Multi-scale tensor network architecture for machine learning

IOP Publishing Mach. Learn.: Sci. Technol. 2 (2021) 035036 https://doi.org/10.1088/2632-2153/abffe8

MACHINE LEARNING
Science and Technology

OPEN ACCESS

RECEIVED
21 January 2021

REVISED
25 April 2021

ACCEPTED FOR PUBLICATION
11 May 2021

PUBLISHED
14 July 2021

PAPER

Multi-scale tensor network architecture for machine learning

J A Reyes[1,*] and E M Stoudenmire[2]

[1] Department of Physics, University of Central Florida, University Blvd, Orlando, FL 32816, United States of America
[2] Center for Computational Quantum Physics, Flatiron Institute, 5th Avenue, New York, NY 10010, United States of America
* Author to whom any correspondence should be addressed.

E-mail: jreyesucf@knights.ucf.edu

Keywords: tensor network, machine learning, dimensionality reduction

Abstract

We present an algorithm for supervised learning using tensor networks, employing a step of data pre-processing by coarse-graining through a sequence of wavelet transformations. These transformations are represented as a set of tensor network layers identical to those in a multi-scale entanglement renormalization ansatz tensor network. We perform supervised learning and regression tasks through a model based on a matrix product states (MPSs) acting on the coarse-grained data. Because the entire model consists of tensor contractions (apart from the initial non-linear feature map), we can adaptively fine-grain the optimized MPS model 'backwards' through the layers with essentially no loss in performance. The MPS itself is trained using an adaptive algorithm based on the density matrix renormalization group algorithm. We test our methods by performing a classification task on audio data and a regression task on temperature time-series data, studying the dependence of training accuracy on the number of coarse-graining layers and showing how fine-graining through the network may be used to initialize models which access finer-scale features.

1. Introduction

Computational techniques developed across the machine learning and physics disciplines have consistently generated promising methods and applications in both areas of study. The application of well established machine learning architectures and optimization techniques has enriched the physics community with advances in modeling such as recognizing topological quantum states [1–3], optimizing quantum error correction codes [4], classifying quantum walks [5], and solving partial differential equations [6]. Conversely, tensor networks, which are well established numerical physics techniques with close connection to physics principles and used for modeling high-dimensional functions, have begun to be explored more in applied mathematics and machine learning [7–19]. The benefits of the tensor network approach in machine learning include the compression of model parameters (i.e. dimensionality reduction) [7, 14, 15], the implementation of adaptive training algorithms [9], and the possibility of gaining theoretical insights [17, 20].

Generically, a tensor network is a factorization of a very high-order tensor—a tensor with a large number of indices—into a set of low-order tensors whose indices are summed (contracted) to form a network defined by a certain pattern of contractions. Such a network can be viewed as a generalization of the low-rank matrix decompositions familiar to the discipline of applied mathematics.

One of the primary advantages of a tensor network decomposition is that the number of parameters needed to model the high-order tensor can be exponentially reduced while still approximating the high-order tensor accurately within many applications of interest [21]. This is particularly evident in the simulation of quantum many-body systems where tensor networks serve as a foundational tool for describing the system state vector—which encodes the joint probability of many variables. In this case, the state vector often cannot be explicitly stored or manipulated because of its exponentially growing dimension with the number of variables. In similarity to this, certain machine learning approaches, such as kernel learning, also encounter the problem of high dimensionality when the model involves a large number of data features [22]. In this

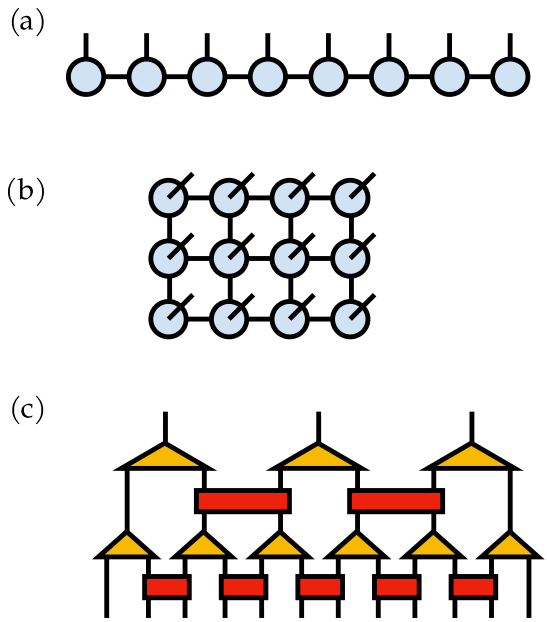

Figure 1. Tensor diagrams for the (a) matrix product state (MPS) tensor network, (b) PEPS tensor network, and (c) two-layer MERA tensor network.

paper, we will focus on utilizing the concepts of tensor network renormalization as a means of accomplishing a dimensionality reduction on the large parameter spaces found in various classification tasks.

Common tensor network families include the matrix product states (MPSs) or tensor trains [23–26], projected entangled pair states [21], and the multi-scale renormalization ansatz (MERA) [27, 28]. Example tensor diagrams defining each of the above mentioned architectures are shown in figure 1. Because tensors are multi-linear objects, controlled transformations of one tensor network into another are possible through techniques such as matrix factorization and tensor contraction [18, 29, 30]. We will take advantage of this capability by utilizing these transformations as an initialization strategy for our model. This will be accomplished by initially training a model defined through a larger number of MERA coarse-graining layers, then fine-graining the top tensor train containing the adjustable model weights to reach the desired architecture.

