PROCEEDINGS A

royalsocietypublishing.org/journal/rspa

Review

**Cite this article:** Rieser H-M, Köster F, Raulf AP. 2023 Tensor networks for quantum machine learning. *Proc. R. Soc. A* **479**: 20230218.
https://doi.org/10.1098/rspa.2023.0218

Received: 29 March 2023
Accepted: 22 June 2023

**Subject Areas:**
quantum computing, artificial intelligence

**Keywords:**
tensor network, quantum machine learning, quantum computing, encoding

**Author for correspondence:**
Hans-Martin Rieser
e-mail: hans-martin.rieser@dlr.de



# Tensor networks for quantum machine learning

Hans-Martin Rieser, Frank Köster and Arne Peter Raulf

Deutsches Zentrum für Luft- und Raumfahrt, Institute for AI Safety and Security, Ulm/St. Augustin, Germany

H-MR, 0000-0002-1921-1436; APR, 0009-0003-8672-3014

Once developed for quantum theory, tensor networks (TNs) have been established as a successful machine learning (ML) paradigm. Now, they have been ported back to the quantum realm in the emerging field of quantum ML to assess problems that classical computers are unable to solve efficiently. Their nature at the interface between physics and ML makes TNs easily deployable on quantum computers. In this review article, we shed light on one of the major architectures considered to be predestined for variational quantum ML. In particular, we discuss how layouts like matrix product state, projected entangled pair states, tree tensor networks and multi-scale entanglement renormalization ansatz can be mapped to a quantum computer, how they can be used for ML and data encoding and which implementation techniques improve their performance.

## 1. Introduction

Quantum computation is widely believed to set a new paradigm in computation. Using quantum phenomena allows to solve certain problems [1] far more efficiently than classical binary algorithms. This raises hope that quantum implementations of other tasks also may provide quantum advantages.

One of the applications that could benefit from the access to the high dimensional Hilbert spaces of quantum computers is machine learning (ML). ML is a data-driven approach for solving complex problems. An ML algorithm generates a model from



THE ROYAL SOCIETY PUBLISHING

training data that can be used to make predictions against previously unseen data. Quantum machine learning (QML) could advance learning by improved generalization to unknown data [2], higher noise robustness and the need for less training data [3] and provide a more natural approach to quantum data analysis circumventing intermediate measurements [4] or generally a better computational complexity scaling [5].

Promising candidates for QML architectures are tensor networks (TNs). They provide a structured approach for handling large objects with tensor structure which carry high amounts of correlated information like quantum states. Initially developed to store and process physical states of many-body quantum systems in numerical simulations [6,7], TNs also turned out to be useful for ML applications. Their approach to realize learning architectures is complementary to neural networks. As the TN description uses a (quantum) state and operator formulation, the transfer to a quantum computer can be done naturally.

In this review, we focus on the application of TNs for QML. We will begin with a short introduction to the classical TN theory including optimization and ML approaches in §2. Then, we will discuss how to apply these concepts to a quantum computer in §3 and the encoding of data to quantum states for ML applications in §4.

We will not cover many aspects of classical TNs in detail. For a deeper technical dive into TNs, the reader may refer to a general introduction [8] and the reviews on specific layouts [9,10] or decomposition and optimization techniques [11–13]. Applications are many-body quantum systems [14], nonlinear system identification [15] and classical ML [16,17].

The field of TN-QML is just developing, and notations and terminology vary throughout the community. Due to their origin in quantum theory, some authors call even ML with classical TNs 'quantum machine learning' [18]. A more suitable term would be *quantum-inspired* here [19]. Furthermore, one can argue that variational quantum circuits (VQCs) [20] require classical optimization and therefore are hybrid. In this article, however, we will use the following convention: methods fully evaluated without a quantum computer will be called *classical*. Methods developed for quantum computers that only require classical optimization of weights will be called *quantum* TNs (QTNs), as full quantum computation is still far out of reach. The term *hybrid* will be used for methods that combine QML with a classical data processing structure, e.g. pre-training or data pre- and post-processing.

## 2. Classical tensor networks

### (a) Introduction on tensors and tensor networks

TNs are a decomposition of large tensorial structures into several connected low rank tensors (figure 1*a*). Tensors are multidimensional arrays and therefore generalizations of vectors and matrices. While a matrix has two indices a tensor may have an arbitrary amount of indices. Technically, tensors describe objects from and maps between tangent and cotangent spaces. A tensor may have regular (lower) and dual (upper) indices depending on whether this index refers to objects from a tangent space or from a cotangent space. Each of these spaces may have different dimensions. A single tensor may have both types of indices and therefore connect to both tangent and cotangent spaces. One has to be careful with nomenclature as definitions are depending on the field tensors are employed in. In Mathematics and Physics, the tensor rank corresponds to its number of free indices, the dimension of an index is given by its range (e.g. four space–time or two spin dimensions). In ML, rank is rather used for the extension of matrix rank to tensors and the tensor's dimension is given by the number of indices. In this review, we will stick to the former notation.

The electromagnetic field tensor $F_{ab}$ from relativistic physics, for example, is a rank two tensor with four dimensional space–time indices each and the Riemann curvature tensor from general relativity $R^{a}{}_{bcd}$ has rank four, with one dual ($a$) and three regular ($b$, $c$, $d$) indices. As writing these tensors with indices can be very complex for larger problems, graphical notations like the Penrose

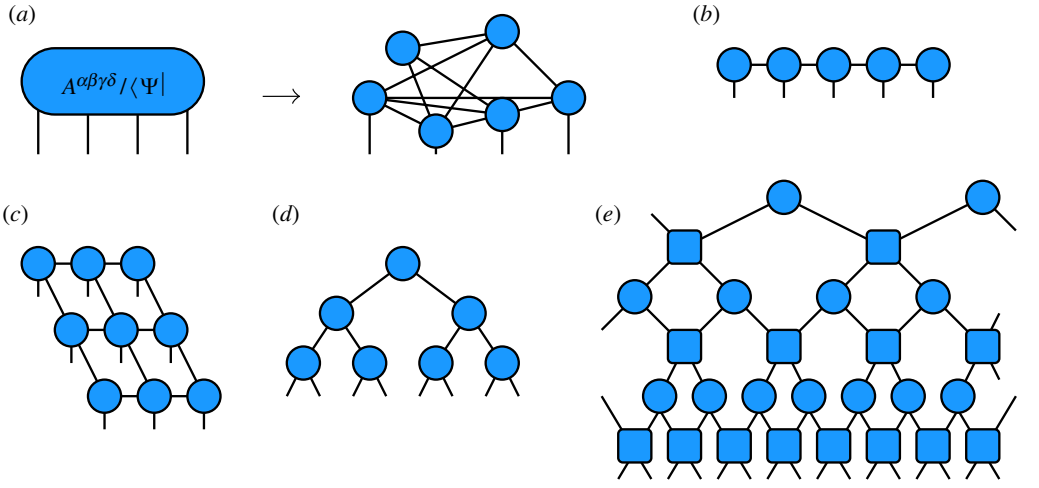


**Figure 1.** Examples for common tensor network layouts. (*a*) A general irregular tensor network. One may use any tensor network structure to express the large tensor $A^{abcd}$ or the wave function $\langle\Psi|$. However, regular tensor networks provide benefits in terms of interpretability and universality. Both (*b*) matrix product states (MPS) and (*c*) projected entangled pair states (PEPS) share the same grid structure with different dimensionality. (*d*) Tree tensor networks (TTNs) and (*e*) multi-scale entanglement renormalization ansatz (MERA) have a hierarchical structure, where MERA entangle between individual branches in contrast to TNN.

diagrams have been developed to simplify the handling of tensor equations [11]

$$R^a{}_{bcd} = \quad . \tag{2.1}$$


Free regular indices can be contracted with free dual indices by summing over all dimensions of this index. Einstein sum convention is a convenient form to express this contraction: having the same index twice automatically implies a summation

$$-\tfrac{1}{4\mu_0}\textstyle\sum_{a,b=0}^{3} F_{ab}F^{ab} = -\tfrac{1}{4\mu_0}F_{ab}F^{ab} = -\tfrac{1}{4\mu_0}\quad . \tag{2.2}$$


The graphical notation actually is one strength of the TN paradigm, as it provides accessibility to high dimensional states: each symbol is a tensor, its rank is given by the number of legs it has and the type of index determines the direction of the associated leg.

Figure 1*a* illustrates the idea behind the TN approach: a large tensor $A^{abcd}$ which may represent some quantum state $\langle\Psi|$ usually is hard to handle computationally. It requires large storage space and the manipulation of a large number of entries for each operation. Breaking down $A$ into a network of smaller connected tensors improves computability when the internal structure of $A$ matches the TN's layout. This requires a third kind of tensor index called an *internal* or virtual index that connects the constituents of the TN. We will denote this kind of index in Greek letters. The dimension of internal indices is called virtual bond dimension $\chi$. It gives an upper bound to how strongly the constituent tensors can be coupled and how much information can be shared between them.

TNs allow to apply local operations individually on each tensor node instead of having to evaluate the whole tensor at once. Tensors can be joined by contracting over connected indices or decomposed into several connected tensors. The most common technique for decompositions along a single direction is singular value decomposition (SVD), a generalization

of diagonalization that allows for arbitrary shaped tensors and zero singular values. Polar decomposition is faster than SVD, but does not allow for reducing bond dimensions easily. Tucker decomposition can be used for decomposing nodes within several directions at once [21].

The general idea behind tensor decomposition methods is to represent an arbitrary tensor with a specific set of constituent tensors. In SVD for instance, a tensor $A^{\alpha}_{a\delta}$ is decomposed into a unitary matrix $U^{\alpha}_{\beta}$, a diagonal singular value matrix $\Sigma^{\beta}_{\gamma}$ and an isometric matrix $V^{\gamma}_{a\delta}$

$$A^{\alpha}_{a\delta} = \overset{\text{SVD}}{=} = U^{\alpha}_{\beta}\Sigma^{\beta}_{\gamma}V^{\gamma}_{a\delta}, \tag{2.3}$$

where isometric tensors with known direction are given by triangles, unitaries and isometries with unknown orientation by squares and any other kind of tensor by a circle. Having access to the singular values in the diagonal matrix $\Sigma$ allows for reducing bond dimensions by removing zero singular values. This also can be used for approximation removing the lowest singular values having the least contribution to the bond.

Since tensor decompositions can be done in any direction on each bond and contracted to each side at any time, TNs are not unique but contain a gauge degree of freedom. One can make use of this property to bring the TN to a canonical form where the bonds form orthonormal Hilbert spaces [14] and the tensors are isometric or even unitary [8]. In many cases, it makes sense to bring the TN to such a canonical form where all tensor nodes are isometric. This has several advantages. First of all, isometric tensors automatically fulfil a normalization condition $A^{a\mu}A^{\dagger}_{a\tilde{\mu}} = \delta^{\mu}_{\tilde{\mu}}$ which enables the application of optimization schemes (see §2c). Second, it is mandatory for techniques that require directionality [22] or make use of the properties of isometries [23]. In particular, mapping a TN to a quantum circuit requires the tensors to be at least isometric (see §3).

*Applications in quantum computing* are based on the original application of TNs: reducing the computational cost of storing and evaluating lowly entangled multi-particle quantum states. This comes in handy for quantum computer simulations both for execution [24,25] and validation [26] of circuits as well as the estimation of errors [27]. Especially for short NISQ era algorithms, entanglement between many qubits usually is not too high and therefore circuit sizes well beyond the power of other simulation methods can be evaluated using TNs [28].

