum speedup. Our aim is not to claim even near-term inference advantage, although future enhancements in quantum technologies may mitigate some of these costs. Instead, the current benefit of our scheme is that MPO compression trades dense linear-algebra costs for tensor network contraction costs that scale polynomially with the MPO bond dimension. Once increasing this bond dimension becomes computationally or memory prohibitive, surrounding the MPO with quantum circuits provides an alternative scaling direction: Increasing circuit depth can enhance the effective expressivity of the linear transformation without enlarging the classical tensor network itself.

We remark that the costs associated with measurement and tomography steps in the quantum workflow could be mitigated by either replacing multiple layers by a single effective circuit or by optimally combining (and compiling) circuits from neighboring layers into shorter effective circuits. It may also be possible to reframe measurements and partial tomography as computational resources rather than costs, as these operations could be exploited to replace some nonlinear transformations within the neural network.

DATA AVAILABILITY

The data that support the findings of this article are not publicly available. The data are available from the authors upon reasonable request.

-
- [1] G. Acampora, A. Ambainis, N. Ares, L. Banchi, P. Bhardwaj, D. Binosi, G. A. D. Briggs, T. Calarco, V. Dunjko, J. Eisert, *et al.*, Quantum computing and artificial intelligence: Status and perspectives, [arXiv:2505.23860](#).
 - [2] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'Brien, A variational eigenvalue solver on a photonic quantum processor, *Nat. Commun.* **5**, 4213 (2014).
 - [3] J. Tilly, H. Chen, S. Cao, D. Picozzi, K. Setia, Y. Li, E. Grant, L. Wossnig, I. Rungger, G. H. Booth, and J. Tennyson, The variational quantum eigensolver: A review of methods and best practices, *Phys. Rep.* **986**, 1 (2022).
 - [4] E. Farhi, J. Goldstone, and S. Gutmann, A quantum approximate optimization algorithm, [arXiv:1411.4028](#).
 - [5] K. Blekos, D. Brand, A. Ceschini, C.-H. Chou, R.-H. Li, K. Pandya, and A. Summer, A review on quantum approximate optimization algorithm and its variants, *Phys. Rep.* **1068**, 1 (2024).
 - [6] M. H. Amin, E. Andriyash, J. Rolfe, B. Kulchitsky, and R. Melko, Quantum Boltzmann machine, *Phys. Rev. X* **8**, 021050 (2018).
 - [7] 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 (London)* **567**, 209 (2019).
 - [8] A. Poggiali, A. Berti, A. Bernasconi, G. M. Del Corso, and R. Guidotti, Quantum clustering with k -means: A hybrid approach, *Theor. Comput. Sci.* **992**, 114466 (2024).
 - [9] S. Lloyd, M. Mohseni, and P. Rebentrost, Quantum principal component analysis, *Nat. Phys.* **10**, 631 (2014).
 - [10] A. W. Harrow, A. Hassidim, and S. Lloyd, Quantum algorithm for linear systems of equations, *Phys. Rev. Lett.* **103**, 150502 (2009).
 - [11] J.-G. Liu and L. Wang, Differentiable learning of quantum circuit Born machines, *Phys. Rev. A* **98**, 062324 (2018).
 - [12] Z. Holmes, K. Sharma, M. Cerezo, and P. J. Coles, Connecting ansatz expressibility to gradient magnitudes and barren plateaus, *PRX Quantum* **3**, 010313 (2022).
 - [13] S. Sim, P. D. Johnson, and A. Aspuru-Guzik, Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms, *Adv. Quantum Technol.* **2**, 1900070 (2019).

- [14] J. Preskill, Quantum computing in the NISQ era and beyond, *Quantum* **2**, 79 (2018).
- [15] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio, *et al.*, Variational quantum algorithms, *Nat. Rev. Phys.* **3**, 625 (2021).
- [16] J. R. McClean, S. Boixo, V. N. Smelyanskiy, R. Babbush, and H. Neven, Barren plateaus in quantum neural network training landscapes, *Nat. Commun.* **9**, 4812 (2018).
- [17] S.-J. Ran, Encoding of matrix product states into quantum circuits of one- and two-qubit gates, *Phys. Rev. A* **101**, 032310 (2020).
- [18] A. Novikov, D. Podoprikhin, A. Osokin, and D. P. Vetrov, Tensorizing neural networks, in *Advances in Neural Information Processing Systems*, edited by C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett (Curran Associates, Inc., Red Hook, NY, USA, 2015), Vol. 28.
- [19] M. Wang, Y. Pan, Z. Xu, G. Li, X. Yang, D. Mandic, and A. Cichocki, Tensor networks meet neural networks: A survey and future perspectives, [arXiv:2302.09019](https://arxiv.org/abs/2302.09019).
- [20] A. Tomut, S. S. Jahromi, A. Sarkar, U. Kurt, S. Singh, F. Ishtiaq, C. Muñoz, P. S. Bajaj, A. Elborady, G. del Bimbo, *et al.*, CompactifAI: Extreme compression of large language models using quantum-inspired tensor networks, [arXiv:2401.14109](https://arxiv.org/abs/2401.14109).
- [21] B. Aizpurua, S. S. Jahromi, S. Singh, and R. Orus, Quantum large language models via tensor network disentanglers, [arXiv:2410.17397](https://arxiv.org/abs/2410.17397).
- [22] L. Deng, The mnist database of handwritten digit images for machine learning research, *IEEE Signal Process. Mag.* **29**, 141 (2012).
- [23] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, Gradient-based learning applied to document recognition, *Proc. IEEE* **86**, 2278 (1998).
- [24] A. Krizhevsky, Learning multiple layers of features from tiny images, Technical Report, University of Toronto (2009).
- [25] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information* (Cambridge University Press, Cambridge, 2010).
- [26] J. Jiang, J. Chen, N. Schuch, and D. Hangleiter, Positive bias makes tensor-network contraction tractable, in *Proceedings of the 57th Annual ACM Symposium on Theory of Computing (STOC 2025)* (ACM, New York, 2025), pp. 471–482.
- [27] G. Vidal, Entanglement renormalization, *Phys. Rev. Lett.* **99**, 220405 (2007).
- [28] G. Evenbly and G. Vidal, Algorithms for entanglement renormalization, *Phys. Rev. B* **79**, 144108 (2009).
- [29] S.-J. Ran, E. Tirrito, C. Peng, X. Chen, L. Tagliacozzo, G. Su, and M. Lewenstein, *Tensor Network Contractions*, Lecture Notes in Physics (Springer, Cham, 2020).
- [30] J. Gray and G. K.-L. Chan, Hyperoptimized approximate contraction of tensor networks with arbitrary geometry, *Phys. Rev. X* **14**, 011009 (2024).
- [31] J. Chen, J. Jiang, D. Hangleiter, and N. Schuch, Sign problem in tensor-network contraction, *PRX Quantum* **6**, 010312 (2025).
- [32] https://docs.pytorch.org/tutorials/beginner/nn_tutorial.html.
- [33] A. M. Saxe, J. L. McClelland, and S. Ganguli, Exact solutions to the nonlinear dynamics of learning in deep linear neural networks, [arXiv:1312.6120](https://arxiv.org/abs/1312.6120).