The specific algorithm we present for training a model for supervised learning and regression applications is built around three main steps. The first step involves coarse graining to reduce the feature space representing the data. This is done through a series of wavelet transformations which are approximated by the layers of a MERA tensor network. The second step is the training of a weight tensor represented as an MPS and optimized by an alternating least squares (ALSs) algorithm analogous to the density matrix renormalization group (DMRG) algorithm. In the third step, we show how the weight tensor can be systematically fine-grained (i.e. back-propagated through the MERA) into a model over a larger set of features, having initially the same performance, but then being further optimized at the fine scale.

This paper is organized as follows: In section 2, we provide a high-level overview of (1) the wavelet transformations which are to be encoded into the layers of a MERA, (2) the DMRG algorithm which provides the foundation for the dimensionally-reduced training protocol, and (3) the MERA tensor network, which will be primarily used as a tool for dimensionality reduction. In section 3, we detail the methods involved in encoding the MERA with wavelet layers, and describe our algorithm for training the weights of the classifier. In section 4, we apply our method to the classification of DCASE audio file data sets and the linear regression of mean temperature data sets. We conclude in section 5 with a discussion on the applicability and outlook of our algorithm.

2. Theoretical background

The steps of the algorithm we will explore combine a number of techniques from different fields of study. In this section we present a brief review of these techniques to aid readers in understanding the motivation behind combining them together. It is noted that previous works in the physics literature have been similarly motivated to adapt multi-scale or multi-grid modeling ideas into the tensor network setting [29, 30].

Additionally, the application of other hierarchical tensor-network architectures have been previously investigated in various discriminative and generative tasks [14, 15].

In the following section, we present background for the machine learning tasks we have investigated; we review the theory of wavelet transformations; we review the MPS tensor network and DMRG algorithm; and finally, we review the MERA tensor network.

2.1. Learning task and model functions

We will be interested in the tasks of regression and supervised learning. Given a training data set of observed input and output pairs $\{\mathbf{x}_j, \mathbf{y}_j\}_{j=1}^n$, both of these tasks can be formulated as finding a function $f(\mathbf{x})$ satisfying $f(\mathbf{x}_j) \approx \mathbf{y}_j$ to a good approximation. Crucially, out of all functions which fit the input data, $f(\mathbf{x})$ should be selected so as to generalize to unobserved data. Outputs $\mathbf{y}_j$ can be vector-valued, but in the case of regression are more commonly just scalar-valued.

The class of model functions we will optimize have the general form

$$f_W(\mathbf{x}) = W \cdot \Phi(\mathbf{x}) \tag{1}$$

where W is a weight vector which enters linearly. The function Φ is a feature map, which, in general, is non-linear and maps the input $\mathbf{x}$ to a feature vector $\Phi(\mathbf{x})$. This class of model functions formally has the same starting point as the kernel learning and support vector machine approaches to machine learning. However, in contrast to these approaches, we will optimize over the weights W without introducing any dual parameters as in the 'kernel trick'. To make such an optimization feasible, we represent the weights in a compressed form as a tensor network.

The feature maps we consider take the form

$$\Phi^{s_1\ s_2\ \cdots s_N}(\mathbf{x}) = \phi^{s_1}(x_1)\phi^{s_2}(x_2)\cdots\phi^{s_N}(x_N) \tag{2}$$

where N is the dimension of the input vectors $\mathbf{x}$, and each index s_i runs over d values. The *local feature maps* ϕ map each input component x_i to a d-dimensional vector. The resulting feature map Φ is a map from $\mathbb{R}^N$ to a rank-1 tensor of order N, whose indices are all of dimension d. Informally, Φ is a product of vector-valued functions of each of the input components.

Our goal is to find a suitable weight vector W^* that minimizes some appropriate cost function between the model output $f_{W^*}(\mathbf{x}_j)$ and the expected output $\mathbf{y}_j$ over the observed data. The cost function we will use is the quadratic cost

$$C(W) = \frac{1}{2n}\sum_{j=1}^{n} ||f_W(\mathbf{x}_j) - \mathbf{y}_j||^2 + \lambda||W||^2 \tag{3}$$

where the summation is over n training examples and λ is an empirical regularization parameter. Because of the quadratic form of this function, optimization can be conveniently carried out by efficient methods such as the conjugate gradient descent algorithm.

2.2. Wavelet transformations

We will be interested in dimensionality reduction transformations which preserve information across scales. To this end, we consider transformations common to signal processing techniques. In signal processing, the Fourier transform is a ubiquitous transformation which can be used to take an input signal from the time domain to the frequency domain. However, there are many instances where a signal is not stationary, requiring analysis to be done in both the time and the frequency domain. Wavelet transformations are transformation which facilitate this type of analysis by transforming a signal into the time-frequency domain [31].

The continuous wavelet transformation of an input signal $x(t)$ is given by

$$X(\tau, s) = \frac{1}{\sqrt{|s|}}\int x(t)\bar{\psi}\left(\frac{t-\tau}{s}\right)dt \tag{4}$$

where ψ is the *mother wavelet* which characterizes the transformation, τ is the translation parameter, and s is the scale parameter.

For discrete signals taken over constant time intervals, the mother wavelet becomes a vector whose inner product is taken with a subset of the signal x. The integral becomes a summation over the elements within that subset (neighboring subsets can generally overlap).

The two types of discrete wavelet transformations we will use below are the Haar transformation

$$h_i^{(2)} = \frac{x_{2i} + x_{2i+1}}{\sqrt{2}} \tag{5}$$

for $i = 0$ to $i = N/2 - 1$; and the Daubechies-4 (Daub4) transformation

$$d_i^{(4)} = \sum_{j=0}^{3} x_{2i+j} D_j \tag{6}$$

where the coefficients D_j are the elements of the vector

$$D = \left(\frac{1+\sqrt{3}}{4\sqrt{2}}, \frac{3+\sqrt{3}}{4\sqrt{2}}, \frac{3-\sqrt{3}}{4\sqrt{2}}, \frac{1-\sqrt{3}}{4\sqrt{2}} \right). \tag{7}$$

For both of these discrete transformations the translation parameter is $\tau = 2$. The summation is over two elements for the Haar transformation and four elements for the Daub4 transformation. These discrete wavelet transformations can be used to average and rescale the initial signal x from size N to size $N/2$, while preserving local information in both the time and frequency domains. The information that is preserved is that associated with the part of the signal which varies more smoothly with the time index i.