Additionally, TNs have been proposed to parallelize quantum simulations by cutting the system into several weakly entangled pieces and approximating the state of all but one piece by TNs [29]. Simulating a quantum computer may indeed be more resource efficient than using quantum hardware itself for a lot of low-entanglement applications [30]. This idea has been used already to develop quantum-inspired algorithms executed on classical hardware, e.g. for optimizing stock market portfolios [31,32] or radiotherapy plans [33] with quantum algorithms compressed to a classical TN approximation.

## (b) Tensor network layouts

Technically, the TN may have any shape but using regular TNs provides many benefits like simpler optimization, simpler control and transferability to problems with different structure. Such TNs are also more interpretable than arbitrary networks. The most common layouts either are grid (figure 1*b*,*c*) or hierarchical (figure 1*d*,*e*) states. Promoting state layouts to operators is either done by allowing every individual grid tensor node to have regular and dual indices or by connecting a complete hierarchical network with its dual on their topmost layers.

*Grid layouts* are the most natural TN description of physical lattices as the layout has a similar structure to the system. These layouts can be seen as derivatives of projected entangled pair states (PEPS) [34]. In quantum applications, PEPS nodes are constructed as composite objects consisting of coupled internal spins. Each spin connects to a neighbouring site via an edge and at each node the constituent spins are entangled and truncated, thus the name PEPS. The

number of spin tuples depends on the dimensionality of the network [9], typically a hypercube or hexagonal.

The constituent spin construction is very useful when employing PEPS for the description of quantum systems as this allows for spin constraints on the bonds. For ML applications, however, ansaetze for the nodes reflect computational approximations or inductive biases.

Although PEPS are defined for arbitrary dimensions, usually low dimensional layouts are used. One-dimensional PEPS are called matrix product states (MPS) or tensor trains. These are the simplest and most studied TN layouts [9]. In index notation, the MPS from figure 1*b* will look like

$$A^{abcde} = \tilde{A}_{\alpha}^{a(1)}\, \tilde{A}_{\beta}^{\alpha b(2)}\, \tilde{A}_{\gamma}^{\beta c(3)}\, \tilde{A}_{\delta}^{\gamma d(4)}\, \tilde{A}^{\delta e(5)}, \tag{2.4}$$

with constituent tensors $A^{(k)}$. Common gauges for MPS are called left, right and site canonical forms depending on the orientation of the isometric tensor nodes [9].

Brickwall or checkerboard TNs used in some quantum computing applications [35–37] are another variety of two-dimensional grid layouts equivalent to a hexagonal PEPS. The brickwall layout is a superposition of MPS up to a certain bond dimension [36] as it allows for the realization of MPS of different gauges overlapping at the same time.

*Hierarchical layouts* have input or output tensor nodes that are not coupled directly but are pooled on several internal layers. The simplest hierarchical structure is a tree tensor network (TTN) where two or more child nodes are connected to a parent node in the next layer until only a single node is left on the top. This layout is also called hierarchical Tucker decomposition. TTNs are able to catch both local entanglement on the lowest hierarchical level and long range entanglement between groups of nodes that are connected on higher levels of the hierarchy. As only aggregated information is passed on to the next hierarchy level, long range entanglement between individual input or output nodes cannot be captured. A TTN may have variable depth on different branches when the considered system is not homogeneous [38].

The multi-scale entanglement renormalization ansatz (MERA) is a derivative of an isometric TTN with better entropy scaling [39]. The main idea is to enhance the hierarchy with layers of unitary nodes connecting neighbouring branches. These so-called *disentanglers* reduce entanglement passed on to the next level (figure 1*e*). MERA has a higher computational cost than other layouts due to the loops, but it can capture symmetry and far higher entanglement [9,40] while still being efficiently storable [8]. Varieties of MERA offer even better entropy scaling [14]. Both TTN and MERA can be generalized to higher dimensions by considering unit cells of the respective dimension at each node [41,42].

The layout of a TN determines the maximal entanglement or internal correlation it can support. This gives a bound on the system type the TN can approximate without having a virtual bond dimension scaling exponentially with the system size. For MPS and PEPS entanglement fulfils an area law which means, that the amount of entanglement between a sub-network and its surroundings scales with its boundary [43]. This means, the entanglement for a one-dimensional MPS is constant [9], for a two-dimensional PEPS it scales linearly. For MERA layouts, the entanglement scaling gets a logarithmic correction to the area law [39,44,45]. Complex variants like branching MERA offer a steeper scaling up to a volume law [14,46], where a sub-network's entanglement with the surroundings depends on the number of nodes within itself. However, these layouts come with the trade-off of high computational costs that may render the layout unpractical without using superposition on a quantum computer.

In practice, the choice for a specific layout usually is a trade-off between the possible entanglement and the computational cost: MPS and TTN can be contracted efficiently, MERA and PEPS usually are costly.

Further refinements can be made by applying symmetries to the TN [9]. Relevant symmetric systems are homogeneous or periodic grids or layers in hierarchical networks [8]. For ML, this reduces the complexity of the TN and makes it easier to train.

## (c) Optimization methods

The term *optimizing TNs* can refer to two things. The size of a TN representation can be reduced by iterative executions of tensor decompositions along the internal bonds. This allows for the local adaption of bond dimensions to relevant degrees of freedom, e.g. by defining a threshold for relevant singular values.

More often, however, one seeks to optimize the value of some function of the TN. In quantum physics for example, this means to maximize the overlap between some given state and a TN approximation or minimizing the energy expectation value with respect to some Hamiltonian to find its TN ground state. This corresponds to minimizing a loss function of a TN-based ML approach. The optimization can be achieved via several well established methods. In particular, general global gradient methods are available as well as TN-specific techniques that make use of the network's locality and the tensorial nature of the nodes.

*Renormalization methods* make use of the gauge ambiguity in TNs. They exploit the locality of operators to optimize the TN site by site. Density matrix renormalization group (DMRG), the first method of this kind, was developed to optimize spin chain Hamiltonians efficiently [6]. Soon, it was understood that restricting the maximum entanglement at each site reduces computational resources while describing lowly entangled chains very well [47] and further renormalization techniques were developed [48]. These provide powerful tools for optimizing MPS. Renormalization methods for TNs have been reviewed extensively before [8,11,14]. Hence, we will only sketch the basic idea of DMRG for a finite MPS here.

DMRG can be applied to Hamiltonians $H$ that consist of independent blocks connecting neighbouring MPS nodes. First, initialize a state randomly and consider the expectation value $\langle\Psi_0|H|\Psi_0\rangle$. Start with a block at one end of the chain and contract all other nodes to an environment tensor generating an effective Hamiltonian for the first site. Diagonalize the effective Hamiltonian and truncate its Hilbert space to the lowest (effective) eigenvalues. Subsequently iterating this procedure at each site will deterministically evolve the MPS towards the Hamiltonian's ground state.

Renormalization methods provide a local and fast way of optimization adapted to the structure of TNs but also have some disadvantages. First, DMRG is hard to implement in standard ML frameworks, especially when combining TNs and neuronal layers [49]. The algorithm has to be handcrafted for each problem [23]. Second, generalization to higher dimensions is possible [22,50] but not as efficient as for MPS due to entropy scaling [9].

*Global gradient methods* are standard optimization techniques that also apply to TNs. While using an overall global gradient usually is outperformed by renormalization methods, global methods make sense in special cases. In particular, renormalization methods have not been established yet for QTNs. Currently, stochastic gradient approximation methods [51] are employed in QML to circumvent the need for costly calculations of total gradients in high dimensional parameter spaces [52].

Global gradients have the downside that the gradient may vanish for random initial conditions in high dimensional parameter spaces. In QML, this is usually referred to as the *barren plateau* phenomenon [53] and is similar to the vanishing gradient problem known from classical ML [54].

The performance of gradient methods can be boosted by considering the special structure of TNs, e.g. with adapted initialization schemes [49]. Introducing locality either on the optimization routine or the loss can also mitigate barren plateaus (see §3).

*Geometric methods* make use of the network's underlying tensorial geometry. Tools from differential geometry can be used for analysing the TN on the space of entanglement patterns [55] and optimizing on loss manifolds [56]. This kind of optimization performs well on high dimensional parameter spaces, especially in combination with stochastic gradient descent [57] and auto-differentiation on individual nodes [58,59] or whole layers [23].

More advanced geometric methods reuse previous update steps. For this, their gradient vectors have to be transported along the optimization manifold [16]. However, they have to be applied in practice yet.

## (d) Classical machine learning with tensor networks

We already discussed in §2a that TNs are able to approximate high dimensional states within a regular, less complex structure. In ML, such states arise as maps of data features and as weight tensors that connect the data features to the desired result, e.g. a label in classification [60].

In principle, an ML algorithm seeks to find a function $f_l(x) : \mathcal{D} \to \mathcal{S}$ of some datum $x$ within the space of all possible inputs $\mathcal{D}$ that is mapped to a space of possible results $\mathcal{S}$, for instance a set of labels $l$. This function is called the model. Usually, the model is a composition of a data embedding $\Phi(x)$ and a trainable weight tensor $W_l$ connecting the embedded data to the output, as shown in figure 2*a*,*b*. The weight tensor $W_l$ can be approximated as a TN whose output represents the choice of labels (figure 2*c*). We get

$$f_l(x) = W_l \circ \Phi(x) \approx \langle W_{l,TN} | \Phi(x) \rangle, \tag{2.5}$$

where $\langle W^l_{TN}|$ is the TN approximation of the weight tensor. The dimensions of the weight tensor's index $l$ store the probabilities $P^{l_i}$ of the corresponding labels $l_i$

$$W^{\chi}_{l_i} \circ \Phi_\chi(x) = P_{l_i}(x). \tag{2.6}$$

Multi-class classifications are either done by training a single TN with large outgoing bond dimension or a set of networks with a single outgoing label bond each (one versus all). The data is embedded with a feature map that can transform the data before mapping it to the network [61]. This approach is very similar to encoding maps for QML [20] and can be approximated as TN as well.

A second way of embedding data into a feature space is using a density matrix $|\Phi(x)\rangle\langle\Phi(x)|$ and contracting it with a label dependent weight state $|W_l\rangle$. In this construction, the bond dimension $\chi$ is given directly by the non-vanishing eigenvalues of the covariance matrix [50] and the decision function is realized as the maximum overlap

$$f_l(x) = \mathrm{argmax}_l \langle W_l | \Phi(x) \rangle \langle \Phi(x) | W_l \rangle. \tag{2.7}$$

This construction has the advantage of being able to process incomplete data by contracting over missing bonds and can represent specific probability distributions based on the datasets [61].

Building generative TN models is also straightforward [62]. The goal of a generative ML model is to learn the distribution of its training data and to generate additional samples from this distribution. The simplest possibility is to use the dual of a trained classifier or a regressor obtained by adjoining all tensor nodes within the network.

Due to their quantum-inspired construction, TNs have the issue of not being able to copy information within their structure. This means that information cannot be distributed to different branches of a TN in a way a neural network uses information to activate its neurons for example. If for instance an operation in image analysis needs to use the value of adjacent pixels, one has to pass the same data into several input nodes by using overlapping observation windows [63]. However, this approach does not allow copying connected tensors to different locations.

Often, it makes sense to combine different layouts to use advantages of both. As an example, hierarchical layouts coarse grain the data and grid layers can be used to efficiently combine the information from different branches of the hierarchical TN [64]. The hierarchical part can be optimized with unsupervised ML methods where the ideal weight tensor is derived from the data covariance matrix [19]. It is even possible to add a TN layer to a neural network architecture, e.g. for complexity reduction in the input layer with MPS [65], MERA convolutional layers [66] or approximating a fully connected layer [67].

TN architectures are closely related to neural networks. Restricted [65,68] and deep [69] Boltzmann machines can be mapped to a two-dimensional TN consisting of MPS and matrix product operators, an operator valued version of MPS. Boltzmann machines, therefore, may be simulated using an MPS which allows for adjusting accuracy and execution time via the bond dimension allowing for a compression of neural network representations [16].

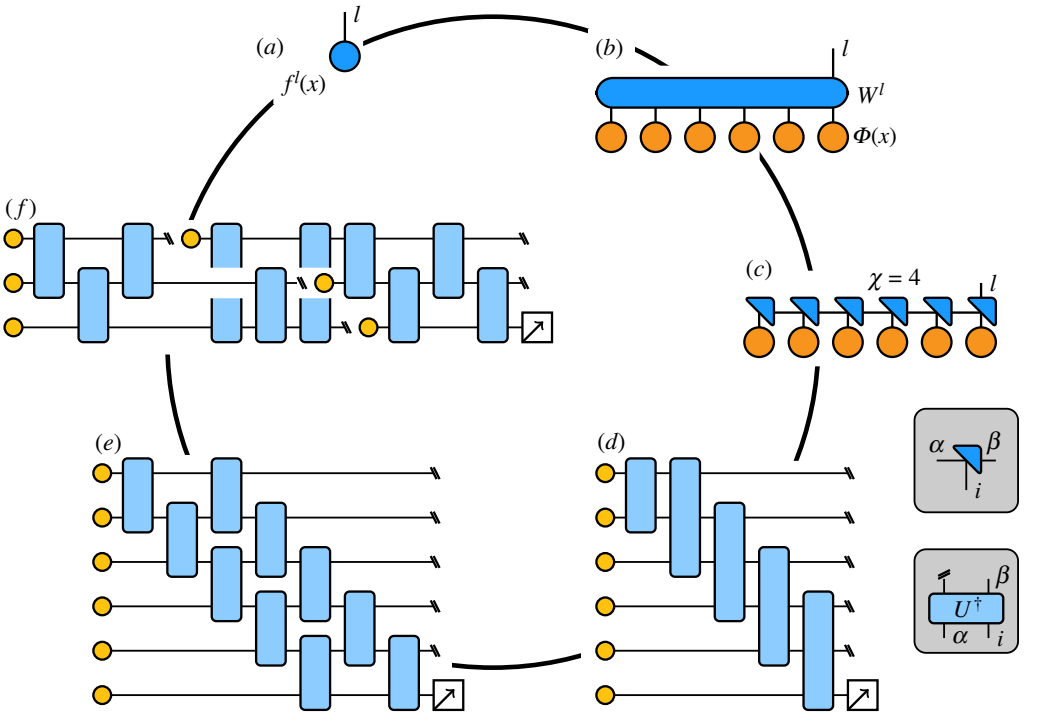


**Figure 2.** From a classical classifier to an efficient quantum tensor network. (*a*) Formally, the task of classification is performed by some function $f_l(x)$ which is an object, that accepts input data $x$ and outputs some label $l$. (*b*) In ML, one realizes the classification function $f_l(x) = \langle W_l | \Phi(x) \rangle$ with a weight tensor $W^l$ with trainable parameters where the (possibly transformed) input $\Phi(x)$ is fed into. The classification function is constructed as an overlap between both tensors. (*c*) In a TN approach, one decomposes the large tensor $W^l$ into a network of smaller tensors, e.g. by restricting the structure to a MPS layout. In this case, the bond dimension is $\chi = 4$. (*d*) By identifying isometric tensor nodes with unitary quantum gates (grey boxes), the MPS classifier can be mapped to a quantum computer. Higher bond dimensions between the tensor nodes require multi-qubit gates. In this case, the resulting circuit needs $\log \chi = 2$ internal qubits and three qubit gates. (*e*) The multi-qubit gates can be expressed by a repetition of the MPS two-qubit gate structure. Each additional internal qubit requires another layer of two-qubit gates. (*f*) If the quantum hardware supports resetting qubits during execution, a qubit efficient approach can be implemented reusing discarded qubits. The efficient circuit is a trade-off between qubit number and circuit length.

The map between both architectures has been exploited in both ways to compare specific network layouts. On the one hand, node numbers in an MPS representation of a Boltzmann machine will scale exponentially with the number of neurons [70] and recurrent neural networks can simulate MPS with reduced computational effort for certain cases [71]. On the other hand, hierarchical TNs efficiently implement convolutional or recurrent neural networks [72].

*Applications in ML* can be found for a wide variety of tasks. In image analysis, TN-based ML models are used for classification [50,73,74], compression [75] or feature extraction [66,76]. TN-based regressors have been successfully applied to nonlinear system identification [15] where the task is to generate a model of a nonlinear system from its behaviour.

Generative TN structures have been employed in anomaly detection [77] and as classifiers when reversing the generative TN structure [78]. A TN can be used to learn a probability distribution from data and a simulated 'measurement' of the TN state will generate a new instance from the distribution [79,80]. Generative TNs have also been applied to unsupervised feature identification in images [81].

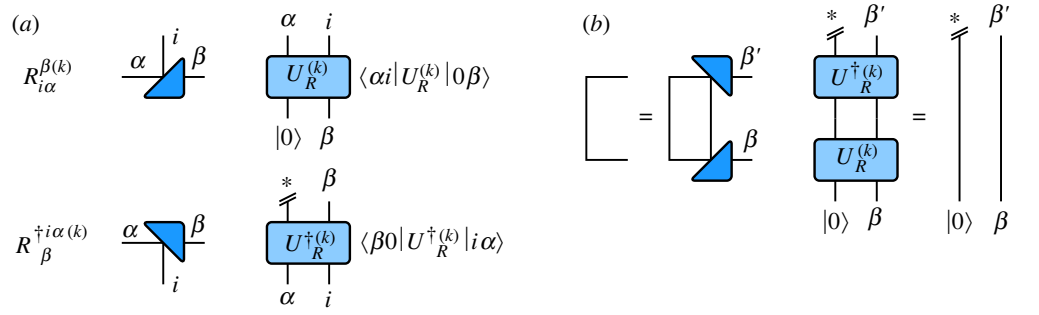


**Figure 3.** Mapping from isometric tensors to unitary quantum gates. (*a*) Whether a bond $\alpha, \beta$ is mapped to an incoming or outgoing qubit bundle depends on whether the tensor is given in right or left isometric form. The free bonds $i$ are represented by outgoing qubits. Adjoining a tensor flips its directions. Qubit preservation is taken into account by adding additional auxiliary qubits or discarding leftover ones. (*b*) Mapping the normalization condition for isometric tensors illustrates that discarding qubits actually has to fulfil a condition: for an exact representation of the classical network, discarded qubits have to be post-selected to $|0\rangle$ to be the dual of the auxiliary state.

## 3. Quantum tensor network machine learning

### (a) Mapping to quantum circuits

The quantum-inspired construction of TNs makes it straightforward to translate the concept to quantum computations. Tensor nodes are realized by multi-qubit gates with incoming and outgoing qubits carrying the bonds of the node.

The procedure for mapping the classical TN to a QTN is shown in figure 2*c*,*d*. Quantum gates are unitary, therefore, the corresponding TN has to be in canonical form with at least isometric nodes [82] (see §2b). Figure 3*a* shows how isometric tensors are mapped to gates. The bond dimension $\chi$ is determined by the number of qubits $n$ transferred between connected gates, i.e. $\chi = 2^n$. These qubits are called internal or virtual qubits. The qubits carrying the free (or physical) bonds are either forward or backwards directed, depending on whether the node has a vector or dual valued index. To preserve the number of qubits, the sum of all incoming qubits (free and internal) must equal the sum of the outgoing qubits at each gate. Therefore, one prepares necessary additional incoming qubits in a dummy state $|0\rangle$ or discards leftover outgoing qubits.

Discarded qubits usually are carried on unobserved, but a direct correspondence to classical TNs requires post-selection to a reference state $\langle 0|$ on these qubits [83]. From the normalization condition in figure 3*b*

$$\delta^{\beta}_{\beta'} = R^{\beta(k)}_{i\alpha} R^{\dagger i\alpha(k)}_{\beta'} \leftrightarrow \langle \beta' * | U^{\dagger(k)}_R U^{(k)}_R | 0\beta \rangle = \langle * | 0 \rangle \langle \beta' | \beta \rangle, \tag{3.1}$$

it follows that a post-selection measurement on $\langle *|$ is the counterpart of the auxiliary $|0\rangle$ initialization. This is caused by the fact that an isometry is mapped to a unitary and classical dimensional reduction or information loss has to be accounted for. Instead, one also can perform an uncomputation operation for each gate used [52]. For a network fully optimized on the quantum machine, the post-selection requirement can be released which allows for hybrid methods [83,84] or efficient layouts where discarded qubits can be reset and reused [62].

Using this recipe, one can map the TN layouts known from §2b to quantum circuits. Figure 4 shows a central gauge MPS and a TTN. More advanced networks like a brickwall, MERA and a square PEPS are shown in figure 5. Mapping these networks to a quantum computer gets more and more involved with growing bond dimension and requires larger circuits with high connectivity (or many swap gates) between the qubits.

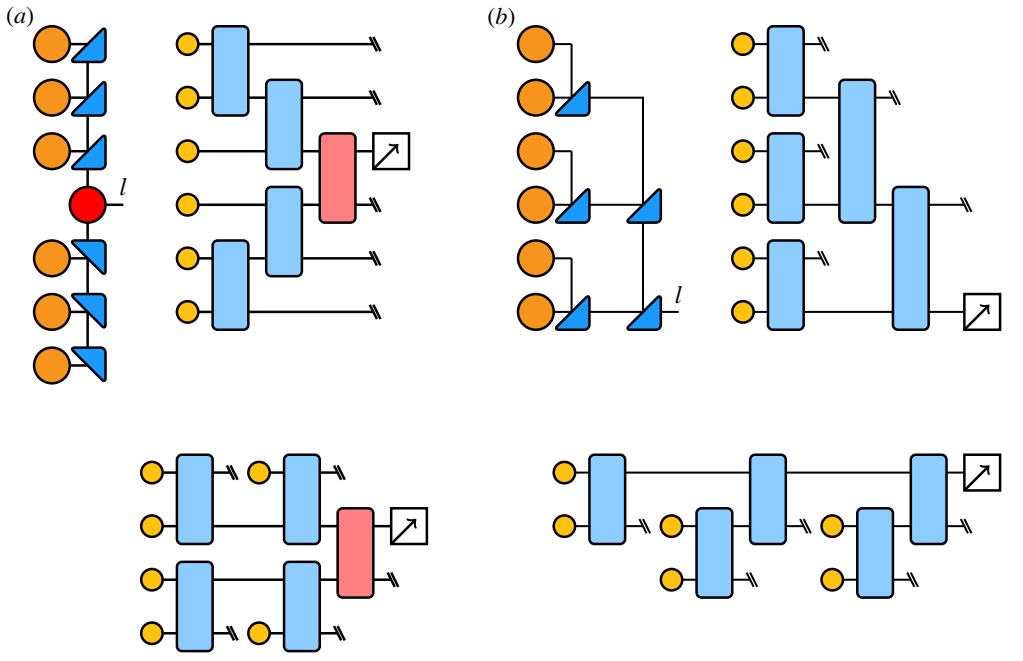


**Figure 4.** Simple tensor networks and their quantum counterparts. Single qubit unitary gates are omitted as they can be absorbed into an adjacent two-qubit unitary. Each bond may be realized by one or more qubits. (*a*) A MPS in site canonical form and its quantum implementation. Choosing a central gauge halves the circuit depth compared with the left canonical MPS from figure 2. However, the central node is not isometric in general and can only be mapped to the unitary quantum gate approximately. If the quantum computer supports resetting qubits during execution, a qubit efficient approach can be implemented reusing discarded qubits with constant qubit number. (*b*) A TTN offers higher entanglement than a MPS, but its qubit efficient quantum representation will need a total of log *n* qubits for *n* inputs.