2.3. MPS tensor network and optimization

A MPS or tensor train is a decomposition of a high order tensor into a contracted network order-3 tensors with a one-dimensional, chain-like structure as in figure 1(a). In traditional tensor notation an MPS decomposition of a tensor $W^{s_1\ s_2\ \cdots s_N}$ can be written as

$$W^{s_1\ s_2\ s_3\ \cdots s_N} = \sum_{\{a\}} A^{s_1}_{a_1} A^{s_2}_{a_1 a_2} A^{s_3}_{a_2 a_3} \cdots A^{s_N}_{a_{N-1}} \tag{8}$$

where the factor tensors $A^{s_j}_{a_{j-1} a_j}$ can in general be different from each other.

The key parameter controlling the expressivity of an MPS is the *bond dimension*, also known as tensor-train rank. This is the dimension of the internal, contracted indices $\{a_j\}$ connecting neighboring factor tensors of the MPS. By allowing the bond dimension large enough, an MPS can represent any tensor [23, 24]. In general, the dimensions of the bond indices vary, in which case the bond dimension of the MPS as a whole means the maximum over the bond dimensions.

There are many important theoretical results about MPS, such as finding minimal sets of conditions sufficient to make the parameters of the factor tensors uniquely specified [24–26]. For our purposes, the most important fact about MPS will be that they are well-suited for modeling functions or processes with strongly one-dimensional correlations, such as samples of a time-dependent random variable with decaying correlations. MPS can exactly reproduce long-time correlations which decay exponentially, and can also approximate power-law correlations to high accuracy [32].

A straightforward but powerful way to optimize a MPS for a given objective is to optimize each of the factor tensors $A^{s_j}_{a_{j-1} a_j}$ one-by-one, keeping the others fixed. In physics, the adaptive version of this approach is generally referred to as the DMRG algorithm [33], while in mathematics it is known as ALSs. By sweeping from the first tensor to the last and then back when optimizing, computations involving the other factor tensors can be reused, making the approach highly efficient. Another advantage of alternating optimization is that it can be made adaptive by temporarily contracting over the bond index shared between two factor tensors. After optimizing the resulting tensor, the MPS form can be restored by an singular value factorization, introducing a new bond index that can be selected larger or smaller depending on the desired tradeoff between the quality of the results and the computational cost [9].

2.4. MERA

The MERA is a tensor network whose geometry and structure implements multiple coarse grainings of input variables. In physics terminology, such a coarse-graining process is known as *renormalization* [34]. While it is true that a coarse-graining process could be implemented by a tree tensor network, a tree has the drawback that information or features processed by different subtrees are not merged until the subtrees meet, which can happen at an arbitrarily high scale. The MERA architecture accounts for this by including extra 'disentangler' tensors which span across subtrees [28, 35]—these are the four-index tensors U shown in figure 2.

Introducing tensors which connect subtrees has the potential to make computations with the resulting network prohibitively expensive. However, in a MERA the disentangler tensors U are constrained to always

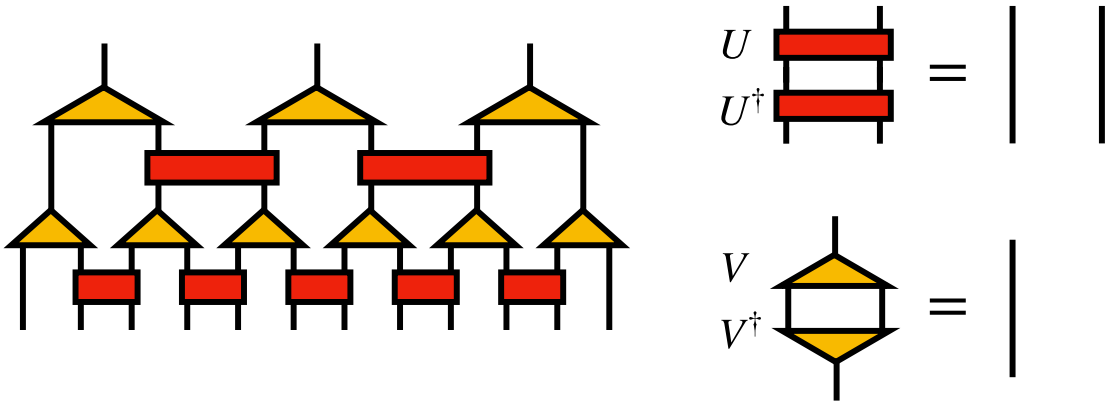

Figure 2. Two layers of a MERA tensor network, showing the unitary condition obeyed by the disentanglers U and isometric condition obeyed by the isometries V.

be unitary with $U^\dagger U = UU^\dagger = 1$. Likewise, the tree tensors, or isometry tensors V are constrained to obey an isometric condition $V^\dagger V = 1$ (yet $VV^\dagger \neq 1$). These conditions are depicted in diagrammatic form in figure 2. Under these constraints, computations of the norm of a tensor represented by a MERA (or of marginals and correlation functions when the MERA represents a distribution) can be carried out with a cost scaling polynomially in the dimensions of the internal indices of the network.