## (b) Efficiently implementing quantum tensor networks

Large circuits, multi-qubit gates and gates not using the standard gate set are hard to implement on near-term noisy intermediate scale quantum (NISQ) computers. To reduce circuit complexity, several approaches exist. A major step in bringing TNs to quantum computers was the development of a breakdown method for multi-qubit nodes to two-qubit unitaries with high fidelity [85,86], which can be implemented on NISQ devices efficiently. This approach has been based on a classical procedure for photonic qubits [87]. The approach is shown for an MPS in figure 2*d*,*e*: this MPS has left canonical gauge and therefore has a quantum gate equivalent looking like a staircase of multi-qubit gates. The size of the gates is given by the number of internal qubits $n = \log \chi$ that have to be passed on to the next gate (*d*). The three qubit gates in this example may be replaced by two layers of two-qubit gates which provide the same connectivity between adjacent incoming free qubits (*e*). Each additional internal qubit would add another layer of two-qubit gates to the circuit.

The most general ansatz for the gates within a tensor node is a full unitary gate [62,88]. Representing these gates with simple gates available on a NISQ device however results in long circuits that are prone to noise. Therefore, simplified ansaetze for the two-qubit gates are commonly used [35,40,84,89,90]. For quantum input, the performance of these simplified ansaetze can yield comparable maximum performance to a general unitary ansatz, but they seem to be harder to train. For classical data, the performance was much lower using simple nodes. This holds for both grid [37] and hierarchical layouts [52].

To reduce the total qubit count, the structure of many TNs allows for an efficient reordering of its blocks, such that discarded qubits can be reset and reused for the input of new information

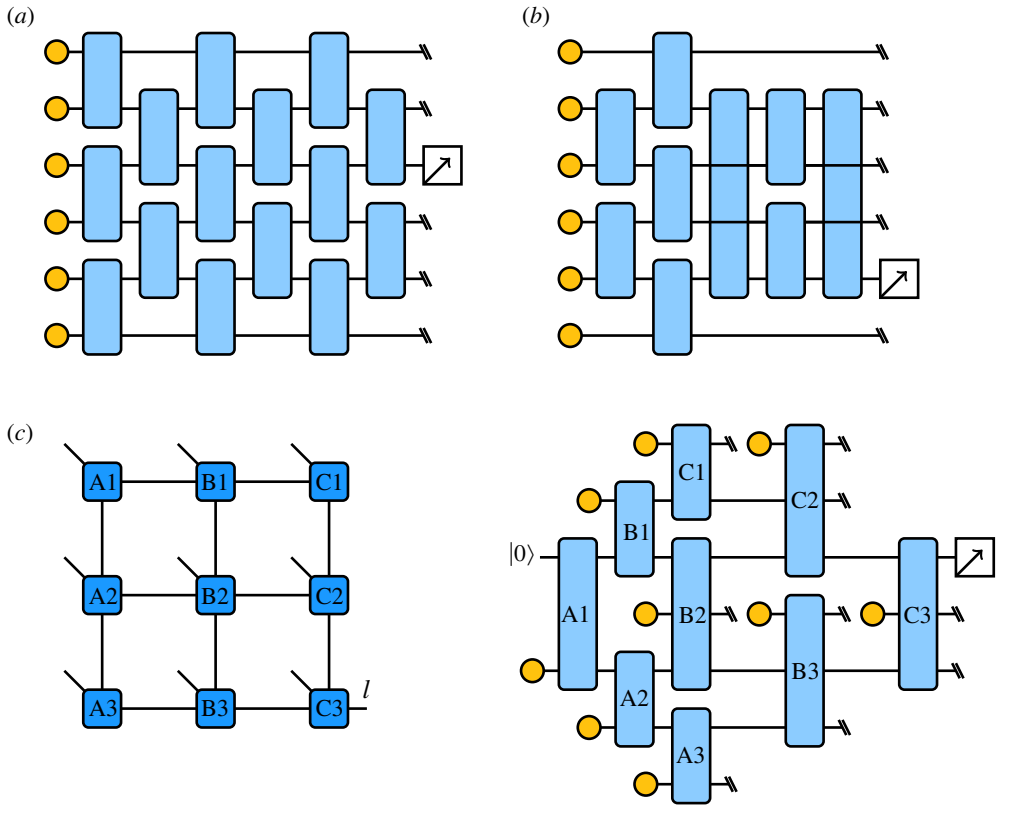


**Figure 5.** Higher dimensional quantum tensor network structures. (*a*) The brickwall architecture offers a higher amount of entanglement than MPS and can be seen as a derivative of hexagonal projected entangled pair states. A brickwall allows for the representation of every MPS gauge up to a bond dimension given by the depth of the circuit. (*b*) The multi-scale renormalization ansatz (MERA) quantum network requires gates between qubits further apart which may be realized by introducing swap gates in between on current hardware. Both brickwall and MERA do not allow for qubit efficient implementations. (*c*) The quantum circuit of pair entangled product states (PEPS) heavily depends on the order in which the PEPS nodes are evaluated. The realization will feature coupled staircase structures similar to MPS. Here, a qubit efficient approach scales linearly with the length of the diagonal.

(figure 2*f*). A qubit efficient MPS only requires a constant amount of qubits determined by the dimension of the inputs and the desired bond dimension. For a TTN, the qubit number scales logarithmically with the size of the input.

Using the qubit efficient approach may not have an effect on the optimized model parameters because the circuit is trained to carry on the label information and the 'no signalling' principle, therefore, forbids an influence of these discarded qubits on the result [62]. Instead of simply resetting, the information in the discarded qubits can be used for quantum error correction within the nodes [91] which improves performance on NISQ devices. In general, the influence of the qubit efficient procedure, e.g. on trainability is still not clear. When combined with a local loss however, no barren plateaus arise in the error correcting ansatz [92].

Combining these simplifications reduces both qubit number and gate complexity [85]. The overall circuit depth is harder to reduce. Choosing a central gauge for MPS at least halves circuit depth compared with left or right gauges [84] (figure 4).

## (c) Variational machine learning with quantum tensor networks

Recently, a wide variety of ML architectures employing VQCs have been developed. A VQC is a quantum circuit whose gates have tunable parameters. General unitaries can be constructed

from a combination of rotation and entangling gates like the CNOT gate. A common architecture for QML is a layered VQC. Here, the circuit consists of encoding blocks that map the data to the circuit and parametrized variational blocks which entangle the qubits. To increase the expressivity of the quantum circuit, these blocks can be repeated before the measurement [20].

A QTN with tunable gates is also a variety of VQC with an internal TN layout. The structure of TNs provides several advantages for QML. First of all, insights from the available theory on classical TNs also apply to their quantum counterparts. Due to the direct correspondence, data and models from classical TNs can be translated to QTNs and vice versa. This can be used to better initialize quantum models (see §3d). Furthermore, the choice of a specific TN layout allows for the introduction of inductive bias, e.g. knowledge about the type of data and therefore the construction of a QML structure that will fit the data well. Finally unlike for general VQC algorithms, the space of possible weights in TN-based ML can be adjusted easily by varying the bond dimension. This allows for tuning the expressivity of the circuit to mitigate under- and overfitting [62]. It is not clear yet how the expressivity of a QTN scales or compares with layered VQC approaches but both architectures can be mapped onto each other [93].

Until now, the development of QTN ML approaches focuses on supervised classificators and generators. Supervised learning with QTNs works similarly to the classical approach shown in §2d. Examples for QML circuits based on different layouts are shown in figures 4 and 5. The data is mapped to the quantum computer using some feature map $\Phi(x)$ which is shown as orange dots in the images. The feature map may be a TN itself (see §4). The weight tensor $W_l$ is represented as the blue quantum TN. In the end, a measurement on the remaining qubits yields a result, e.g. a classification. If a multi-class output is needed, introducing an exit node (figure 7*b*) will improve the fraction of correct classifications [88].

Generative TNs can be realized by reversing the TN structure. The inputs of the generative network are given by some reference computational basis state which are entangled by the TN (figure 7*a*). These generative networks can be trained either by sampling the generative QTN and comparing the results with a given training set [62] or by training a classifier and adjoining every gate as noted in §2d.

Some studies already include an investigation of the influence of noise on the QTN circuit. Numerical results indicate that low level noise is not a problem for classification [52,62]. It even may be used to enhance the performance of the algorithm by adding auxiliary qubits initialized with noise to the circuit. This effectively generates a probabilistic model which is easier to train. However, if the noise is too high this also leads to decoherence rendering the circuit unfunctional [94].

Optimizing the parameters of these QTNs relies on some variety of global gradient descent for the majority of literature. Geometric [83] or genetic methods [95] are used only rarely. Renormalization methods like for classical TNs have not been adapted to QTNs but may be employed in hybrid methods [36]. Some proposals even consider employing TNs for optimizing parameters or hyperparameters of QML algorithms [96]. For specific implementations, first evidence exists that the locality of TNs can overcome barren plateaus [97,98]. Especially the use of local loss functions, which can be implemented using local Hamiltonians, provides a favourable loss landscape without gradients vanishing exponentially fast [92,99,100]. The same approach may also reduce the amount of training data needed [40]. Furthermore, recent results suggest that generative hierarchical QTNs can mitigate the exponentially vanishing gradients as single qubit observables causally depend only on a logarithmic number of gates [101]. This coincides with findings using ZX-calculus that MPS and layered VQCs will experience barren plateaus, but hierarchical TNs will not as their variance does not vanish exponentially with the system size [100].