In this work, we will specifically consider disentanglers and isometries parameterized as matrices with non-zero elements given as:

$$U = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta_U & \sin\theta_U & 0 \\ 0 & -\sin\theta_U & \cos\theta_U & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$V = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \sin\theta_V & \cos\theta_V & 0 \end{pmatrix}$$

which will be sufficient to parameterize MERA layers which approximately compute wavelet coarse graining transformations of input data. We discuss how to choose the angles θ_U and θ_V to approximate Haar and Daubechies wavelets in the next section. The existence of a correspondence between the MERA tensor network and discrete wavelet transformations was first described by [36, 37].

3. Model and training algorithm

The model function we will now discuss for regression and supervised learning first coarse grains input data through some number of discrete wavelet transformations, implemented as MERA tensor network layers. Following this, an MPS tensor network is used to represent the top layer of trainable weights.

After discussing how to train a model of this type, we highlight one of its key advantages: the amount of coarse graining can be adjusted *during training* to adaptively discover the number of coarse graining steps needed to obtain satisfactory results.

3.1. Coarse graining

To reduce the parameter space necessary for training the weights of our classifier, we coarse grain the input data through a series of wavelet transformations, effectively reducing the size of the data by a factor of two after each transformation. This is done by first mapping each input data element x_i to the vector $|\phi(x_i)\rangle = |0\rangle + x_i|1\rangle$, where in this section we use the physics notation that $|v\rangle$ is a vector labeled v. We have also defined

$$|0\rangle = (1\ 0)^T \tag{9}$$

$$|1\rangle = (0\ 1)^T. \tag{10}$$

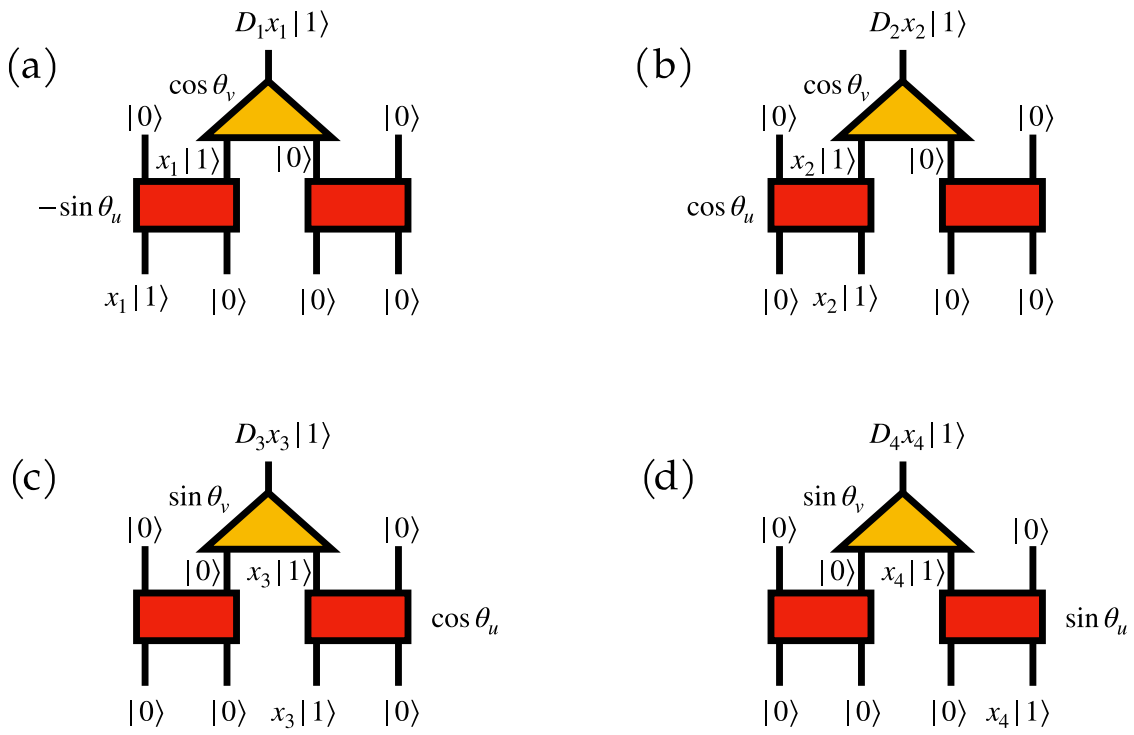

Figure 3. Decomposition of each transformation element D_{1-4} into the unitaries and isometries of the wavelet MERA. The complete Daub4 wavelet transformation over data elements x_{1-4} is given by the contraction over connected indices. Each transformation is parameterized by θ_U and θ_V.

The feature map applied to each data sample is taken to be the tensor product

$$|\Phi(\mathbf{x})\rangle = |\phi(x_1)\rangle \otimes |\phi(x_2)\rangle \otimes \ldots \otimes |\phi(x_N)\rangle \,. \tag{11}$$

This input tensor can be thought of as an MPS of bond dimension 1. As shown in figure 3, this MPS becomes the bottom layer of a network with the wavelet transformation MERA representing the upper layers. Each subsequent layer in the MERA is constructed by encoding wavelet transformations into the disentangler and isometry tensors U and V.

To accomplish the encoding of these wavelet transformations, we first decompose each of the wavelet coefficients in the set $\{D_i\}$ given in equation (7) into two sequentially applied transformations. Graphically, if we consider the term $x_i|1\rangle$ in each local feature vector $|\phi(x_i)\rangle$ of equation (11) as a 'particle' whose state has a coefficient given by the input component x_i, we can trace the path of this particle through the MERA as shown in figure 3, assigning appropriate transformations to x_i as it propagates through the tensors. Following this construction, one can work out the result of applying the MERA layer to a patch of four adjacent input tensors, whose dependence on input components (x_1, x_2, x_3, x_4) is to leading order:

$$\begin{aligned} |\phi(x_1)\rangle|\phi(x_2)\rangle|\phi(x_3)\rangle|\phi(x_4)\rangle &= (|0\rangle + x_1\,|1\rangle)\,(|0\rangle + x_2\,|1\rangle)\,\cdots \\ &= |0\rangle|0\rangle|0\rangle|0\rangle|0\rangle + x_1\,|1\rangle|0\rangle|0\rangle|0\rangle + x_2\,|0\rangle|1\rangle|0\rangle|0\rangle + x_3\,|0\rangle|0\rangle|1\rangle|0\rangle \\ &\quad + x_4\,|0\rangle|0\rangle|0\rangle|1\rangle + \cdots \end{aligned} \tag{12}$$

where the omitted terms are higher-order in components of $\mathbf{x}$, such as $x_2x_4\,|0\rangle|1\rangle|0\rangle|1\rangle$.