## (d) Hybrid training

Hybrid QTN architectures combine quantum and classical elements to use the advantages of both worlds. Compared with NISQ devices, classical computers are able to perform computations on

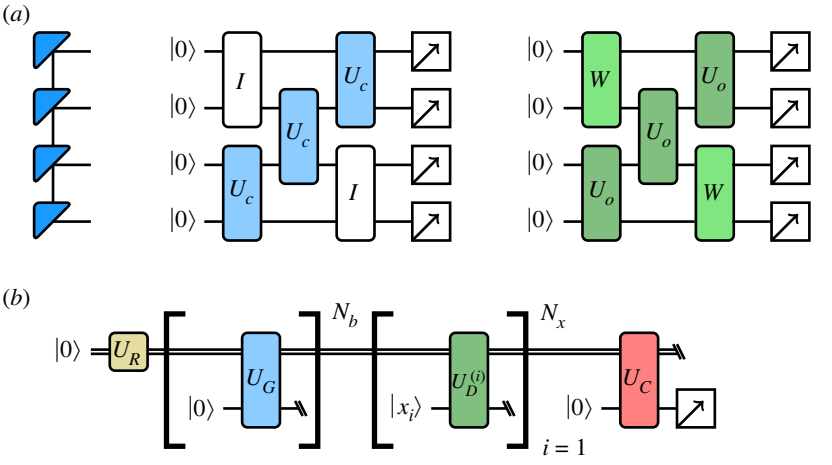


**Figure 6.** Hybrid training methods for quantum tensor networks. A pre-trained classical TN provides suitable initial values for further optimization on a quantum computer. The direct approach may be refined to make training easier or to have access to a larger part of the multi-qubit Hilbert space. Method (*a*) [84] maps a classically optimized MPS to the diagonal gates $U_c$ of a brickwall ansatz; all off-diagonal gates are initialized as identities. Then a second optimization step on the quantum computer is conducted. While the optimized diagonal gates $U_o$ have to be full unitary gates to enable the transfer from the classical network, the off-diagonal gates $W$ can use a simpler ansatz with fewer parameters. Approach (*b*) [83] maps two classical MPS to the quantum circuit. A homogeneous MPS ($N_b$-times $U_G$) truncated to an appropriate boundary condition $U_R$ prepares an initial state. In the second MPS, the nodes $U_D^{(i)}$ upload and process the $N_x$ elements of the datum $x$. Finally, the classification is performed on the exit node $U_C$. Method (*a*) is shown as a generator, method (*b*) as an efficient classifier.

far larger datasets and their use is very cheap. The quantum part of the algorithm may introduce some qualitative quantum advantage like higher maximum performance or generalization of the model. At the moment, two hybrid strategies make use of these characteristics. First, the classical reduction of the input data's dimensionality with pre-processing like PCA [84], auto-encoders or TN-based encodings discussed in §4. If the classical part is trainable, it may be optimized together with the subsequent QTN. Second, the direct maps between TNs and their quantum counterparts allow for classical pre-training of the quantum model's initial values. Even when more powerful quantum computers are available, the execution of quantum circuits will still be expensive and pre-training methods to reduce the number of quantum circuit executions will stay relevant. In this section, we will focus on hybrid pre-training methods.

As discussed in §3a, a TN in canonical form can be mapped exactly to a quantum computer. This allows to train a coarse classical TN model which can be refined and expanded after mapping it to a quantum computer. Any standard QTN layout may be prepared with classically prepared initial conditions [50,61] and for providing efficient initial values, no post-selection on the quantum computer is required [83]. Using these initial values for the QTN's parameters makes the training of larger quantum circuits far more efficient in comparison with random or identity initialization schemes. The main benefit is that the initial training phase, where the gradients decrease exponentially with the qubit number, already has been performed classically and therefore the training on the quantum device starts in a favourable spot of the parameter space [84].

Modifications to the basic procedure of classically pre-training a QTN have been developed to lower the requirements on the classical preparation and to make the quantum part easier to train. For training a brickwall layout (figure 5*a*), it is sufficient to prepare an initial MPS state that is embedded within the brickwall, e.g. the diagonal, and the remaining gates start as identity gates [84] as shown in the centre panel of figure 6*a*. To make quantum training easier, one does not have to use full unitary gates $U$ on the whole circuit, but can restrict the off-diagonal gates to some simple ansatz $W$ as shown in the right panel of this figure. In principle, any other layout

than a MPS also can be embedded into the brickwall. As their QTN pendants are more efficiently trainable [101], this may result in further improvement over MPS. The pre-training approach can be seen as a quantum version of the copy node initialization for classical TNs, where most of the tensor nodes are initialized with identity tensors [49]. It also shares similarities with transfer learning inspired quantum pre-training approaches that are used for layered VQCs [102]. These may be transferred to QTNs by preparing sub-networks or networks with lower bond dimension on a quantum computer first.

Another modification considers preparing a prior distribution within the feature space before uploading the data. This reduces the bond dimensions and gate complexity needed [83]. Figure 6*b* shows the approach for an efficient MPS classifier. At the beginning of the circuit, a homogeneous MPS $U_G^{N_b}$ of length $N_b$ prepares the prior distribution. Setting a trainable boundary condition $U_R$ reduces the number of nodes the MPS needs to represent an effective prior. The second part $U_D^{(i)}$ is a standard efficient MPS similar to figure 2*f*, where the $N_x$ data features are introduced into the QML and an output node $U_C$ prepares the classification result in the end. The circuit technically can be optimized without classical pre-training. But for higher bond dimensions, this construction is far easier to train having initial values obtained with classical TN-specific methods like DMRG [83].

Using pre-training methods with embedded sub-networks might lead the full circuit only into some local minimum. To assess the impact of this issue, one would need to investigate the structure of loss landscapes of QTNs. As this is not feasible for large applications, additional mitigation techniques could help finding global minima, e.g. by using local or blockwise optimization techniques.

### (e) Case studies and implementations

The application of QTN ML methods has been limited to demonstrative feasibility studies up to now. Most authors focus on classification tasks for image recognition either with binary classes [40] or multi-class set-ups [88]. One implementation of binary image classification [62] has been performed on real photonic hardware [103]. Other uses are classifications on parameterized classical data [104] and on quantum simulation results [35,37]. Besides the proof of concept, these studies demonstrate that QTN approaches already can process relatively high dimensional input data like greyscale images of up to $37^2$ pixels. They show that QTNs can achieve accuracies for the classification of both classical and quantum datasets in the range of 0.85–0.95 with only a small amount of parameters and internal qubits.

The application of QTNs for a regression of continuous properties has not been discussed widely yet. One proposition for this application is to approximate eigenvectors of unitary matrices [105], but finding the right bond dimension is crucial to find an approximate state having sufficient overlap with the real eigenvector without using huge circuits.

QTN generators have been implemented by various authors to provide quantum state samples from learned distributions [61,62,84] as a feasibility study.

Most case studies that use publicly available frameworks rely on the Qiskit [106], as it supports resets in the middle of an execution, which are necessary for efficient TNs. For ML, Qiskit is compatible with the pytorch framework. Cirq [107] also provides a reset functionality and integrates with the tensor flow ML suite. Pennylane [108], which focuses on QML applications, currently cannot implement mid-circuit measurements for efficient QTNs, but provides methods for both basic MPS and TTN-based quantum classifiers. The implementation allows for varying virtual qubit bonds and connects the quantum circuits to most common ML frameworks in Python.

## 4. Tensor networks for data encoding

For the performance of data-driven quantum algorithms and QML algorithms in particular, encoding of data plays a crucial role. Current quantum computing hardware provides neither

a sufficient amount of qubits nor gate depth to encode high dimensional datasets in a straightforward fashion. However, this does not necessarily mean that the problem size to be tackled with current quantum algorithms has to be small. Instead, one relies on classical and quantum pre-processing steps that reduce the data to its essential features.

By adjusting the bond dimension, TNs provide a direct way of compressing data both lossless by discarding dimensions with singular values equal to zero and approximate by setting upper bounds on the bond dimension. The input QTN state can be prepared in at least five different ways. First, by maximizing the overlap between a classical representation of a quantum state and a TN. Second, by encoding classical data in a TN and reducing its bond dimensions by tensor decompositions. In both approaches, the network will then be mapped to quantum circuits as shown in figure 7*a* for MPS and TTN. Efficient mapping methods are also available for PEPS [109]. A third classical method is building a classical auto-encoder from two TNs to compress the data into a latent representation. The bond dimension at the encoder's output nodes gives the size necessary for the quantum circuit and its state can be mapped directly to the circuit. On a quantum computer, one can directly maximize the overlap between an existing quantum state and a QTN. This requires preparing the reference quantum state multiple times until convergence which may be very costly. Finally, one can train a generative network to output a state from some distribution (see §3c). Therefore we focus on classical pre-processing in this section.

Encoding data in TN-based quantum layouts promises some benefits over classical encoding. Especially, the access to a large state space even when using few qubits could boost efficiency of information storage. For example, the number of parameters needed to represent certain time evolutions of quantum states is exponentially reduced with QTNs compared with classical ones [86].

Depending on the type of data, different layouts of TNs provide the most efficient storage because the scaling of their mutual information has to match the scaling behaviour of entanglement in the TN. As discussed in §4, MPS suit one-dimensional data like time series and logarithmic TTNs or two-dimensional TNs like MERA or PEPS are better suited for images depending on the amount of local correlation. For text, the information scales even steeper [110], which requires three-dimensional PEPS or high dimensional MERA variants which have not been implemented on a quantum computer yet. Exploiting symmetries reduces the need for complexity within the structure, e.g. by using wavelet transform techniques in images [111].

Nevertheless, MPS and TTN can be implemented and optimized easily and still provide an improvement over direct encoding methods. They are, therefore, widely used in encoding for QML. The performance of MPS and TTN can be improved by combining them with other methods. When encoding images, one can split the whole image into patches and encode each patch into an MPS (figure 7*c*) which will catch local entanglement better but requires more storage. For a fixed bond dimension, the number of qubits is proportional to the number of patches encoded. The pixels in each patch are addressed by a method that is known as flexible representation of quantum images [112]. The method was developed as a classical compression method [113] and has recently been transferred to quantum computers [65,85]. Patchwise MPS encoding can be easily combined with MPS QML methods (figure 7*b*) [88].

Trainable TN encoding using a latent space representation from the outgoing bond dimensions usually is optimized together with the parameters of the QML circuit [40,114]. A theoretical study on the error performance of function regression models finds upper bounds when certain continuity requirements on the loss and the network are met [98]. Particularly, they find that the optimization error connected to barren plateaus will be negligible if the loss on the TN parameters is Lipschitz and satisfies a Polyak–Lojasiewicz condition. However, they do not develop a method to set up a TN that actually fulfils these conditions. Trainable encoding can be improved by a patchwise approach, too. Applying trainable MPS approximators on small regions of the image yields a linear model of the image where the spatial information is stored in the feature space [75]. Due to the independence of the various layers, this method also could be realized with a hybrid circuit, where the initial layers are classical and the final layers are quantum.

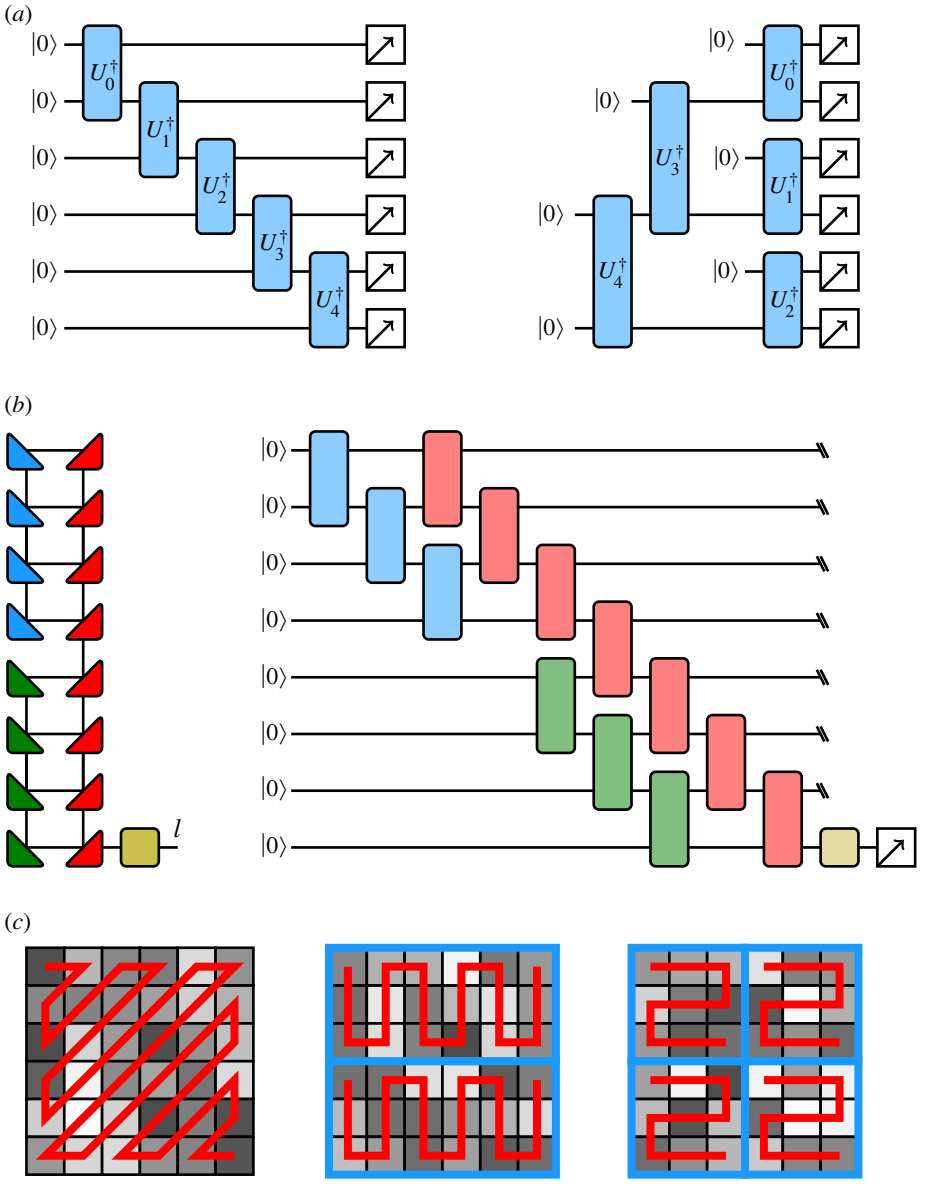


**Figure 7.** Encoding strategies for ML using tensor networks. (*a*) Encoding classical or compressing quantum data using MPS (left) or a TTN (right). Each gate $U_i^\dagger$ is a direct mapping from an isometric node of a classical tensor network. A generative quantum tensor network has the same structure as an input state, but with one or more $|0\rangle$ input qubits replaced by a label or noise encoded input. (*b*) Data may be encoded in several independent MPS (blue, green) and fed into the circuit to reduce information loss. A quantum ML algorithm can make use of the same MPS structure (red) and directly connect to the encoding MPS. An additional output gate (yellow) improves classification accuracy for multi-class tasks. (*c*) To encode two-dimensional data like images into MPS, one needs to choose a one-dimensional path either at the cost of losing parts of the information or introducing high bond dimensions. Cutting the area into patches improves the encoding result as this reduces the maximum distances on the MPS between neighbouring sites in the original data.

TN encoding pairs well with TN-based ML but it is applicable to any other QML approach. For layered VQC approaches, first results imply that TN pre-processing trained together with the VQC classifier performs better than regular PCA on image data [114] and can be used as an estimator for the Q-value function of a reinforcement learning ansatz [95].

TN encoding is not only relevant for QML, but can be used to provide states for any other quantum application that requires complex input. For example, overlaps of QTN generated basis functions can be used to approximate nonlinear functions [36]. This approach may reduce the number of grid points needed in quantum simulations with nonlinear PDEs as couplings compared with the classical approach.

## 5. Conclusion

TNs have proven themselves useful for storing and processing quantum states as well as for classical ML applications. Combining both aspects makes them a suitable tool for QML as well. We have seen in §§3 and 4 that TNs can be employed for various tasks within the QML pipeline, from pre-processing and encoding to the variational part and the optimizers [96]. They have a very flexible representation as they allow for both pure quantum algorithms and classical-quantum hybrids while a wide range of optimization methods can be applied.

Bringing TNs to a quantum computer has advantages considering architecture design. The representation of a quantum state with TNs on a quantum computer reduces the necessary amount of qubits compared with other encoding methods [29]. Thereby TNs provide an efficient way of mapping classical data to quantum applications. The tensors of a QTN do not have to be contracted costly as on classical hardware since the contraction happens as part of the execution of the quantum circuit. When using hybrid approaches, TNs allow for a seamless connection between classical and quantum methods enabling pre-training and gradual tuning of the border between both systems—which will become important when the power of NISQ devices scales up significantly. Having the possibility of choosing a qubit efficient implementation is also a very important feature although its effects on trainability are not yet fully understood and require further investigation.

In comparison with classical TNs, QTNs are expected to provide several benefits for the algorithms themselves. As quantum algorithms naturally implement entanglement, QTNs will have access to a Hilbert space that grows exponentially with the number of qubits. This enlarges storage capacity and the available parameter space for QML algorithms. While classical TNs are able to represent only low-entangled, low-complexity states, QTNs have access also to low-complexity states that can be generated by Hamiltonian time evolution [86] independent of the amount of entanglement. However, this may need very large circuit depths. Additionally, QTNs provide a natural way of using complex numbers instead of real ones which reduces the number of parameters necessary greatly in certain architectures [68]. It is yet unclear this is a general advantage of QTNs.

Regardless of that, QTNs seem to be easier to train than other QML methods. For example, using local optimization routines that make use of the localized TN structure can help to overcome problems like barren plateaus and reduce the amount of training data needed. However, these results have been obtained using specific implementations, some combined with special features like error correction. The results are, therefore, not generalizable to all QTN layouts yet. Choosing a layout that fits the data structure well also can reduce the need for large general circuits that are hard to train due to their large amount of parameters.

The mentioned actual and possible benefits come with downsides compared with classical networks. Gates are directed and reshaping the network cannot be performed in a straightforward way. The usual difficulties with quantum computations like encoding classical data and the need to perform non-reversible measurements to obtain a result still apply. Moreover, compared with more general QML methods like layered VQCs, the strict structure of a TN layout may render it an architecture which cannot be applied on general problems but has to be handcrafted each time. Therefore it is unclear at the moment, whether the benefits of TNs really can be translated to a relevant quantum advantage outside the laboratory.

Although QTNs have the potential to be a successful framework for QML, their development has just begun and further research is needed in many directions. Modifications to the basic layouts like variable bond dimensions which can be used to reduce computational costs have

not been adapted to QTNs yet. In particular, a quantum version of TN-specific local optimization methods is interesting for building algorithms that can be trained more easily.

However, most important is a more fundamental insight into the capabilities of QTNs—especially in comparison with classical and other VQC architectures. This includes methods to assess ML performance theoretically on the layout level and not just for specific implementations. In our opinion, a thorough understanding of the theoretical background of TNs will enable us to identify the range of application where it makes sense to use a QTN architecture and provide design principles for application-specific QTN layouts. In particular, we suggest three areas where future research is needed most: first, transferring insights from classical TN theory to the quantum world, e.g. optimization procedures or insights on scaling behaviour. Second, finding general measures of the capabilities of QTNs, especially for expressivity, generalization and trainability. Third, investigating the ideal connection for quantum-classical hybrid TN architectures to make ideal use of both computing paradigms.

Data accessibility. This article has no additional data.

Authors' contributions. H.-M.R.: conceptualization, investigation, visualization, writing—original draft; F.K.: funding acquisition, resources; A.P.R.: supervision, writing—review and editing.

All authors gave final approval for publication and agreed to be held accountable for the work performed therein.

Conflict of interest declaration. We declare we have no competing interests.

Funding. No funding has been received for this article.

Acknowledgements. The authors thank Bogusz Bujnowski, Lautaro Hickmann, Markus Lange and Pia Siegl for our discussions on classical TNs, QTNs and ML and their very helpful remarks on this review. Furthermore, we would like to thank the anonymous reviewers for their helpful comments on our manuscript.

## References

1. Shor PW. 1994 Algorithms for quantum computation: discrete logarithms and factoring. In *Proc. 35th Annual Symposium on Foundations of Computer Science, Santa Fe, NM, 20–22 November*, pp. 124–134. IEEE Computer Society Press, USA. (doi:10.1109/SFCS.1994.365700)
2. Caro MC, Huang HY, Cerezo M, Sharma K, Sornborger A, Cincio L, Coles PJ. 2022 Generalization in quantum machine learning from few training data. *Nat. Commun.* **13**, 4919. (doi:10.1038/s41467-022-32550-3)
3. Abbas A, Sutter D, Zoufal C, Lucchi A, Figalli A, Woerner S. 2021 The power of quantum neural networks. *Nat. Comput. Sci.* **1**, 403–409. (doi:10.1038/s43588-021-00084-1)
4. Perrier E, Youssry A, Ferrie C. 2022 Qdataset, quantum datasets for machine learning. *Sci. Data* **9**, 582. (doi:10.1038/s41597-022-01639-1)
5. Boixo S, Isakov SV, Smelyanskiy VN, Babbush R, Ding N, Jiang Z, Bremner MJ, Martinis JM, Neven H. 2018 Characterizing quantum supremacy in near-term devices. *Nat. Phys.* **14**, 595–600. (doi:10.1038/s41567-018-0124-x)
6. White SR. 1992 Density matrix formulation for quantum renormalization groups. *Phys. Rev. Lett.* **69**, 2863. (doi:10.1103/PhysRevLett.69.2863)
7. Östlund S, Rommer S. 1995 Thermodynamic limit of density matrix renormalization. *Phys. Rev. Lett.* **75**, 3537. (doi:10.1103/PhysRevLett.75.3537)
8. Bridgeman JC, Chubb CT. 2017 Hand-waving and interpretive dance: an introductory course on tensor networks. *J. Phys. A: Math. Theor.* **50**, 223001. (doi:10.1088/1751-8121/aa6dc3)
9. Cirac I, Perez-Garcia D, Schuch N, Verstraete F. 2021 Matrix product states and projected entangled pair states: concepts, symmetries, and theorems. *Rev. Mod. Phys.* **93**, 959. (doi:10.1103/RevModPhys.93.045003)
10. Evenbly G, Vidal G. 2009 Algorithms for entanglement renormalization. *Phys. Rev. B* **79**, 144108. (doi:10.1103/PhysRevB.79.144108)
11. Schollwöck U. 2011 The density-matrix renormalization group in the age of matrix product states. *Ann. Phys.* **326**, 96–192. (doi:10.1016/j.aop.2010.09.012)
12. Cichocki A *et al.* 2016 Low-rank tensor networks for dimensionality reduction and large-scale optimization problems: perspectives and challenges part 1. *Found. Trends Mach. Learn.* **9**, 249–429. (doi:10.1561/2200000059)
13. Vanderstraeten L, Haegeman J, Verstraete F. 2019 Tangent-space methods for uniform matrix product states. *Sci. Post Phys. Lect. Notes* **7**, 1–77. (doi:10.21468/SciPostPhysLectNotes.7)