From the conditions shown in figure 3, it follows that the result of acting with the MERA layer on this patch of inputs is, to linear order in the components of $\mathbf{x}$, an output vector

$$(D_1x_1 + D_2x_2 + D_3x_3 + D_4x_4)|1\rangle \tag{13}$$

where now $|1\rangle$ labels the second basis vector of the vector space defined by the tensor indices along the top of the MERA layer, and the Daub4 wavelet coefficients D_i are related to the parameters in the MERA factor tensors as:

$$D_1 = -\sin\theta_U \cos\theta_V \tag{14}$$

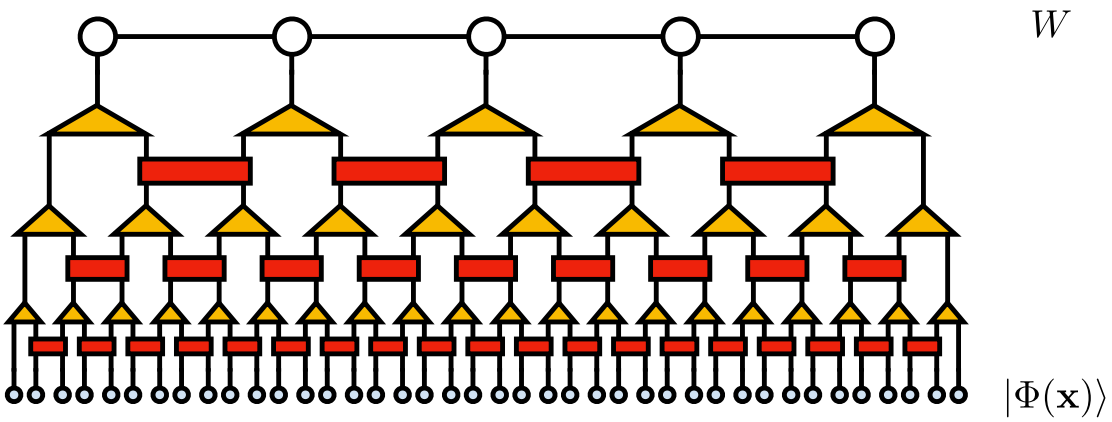

Figure 4. The tensor diagram for the model we use for classification and regression, showing the case of three MERA layers. Each input data $\mathbf{x}$ is first mapped into a (rank-1) MPS $|\Phi(\mathbf{x})\rangle$, then acted on by MERA layers approximating Daub4 wavelet transformations. At the top layer, the trainable parameters of the model W are decomposed as an MPS. Because all tensor indices are contracted, the output of the model is a scalar.

$$D_2 = \cos\theta_U \cos\theta_V \tag{15}$$

$$D_3 = \cos\theta_U \sin\theta_V \tag{16}$$

$$D_4 = \sin\theta_U \sin\theta_V. \tag{17}$$

With the D_i chosen to be the Daub4 wavelet coefficients equation (7), this system of equations is easily solved by setting $\theta_U = \pi/6$ and $\theta_V = \pi/12$. With this choice of angles, the wavelets have successfully been encoded into the unitaries and isometries of our MERA and can successively be applied to the initial dataset to reduce the training parameter space by a factor of two for each layer.

While the above construction guarantees that the action of a MERA layer on the input feature vector $\Phi(\mathbf{x})$ reproduces the wavelet transformation for terms involving up to a single $|1\rangle$ basis vector, such an exact correspondence no longer holds for terms with two or more $|1\rangle$ basis vectors in the tensor product. It is easy to show that when the $|1\rangle$ vectors are far enough apart, the wavelet correspondence holds; when the $|1\rangle$ vectors are closer together, there are corrections. However, we simply choose to accept these as a defining property of our coarse-graining process, since choosing one wavelet family over another is already somewhat arbitrary. It will be interesting in future work to explore how to make the mapping between wavelets and MERA layers more precise in the context of coarse-graining data.

Finally, we turn to how we coarse-grain each training sample through the MERA layers efficiently. Recall that each sample is first converted to a rank-1 tensor (or product state in physics terminology) of the form equation (11), which we choose to view as an MPS tensor network of bond dimension 1. Applying the first MERA layer to this MPS in a naive way would destroy the MPS form, making any steps afterward very inefficient. However, we can proceed using a very accurate, controlled approximation which is to apply the tensors in each MERA layer one by one, factoring the resulting local tensors using a truncated singular value decomposition (SVD). This process is closely analogous to time evolution methods for MPS such as time-evolving block decimation which are well developed in physics [38, 39].

3.2. Training

After each training data sample $|\Phi(\mathbf{x}_i)\rangle$ has been coarse grained through the MERA from size N to $N' = N/2^L$ where L is the number of wavelet-MERA layers used, we compute the (scalar) output of the model by an inner product with the tensor of weights W at the topmost scale. We choose this weight tensor to be represented by an MPS:

$$W^{s_1\ s_2\ s_3\ \cdots s_N} = \sum_{\{a\}} A^{s_1}_{a_1} A^{s_2}_{a_1 a_2} A^{s_3}_{a_2 a_3} \cdots A^{s_N}_{a_{N-1}}. \tag{18}$$

We then optimize the cost function in equation (3) by sweeping back and forth through the tensors of the above MPS, updating each tensor in a DMRG-like fashion [9].

The prominent feature of the optimization is a local update to W by optimizing the MPS tensors at sites j and $j+1$ together. This is accomplished by constructing

$$B^{s_j s_{j+1}}_{\alpha_{j-1} l \alpha_{j+1}} = A^{s_j}_{\alpha_{j-1}\alpha_j} A^{s_{j+1}}_{\alpha_j \alpha_{j+1}}. \tag{19}$$

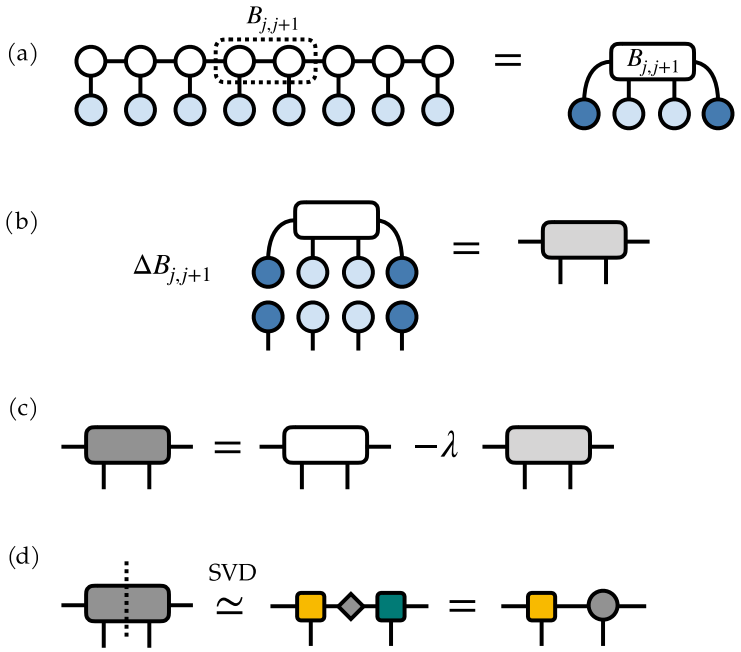

Figure 5. The local update of $W^{(MPS)}$ is as follows: (a) select a pair of neighboring $W^{(MPS)}$ tensors as B_{jj+1}, (b) compute the gradient of the cost over all training examples, (c) update B_{jj+1}, and (d) decompose back into two MPS tensors using a truncated SVD. This is done for each pair in the MPS, sweeping back and forth until the cost is minimized.

Figure 5(a) shows how the output of the model can be computed by first contracting all of the coarse-grained data tensors with the MPS tensors not currently being optimized, then with the bond tensor B. The gradient of the cost function is proportional to the tensor ΔB which is shown in figure 5(b). The ΔB tensor can be used to improve B as shown in figure 5(c). In practice, we actually use the conjugate gradient algorithm to perform the optimization, but the first step of this algorithm is identical to figures 5(b) and (c) for the proper choice of λ. Finally, to proceed to the next step the MPS form of W must be restored to ensure efficiency. This can be done by treating B as a matrix and computing its SVD as shown in figure 5(d). Truncating the smallest singular values of the SVD gives an adaptive way to automatically adjust the bond dimension of the weight MPS during training.

3.3. Fine scale projection

As a unique feature of our model, we introduce an additional step which provides the possibility for further optimization of the weight parameters for our classifier. Once training over the weight MPS has been completed at the topmost scale, the optimized weights can be projected back through the MERA consisting of N_{d4} layers to the previous scale defined by $N_{d4}-1$ layers.

This fine-graining step is done by first applying the conjugate of the isometries of the topmost MERA layer to each MPS tensor representing W, thereby doubling the number of sites. Next, we apply the conjugate of the unitary disentangler transformations to return to the basis defining the previous scale. Finally, to restore the MPS form of the transformed weights, we use an SVD factorization to split each tensor so that the factors each carry one site or 'feature' index. All of these steps are shown in figure 6(b).

The projection of these trained weights onto to the new finer scale serves as an initialization W' for layer $N_{d4}-1$. At this step, the coarse-grained data previously stored at this finer scale are retrieved (having been previously saved in memory), and a new round of training is performed for W'. The intent is for this projected MPS W' to provide a better initialization for the optimization than could have otherwise been obtained by directly transforming from finer scales. Table 1 enumerates each step in our algorithm from initial coarse-graining to training and fine scale projection.

4. Results

4.1. DCASE

The DCASE audio classification set consists of 15 batches (one batch per label), each containing 234 ten second audio clips for training [40]). Each audio clip is a vector of 441 000 samples, which we embedded into a vector of 2^{19} elements by padding with zeros. We constrained the problem to focus on binary classification,