14. Orús R. 2019 Tensor networks for complex quantum systems. *Nat. Rev. Phys.* **1**, 538–550. (doi:10.1038/s42254-019-0086-7)
15. Batselier K. 2022 Low-rank tensor decompositions for nonlinear system identification: a tutorial with examples. *IEEE Control Syst.* **42**, 54–74. (doi:10.1109/MCS.2021.3122268)
16. Cichocki A, Lee N, Oseledets I, Phan AH, Zhao Q, Mandic DP. 2016 Tensor networks for dimensionality reduction and large-scale optimizations. Part 2 applications and future perspectives. *Found. Trends Mach. Learn.* **9**, 249–429. (doi:10.1561/2200000067)
17. Levine Y, Sharir O, Cohen N, Shashua A. 2023 Bridging many-body quantum physics and deep learning via tensor networks. In *Mathematical aspects of deep learning* (eds P Grohs, G Kutyniok), pp. 439–474. Cambridge, UK: Cambridge University Press.
18. Liu D, Yao Z, Zhang Q. 2020 Quantum-classical machine learning by hybrid tensor networks. (http://arxiv.org/abs/quant-ph/2005.09428)
19. Stoudenmire EM. 2018 Learning relevant features of data with multi-scale tensor networks. *Q. Sci. Technol.* **3**, 034003. (doi:10.1088/2058-9565/aaba1a)
20. Schuld M, Petruccione F. 2021 *Machine learning with quantum computers*. Cham, Switzerland: Springer International Publishing.
21. Kolda TG, Bader BW. Sandia National Laboratories.2018 Tensor decompositions: applications. In *Algorithmic aspects of machine learning* (ed. A Moitra), pp. 48–70. Cambridge, UK: Cambridge University Press.
22. Zaletel MP, Pollmann F. 2020 Isometric tensor network states in two dimensions. *Phys. Rev. Lett.* **124**, 037201. (doi:10.1103/PhysRevLett.124.037201)
23. Geng C, Hu H-Y, Zou Y. 2022 Differentiable programming of isometric tensor networks. *Mach. Learn.: Sci. Technol.* **3**, 015020. (doi:10.1088/2632-2153/ac48a2)
24. Zhou Y, Stoudenmire EM, Waintal X. 2020 What limits the simulation of quantum computers?. *Phys. Rev. X* **10**, 014038. (doi:10.1103/PhysRevX.10.041038)
25. Nguyen T, Lyakh D, Dumitrescu E, Clark D, Larkin J, McCaskey A. 2023 Tensor network quantum virtual machine for simulating quantum circuits at exascale. *ACM Trans. Quantum Comput.* **4**, 1–21. (doi:10.1145/3547334)
26. McCaskey A, Dumitrescu E, Chen M, Lyakh D, Humble T. 2018 Validating quantum-classical programming models with tensor network simulations. *PLoS ONE* **13**, e0206704. (doi:10.1371/journal.pone.0206704)
27. Guo C, Modi K, Poletti D. 2020 Tensor network based machine learning of non-markovian quantum processes. *Phys. Rev. A* **102**, 062414. (doi:10.1103/PhysRevA.102.062414)
28. Pednault E, Gunnels JA, Nannicini G, Horesh L, Magerlein T, Solomonik E, Draeger EW, Holland ET, Wisnieff R. 2017 Pareto-efficient quantum circuit simulation using tensor contraction deferral. (http://arxiv.org/abs/1710.05867)
29. Barratt F, Dborin J, Bal M, Stojevic V, Pollmann F, Green AG. 2021 Parallel quantum simulation of large systems on small NISQ computers. *npj Quantum Inf.* **7**, 79. (doi:10.1038/s41534-021-00420-3)
30. Jaschke D, Montangero S. 2023 Is quantum computing green? An estimate for an energy-efficiency quantum advantage. *Quantum Sci. Technol.* **8**, 025001. (doi:10.1088/2058-9565/acae3e)
31. Alcazar J, Vakili MG, Kalayci CB, Perdomo-Ortiz A. 2021 Geo: enhancing combinatorial optimization with classical and quantum generative models. (http://arxiv.org/abs/2101.06250)
32. Mugel S, Kuchkovsky C, Sanchez E, Fernandez-Lorenzo S, Luis-Hita J, Lizaso E, Orus R. 2022 Dynamic portfolio optimization with real datasets using quantum processors and quantum-inspired tensor networks. *Phys. Rev. Res.* **4**, 77. (doi:10.1103/PhysRevResearch.4.013006)
33. Cavinato S, Felser T, Fusella M, Paiusco M, Montangero S. 2021 Optimizing radiotherapy plans for cancer treatment with tensor networks. *Phys. Med. Biol.* **66**, 125015. (doi:10.1088/1361-6560/ac01f2)
34. Sierra G, Martin-Delgado MA. 1998 The density matrix renormalization group, quantum groups and conformal field theory. (http://arxiv.org/abs/COND-MAT/9811170)
35. Uvarov A, Kardashin A, Biamonte J. 2020 Machine learning phase transitions with a quantum processor. *Phys. Rev. A* **102**, 012415. (doi:10.1103/PhysRevA.102.012415)
36. Lubasch M, Joo J, Moinier P, Kiffner M, Jaksch D. 2020 Variational quantum algorithms for nonlinear problems. *Phys. Rev. A* **101**, 451. (doi:10.1103/PhysRevA.101.010301)

37. Lazzarin M, Galli DE, Prati E. 2022 Multi-class quantum classifiers with tensor network circuits for quantum phase recognition. *Phys. Lett. A* **434**, 128056. (doi:10.1016/j.physleta.2022.128056)
38. Murg V, Verstraete F, Schneider R, Nagy PR, Legeza Ö. 2015 Tree tensor network state with variable tensor order: an efficient multireference method for strongly correlated systems. *J. Chem. Theory Comput.* **11**, 1027–1036. (doi:10.1021/ct501187j)
39. Vidal G. 2007 Entanglement renormalization. *Phys. Rev. Lett.* **99**, 220405. (doi:10.1103/PhysRevLett.99.220405)
40. Araz JY, Spannowsky M. 2022 Classical versus quantum: comparing tensor-network-based quantum circuits on large hadron collider data. *Phys. Rev. A* **106**, 062423. (doi:10.1103/PhysRevA.106.062423)
41. Tagliacozzo L, Evenbly G, Vidal G. 2009 Simulation of two-dimensional quantum systems using a tree tensor network that exploits the entropic area law. *Phys. Rev. B* **80**, 235127. (doi:10.1103/PhysRevB.80.235127)
42. Cincio L, Dziarmaga J, Rams MM. 2008 Multiscale entanglement renormalization ansatz in two dimensions: quantum ising model. *Phys. Rev. Lett.* **100**, 240603. (doi:10.1103/PhysRevLett.100.240603)
43. Wolf MM, Verstraete F, Hastings MB, Cirac JI. 2008 Area laws in quantum systems: mutual information and correlations. *Phys. Rev. Lett.* **100**, 070502. (doi:10.1103/PhysRevLett.100.070502)
44. Vidal G. 2003 Efficient classical simulation of slightly entangled quantum computations. *Phys. Rev. Lett.* **91**, 147902. (doi:10.1103/PhysRevLett.91.147902)
45. Convy I, Huggins W, Liao H, Whaley KB. 2022 Mutual information scaling for tensor network machine learning. *Mach. Learn.: Sci. Technol.* **3**, 015017. (doi:10.1088/2632-2153/ac44a9)
46. Evenbly G, Vidal G. 2014 Class of highly entangled many-body states that can be efficiently simulated. *Phys. Rev. Lett.* **112**, 240502. (doi:10.1103/PhysRevLett.112.240502)
47. Vidal G. 2004 Efficient simulation of one-dimensional quantum many-body systems. *Phys. Rev. Lett.* **93**, 040502. (doi:10.1103/PhysRevLett.93.040502)
48. Daley AJ, Kollath C, Schollwöck U, Vidal G. 2004 Time-dependent density-matrix renormalization-group using adaptive effective hilbert spaces. *J. Stat. Mech: Theory Exp.* **2004**, P04005. (doi:10.1088/1742-5468/2004/04/P04005)
49. Barratt F, Dborin J, Wright L. 2022 Improvements to gradient descent methods for quantum tensor network machine learning. *Second Workshop on Quantum Tensor Networks in Machine Learning*. (http://arxiv.org/abs/2203.03366)
50. Wall ML, Abernathy MR, Quiroz G. 2021 Generative machine learning with tensor networks: benchmarks on near-term quantum computers. *Phys. Rev. Res.* **3**, 023010. (doi:10.1103/PhysRevResearch.3.023010)
51. Spall JC. 1992 Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE Trans. Autom. Control* **37**, 332–341. (doi:10.1109/9.119632)
52. Grant E, Benedetti M, Cao S, Hallam A, Lockhart J, Stojevic V, Green AG, Severini S. 2018 Hierarchical quantum classifiers. *npj Quantum Inf.* **4**, 65. (doi:10.1038/s41534-018-0116-9)
53. McClean JR, Boixo S, Smelyanskiy VN, Babbush R, Neven H. 2018 Barren plateaus in quantum neural network training landscapes. *Nat. Commun.* **9**, 4812. (doi:10.1038/s41467-018-07090-4)
54. Hochreiter J. 1991 *Untersuchungen zu dynamischen neuronalen Netzen*, Diploma Thesis, München.
55. Swingle B. 2012 Entanglement renormalization and holography. *Phys. Rev. D* **86**, 231. (doi:10.1103/PhysRevD.86.065007)
56. Rohwedder T, Uschmajew A. 2013 On local convergence of alternating schemes for optimization of convex problems in the tensor train format. *SIAM J. Numer. Anal.* **51**, 1134–1162. (doi:10.1137/110857520)
57. Novikov A, Trofimov M, Oseledets I. 2018 Exponential machines. *Bull. Pol. Acad. Sci. Tech. Sci.* **66**, 789–797. (doi:10.24425/bpas.2018.125926)
58. Luchnikov IA, Krechetov ME, Filippov SN. 2021 Riemannian geometry and automatic differentiation for optimization problems of quantum physics and quantum technologies. *New J. Phys.* **23**, 073006. (doi:10.1088/1367-2630/ac0b02)
59. Hauru M, van Damme M, Haegeman J. 2021 Riemannian optimization of isometric tensor networks. *SciPost Phys.* **10**, 040. (doi:10.21468/SciPostPhys.10.2.040)