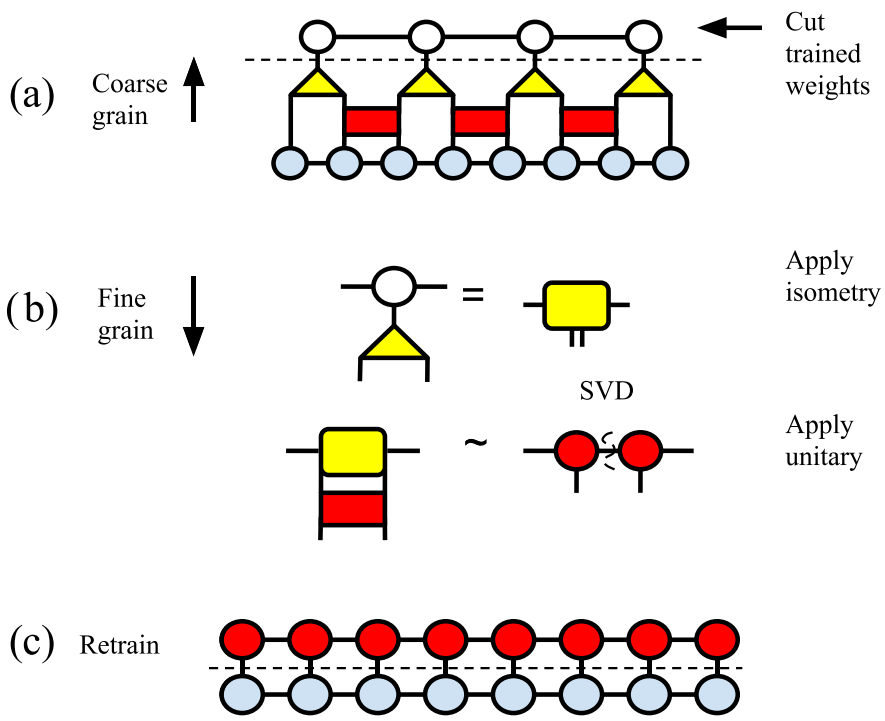

Figure 6. (a) The training sets are embedded into an MPS (light blue) and coarse grained through the wavelet MERA for training the weight MPS (white) at the top most layer. (b) Once training is complete, the weight MPS is fine grained through the wavelet MERA by applying isometries and unitaries, and (c) training is carried out on the training set at the finer scale.

Table 1. Algorithm.

Step 1	Map each training sample $\mathbf{x}_i$ into $\|\Phi(\mathbf{x}_i)\rangle$, equation (11)
Step 2	Coarse grain each $\|\Phi(\mathbf{x}_i)\rangle$ through the wavelet MERA
Step 3	Train W at the current layer
Step 4	Project the trained W to the finer scale, figure 6
Step 5	Repeat steps 3 and 4 as desired

specifically distinguishing between 'bus' and 'beach' environment audio clips. The data that support the findings of this study are available upon reasonable request from the authors.

Each data set for the selected labels was coarse grained through N_{h2} Haar transformation before encoding the data into our Daub4 wavelet MERA. Considering the full set of coarse-graining transformations $N_{\text{tot}} = N_{h2} + N_{d4}$, the full parameter space is reduced as $2^{19}/2^{N_{\text{tot}}}$.

We trained the classifier at the top layer of the MERA for a varying number of $N_{d4}+1$ Daub4 transformations, and subsequently projected the weights to the previous finer scale (scale N_{d4}). The percentage of correctly labeled examples after training at each scale is shown in figure 7. Each two point segment corresponds to training at the coarse scale (the right-most point in a segment), followed by training at the finer scale (the left-most point in a segment). We note that the accuracy of the model decreases with the number of wavelet transformations. But it is also apparent that training the weights after $N_{d4}+1$ layers produces a better initialization and optimization for training at the N_{d4} scale. This is particularly noted for the $N_{h2} = 12$ as shown in figure 7. The accuracy over the testing set in this case is measured well above the DCASE baseline score of 61%, when looking at the scores for the back-projected $N_{d4} = 1$ and 2 layers. Additionally, as can be noted by the closeness in accuracy between the training and testing sets for $N_{h2} = 14$, versus for $N_{h2} = 12$, the generalization of the model improves with the number of wavelet layers applied. Optimization was carried over five sweeps, with the bond dimension of the weight MPS adaptively selected by keeping singular values above the threshold $\Delta = 1 \times 10^{-14}$.

In addition to the above experiments distinguishing clips labeled 'beach' from clips labeled 'bus', we also trained our model for the one-versus-all classification task where we distinguish clips labeled 'beach' from all other clips in the dataset. Here the purpose is to directly compare our architecture and training process to the leading neural-network models using in the 2017 DCASE acoustic scene classification challenge [41–43]. The results are shown in table 2. While our model's test accuracy is not the highest, it is competitive with the

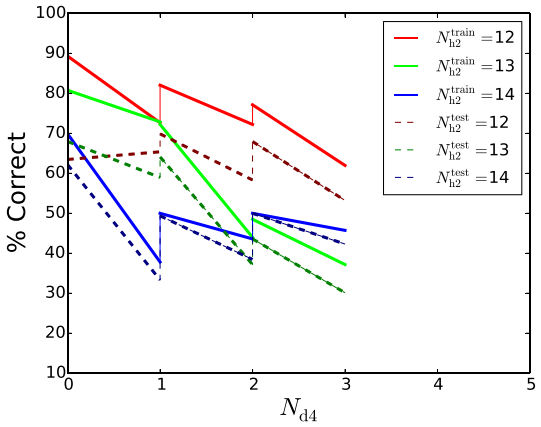

Figure 7. The percentage of correctly labeled samples (training set as solid lines and test set as dashed lines) for the audio experiment after training W over input data coarse grained through $N_{d4}+1$ Daub4 wavelet transformations and then trained again at N_{d4} layers. Each data set was initially embedded into a vector of 2^{19} elements and coarse grained N_{h2} Haar wavelet transformations before being embedded in the Daub4 wavelet MERA. The optimization was carried over five sweeps, keeping singular values above $\Delta = 1 \times 10^{-14}$.

Table 2. The testing accuracy for various machine learning models applied to the binary classification of the 'beach' audio class for the 2017 DCASE acoustic dataset. In this first column are the results for the wavelet-MERA. In the second, the results for a model utilizing a generative adversarial network (GAN) as input for a convolutional neural network (CNN) are given. This was the winning model of the DCASE challenge. Additionally in rows three and four results for a deep CNN (DCNN) with subsequent support vector machine (SVM) and for a convolutional recurrent neural network (CRNN) are given. These neural network models were all ranked amongst the top twenty in the DCASE acoustic scene classification challenge when applied to the classification of all 15 acoustic scenes.