60. Stoudenmire EM, Schwab DJ. 2016 Supervised learning with quantum-inspired tensor networks. In *30th Conf. on Neural Information Processing Systems, Barcelona, Spain, 5–10 December*. Red Hook, NY: Curran Associates Inc. (doi:10.48550/arXiv.1605.05775)
61. Wall ML, D'Aguanno G. 2021 Tree tensor network classifiers for machine learning: from quantum-inspired to quantum-assisted. *Phys. Rev. A* **104**, 1498. (doi:10.1103/PhysRevA.104.042408)
62. Huggins W, Patil P, Mitchell B, Whaley KB, Stoudenmire EM. 2019 Towards quantum machine learning with tensor networks. *Quantum Sci. Technol.* **4**, 024001. (doi:10.1088/2058-9565/aaea94)
63. Glasser I, Pancotti N, Cirac JI. 2020 From probabilistic graphical models to generalized tensor networks for supervised learning. *IEEE Access* **8**, 68 169–68 182. (doi:68169–68182, doi:10.1109/ACCESS.2020.2986279)
64. Reyes JA, Stoudenmire EM. 2021 Multi-scale tensor network architecture for machine learning. *Mach. Learn.: Sci. Technol.* **2**, 035036. (doi:10.1088/2632-2153/abffe8)
65. Chen J, Cheng S, Xie H, Wang L, Xiang T. 2018 Equivalence of restricted Boltzmann machines and tensor network states. *Phys. Rev. B* **97**, 085104. (doi:10.1103/PhysRevB.97.085104)
66. Kong F, Liu X-y., Henao R. 2020 Quantum tensor network in machine learning: an application to tiny object classification. In *34th Conf. on Neural Information, Vancouver BC, Canada, 6–12 December*. Red Hook, NY: Curran Associates Inc.
67. Novikov A, Podoprikhin D, Osokin A, Vetrov DP. 2015 Tensorizing neural networks. In *Advances in neural information processing systems* (eds C Cortes, N Lawrence, D Lee, M Sugiyama, R Garnett), vol. 28. Red Hook, NY: Curran Associates, Inc.
68. Glasser I, Sweke R, Pancotti N, Eisert J, Cirac JI. 2019 Expressive power of tensor-network factorizations for probabilistic modeling, with applications from hidden Markov models to quantum machine learning. In: *Advances in Neural Information Processing Systems 32, Vancouver, Canada, 8–14 December*. Red Hook, NY: Curran Associates, Inc.
69. Li S, Pan F, Zhou P, Zhang P. 2021 Boltzmann machines as two-dimensional tensor networks. *Phys. Rev. B* **104**, 21. (doi:10.1103/PhysRevB.104.075154)
70. Collura M, Dell'Anna L, Felser T, Montangero S. 2021 On the descriptive power of neural-networks as constrained tensor networks with exponentially large bond dimension. *SciPost Phys. Core* **4**, 001. (doi:10.21468/SciPostPhysCore.4.1.001)
71. Wu D, Rossi R, Vicentini F, Carleo G. 2022 From tensor network quantum states to tensorial recurrent neural networks. (http://arxiv.org/abs/2206.12363)
72. Levine Y, Sharir O, Cohen N, Shashua A. 2019 Quantum entanglement in deep learning architectures. *Phys. Rev. Lett.* **122**, 401. (doi:10.1103/PhysRevLett.122.065301)
73. Araz JY, Spannowsky M. 2021 Quantum-inspired event reconstruction with tensor networks: matrix product states. *J. High Energy Phys.* **2021**, 1–28. (doi:10.1007/JHEP08(2021)112)
74. Felser T, Trenti M, Sestini L, Gianelle A, Zuliani D, Lucchesi D, Montangero S. 2021 Quantum-inspired machine learning on high-energy physics data. *npj Quantum Inf.* **7**, 111. (doi:10.1038/s41534-021-00443-w)
75. Selvan R, Ørting S, Dam EB. 2020 Multi-layered tensor networks for image classification. (http://arxiv.org/abs/2011.06982)
76. Liu Y, Li WJ, Zhang X, Lewenstein M, Su G, Ran SJ. 2021 Entanglement-based feature extraction by tensor network machine learning. *Front. Appl. Math. Stat.* **7**, 716044. (doi:10.3389/fams.2021.716044)
77. Wang J, Roberts C, Vidal G, Leichenauer S. 2020 Anomaly detection with tensor networks. (http://arxiv.org/abs/2006.02516)
78. Sun Z-Z, Peng C, Liu D, Ran S-J, Su G. 2020 Generative tensor network classification model for supervised machine learning. *Phys. Rev. B* **101**, 075135. (doi:10.1103/PhysRevB.101.075135)
79. Han Z-Y, Wang J, Fan H, Wang L, Zhang P. 2018 Unsupervised generative modeling using matrix product states. *Phys. Rev. X* **8**, 031012. (doi:10.1103/PhysRevX.8.031012)
80. Cheng S, Wang L, Xiang T, Zhang P. 2019 Tree tensor networks for generative modeling. *Phys. Rev. B* **99**, 155131. (doi:10.1103/PhysRevB.99.155131)
81. Bai S-C, Tang Y-C, Ran S-J. 2022 Unsupervised recognition of informative features via tensor network machine learning and quantum entanglement variations. *Chin. Phys. Lett.* **39**, 100701. (doi:10.1088/0256-307X/39/10/100701)

82. Liu D, Ran SJ, Wittek P, Peng C, García RB, Su G, Lewenstein M. 2019 Machine learning by unitary tensor network of hierarchical tree structure. *New J. Phys.* **21**, 073059. (doi:10.1088/1367-2630/ab31ef)
83. Wall ML, Titum P, Quiroz G, Foss-Feig M, Hazzard KRA. 2022 A tensor network discriminator architecture for classification of quantum data on quantum computers. *Phys. Rev. A* **105**, 520. (doi:10.1103/PhysRevA.105.062439)
84. Dborin J, Barratt F, Wimalaweera V, Wright L, Green AG. 2022 Matrix product state pre-training for quantum machine learning. *Quantum Sci. Technol.* **7**, 035014. (doi:10.1088/2058-9565/ac7073)
85. Ran S-J. 2020 Encoding of matrix product states into quantum circuits of one- and two-qubit gates. *Phys. Rev. A* **101**, 401. (doi:10.1103/PhysRevA.101.032310)
86. Lin S-H, Dilip R, Green AG, Smith A, Pollmann F. 2021 Real- and imaginary-time evolution with compressed quantum circuits. *PRX Quantum* **2**, 010342. (doi:10.1103/PRXQuantum.2.010342)
87. Schön C, Solano E, Verstraete F, Cirac JI, Wolf MM. 2005 Sequential generation of entangled multiqubit states. *Phys. Rev. Lett.* **95**, 110503. (doi:10.1103/PhysRevLett.95.110503)
88. Dilip R, Liu Y-J, Smith A, Pollmann F. 2022 Data compression for quantum machine learning. *Phys. Rev. Res.* **4**, 043007. (doi:10.1103/PhysRevResearch.4.043007)
89. Guala D, Cruz-Rico E, Zhang S, Arrazola JM. 2022 Tensor-network quantum circuits. Internet document, 27 June 2022. Xanadu, Toronto, Canada. (https://pennylane.ai/qml/demos/tutorial_tn_circuits.html)
90. Fastovets DV, Bogdanov Y, Bantysh BI, Lukichev VF. 2018 Machine learning methods in quantum computing theory. In *Int. Conf. on Micro- and Nano-Electron, Zvenigorod, Russian Federation, 1–5 October* (eds VF Lukichev, KV Rudenko), vol. 85. SPIE Digital Library.
91. Cong I, Choi S, Lukin MD. 2019 Quantum convolutional neural networks. *Nat. Phys.* **15**, 1273–1278. (doi:10.1038/s41567-019-0648-8)
92. Pesah A, Cerezo M, Wang S, Volkoff T, Sornborger AT, Coles PJ. 2021 Absence of barren plateaus in quantum convolutional neural networks. *Phys. Rev. X* **11**, 041011. (doi:10.1103/PhysRevX.11.041011)
93. Du Y, Hsieh M-H, Liu T, Tao D. 2020 Expressive power of parametrized quantum circuits. *Phys. Rev. Res.* **2**, 033125. (doi:10.1103/PhysRevResearch.2.033125)
94. Liao H, Convy I, Yang Z, Whaley KB. 2023 Decohering tensor network quantum machine learning models. *Quant. Mach. Intell.* **5**, 7. (doi:10.1007/s42484-022-00095-9)
95. Chen SY-C, Huang C-M, Hsing C-W, Goan H-S, Kao Y-J. 2022 Variational quantum reinforcement learning via evolutionary optimization. *Mach. Learn.: Sci. Technol.* **3**, 015025. (doi:10.1088/2632-2153/ac4559)
96. Sagingalieva A, Kurkin A, Melnikov A, Kuhmistrov D, Perelshtein M, Melnikov A, Skolik A, Von Dollen D. 2022 Hyperparameter optimization of hybrid quantum neural networks for car classification. (http://arxiv.org/abs/2205.04878)
97. Zhang K, Hsieh M-H, Liu L, Tao D. 2021 Toward trainability of deep quantum neural networks. (http://arxiv.org/abs2112.15002)
98. Qi J, Yang C-HH, Chen P-Y, Hsieh M-H. 2023 Theoretical error performance analysis for variational quantum circuit based functional regression. *npj Quantum Inf.* **9**, 4. (doi:10.1038/s41534-022-00672-7)
99. Liu Z, Yu L-W, Duan LM, Deng D-L. 2022 The presence and absence of barren plateaus in tensor-network based machine learning. *Phys. Rev. Lett.* **129**, 177. (doi:10.1103/PhysRevLett.129.270501)
100. Zhao C, Gao X-S. 2021 Analyzing the barren plateau phenomenon in training quantum neural networks with the zx-calculus. *Quantum* **5**, 466. (doi:10.22331/q-2021-06-04-466)
101. Cervero Martín E, Plekhanov K, Lubasch M. 2023 Barren plateaus in quantum tensor network optimization. *Quantum* **7**, 974. (doi:10.22331/q-2023-04-13-974)
102. Liu H-Y, Sun T-P, Wu Y-C, Han Y-J, Guo G-P. 2023 Mitigating barren plateaus with transfer-learning-inspired parameter initializations. *New J. Phys.* **25**, 013039. (doi:10.1088/1367-2630/acb58e)
103. Wang K, Xiao L, Yi W, Ran S-J, Xue P. 2021 Experimental realization of a quantum image classifier via tensor-network-based machine learning. *Photonics Res.* **9**, 2332. (doi:10.1364/PRJ.434217)
104. Bhatia AS, Saggi MK, Kumar A, Jain S. 2019 Matrix product state–based quantum classifier. *Neural Comput.* **31**, 1499–1517. (doi:10.1162/neco_a_01202)

105. Kardashin A, Uvarov A, Biamonte J. 2021 Quantum machine learning tensor network states. *Front. Phys.* **8**, 586374. (doi:10.3389/fphy.2020.586374)
106. Treinish M *et al.* 2023 Qiskit/qiskit-metapackage: Qiskit 0.43.1, 10.5281/ZENODO.8003781.
107. Cirq Developers. Cirq, (doi:10.5281/ZENODO.7465577). 2022.
108. Bergholm V *et al.* 2018 Pennylane: automatic differentiation of hybrid quantum-classical computations. (http://arxiv.org/abs/1811.04968)
109. Schwarz M, Temme K, Verstraete F. 2012 Preparing projected entangled pair states on a quantum computer. *Phys. Rev. Lett.* **108**, 110502. (doi:10.1103/PhysRevLett.108.110502)
110. Lu S, Kanász-Nagy M, Kukuljan I, Cirac JI. 2021 Tensor networks and efficient descriptions of classical data. (http://arxiv.org/abs/2103.06872)
111. McCord JC, Evenbly G. 2022 Improved wavelets for image compression from unitary circuits. (http://arxiv.org/abs/2203.02556)
112. Le PQ, Dong F, Hirota K. 2011 A flexible representation of quantum images for polynomial preparation, image compression, and processing operations. *Quantum Inf. Process.* **10**, 63–84. (doi:10.1007/s11128-010-0177-y)
113. Latorre JI. 2005 Image compression and entanglement. (http://arxiv.org/abs/quant-ph/0510031)
114. Chen SY-C, Huang C-M, Hsing C-W, Kao Y-J. 2021 An end-to-end trainable hybrid classical-quantum classifier. *Mach. Learn.: Sci. Technol.* **2**, 045021. (doi:10.1088/2632-2153/ac104d)