Model	Wavelet-MERA	GAN-CNN	DCNN-SVM	CRNN
Testing accuracy (%)	69.8	83.3	71.3	43.5

other results. We expect these results could be improved in the future with more experience in applying our architecture to this type of data and additional techniques such as making the parameters in the wavelet-derived MERA layers trainable.

4.2. Regression

A single data file of the average daily temperatures of the Fisher River recorded from 1 January 1988 to 31 December 1991 were used to construct input data sets for regression in the following manner. By labeling each temperature x_i for $0 < i < 1462$, the fitting interval N_{fit} was taken as all the temperatures $\{x_i | i \in [731, 1461]\}$. A single training example was constructed by selecting a contiguous block of p temperatures from N_{fit} as input for the MERA. The temperature immediately following this p block was assigned as the label for that training example. By shifting the starting index of the p block of temperatures, multiple examples could be constructed. In this way, the regression task was recast as a classification task over a continuous label. Because nearly every example contained a unique label, we used the average absolute difference of the interpolated label and the actual label, $\langle \epsilon \rangle$, as a measure of the accuracy of the model. The training phase of this data was carried out for forty sweeps, and with a singular value threshold set to $\Delta = 1 \times 10^{-9}$. In figure 8, similar to the audio classification task, $\langle \epsilon \rangle$ is given for input data coarse-grained through $N_{d4}+1$, with the weights trained at this scale, and then projected onto the finer scale N_{d4} for further training. Each two-segment section is representative of training at $N_{d4}+1$ layers on the right-most point and training at N_{d4} layers on the left-most point. Again, we note a general decrease in accuracy with the number of wavelet transformations, but an improvement in accuracy when the weights are initialized by optimization through $N_{d4}+1$ wavelet layers, as compared to directly randomly initializing at N_{d4} layers.

5. Discussion

MERA are a family of tensor network factorizations which process information in a hierarchical way and preserve computational advantages associated with tree-tensor networks, such as efficient marginalization, yet can be more expressive and powerful because of extra 'disentangler' layers which mix branches of the tree. Here we have proposed an architecture for machine learning which uses MERA layers which approximate wavelet scaling functions as a preprocessing step, and explored the advantages of this choice. Because of the

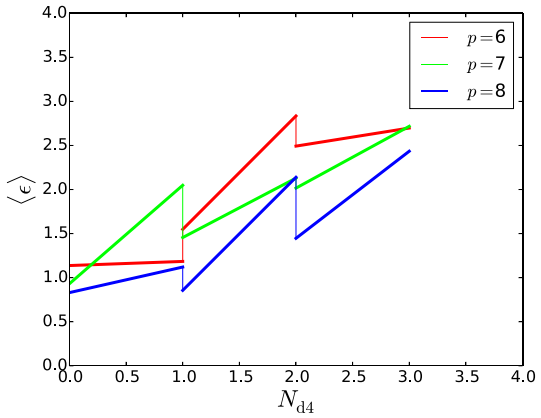

Figure 8. The average absolute deviation $\langle\epsilon\rangle$ (lower is better), for the temperature experiment after training $W^{(MPS)}$ over input data coarse grained through $N_{d4}+1$ wavelet layers and projected back to N_{d4} layers. Each data set was initially constructed by selecting p data points within the set N_{fit}. The optimization was carried over 40 sweeps, keeping singular values above $\Delta = 1 \times 10^{-9}$.

presence of disentanglers, MERA are able to approximate non-trivial families of wavelets with overlapping support, such as Daubechies-4 wavelets.

The fact that a MERA is a tensor network composed of multi-linear transformations allows our model to have an interesting reversibility property: the trainable parameters of the model can be projected to a finer scale of resolution while preserving the output of the model, allowing this procedure to initialize more expressive models by initially training models with fewer parameters. This initialization step showed notable improvement in the accuracy of the model as compared to the direct initialization of the weights at any given scale, for a fixed number of optimization sweeps at the finer scale. This is a unique distinguishing feature of our model. As shown in table 2, by comparing the highest accuracy generated by the fine-graining initialization methodology of the wavelet-MERA algorithm, in the case of binary classification, it is clear that our model can yield comparable accuracy to neural network models which were ranked in the top twenty for acoustic classification models of the 2017 DCASE challenge [41–43]. Further work must be done to improve this model in the context of multi-class classification. However, it is worth noting that the uniqueness of this model stands out in that these neural network models are fixed after training, while our model 'improves itself' by re-scaling a previously trained set of weights for a new initialization of the training.

Among techniques widely used in machine learning, our model architecture most closely resembles a convolutional neural network (CNN) [44]. In both architectures, data is initially processed through a set of layers which mix information in a locality-preserving way. Pooling layers commonly used in CNNs closely resemble the isometry maps in a MERA, which act as a coarse-graining transformation. The connection to CNNs suggests that future directions worth exploring include making the parameters of the MERA layers adjustable (i.e. training them along with the weights in the top layer) or finding a more optimal method for the definition and implementation of disentanglers. More ambitiously, instead of just training the parameters in a given MERA layer, one can envision computing this layer from a set of weights at a given scale through a controlled factorization procedure.

By noting the dependence of the accuracy of our model on the number of wavelet layers, we deduce that the input data can exhibit correlations at length scales that can be lost through the wavelet transformations. It is therefore important to tailor the number of wavelets and initial size to the specific data set being analyzed in order to maintain a desirable accuracy. Our setup gives an affordable and adaptive way to strike a balance between the efficiency and generalization gains obtained by coarse-graining versus model expressivity by trading off one for the other through the fine-graining procedure. One way to select the best number of layers to use would be to use too many intentionally, then fine-grain until the gains in model performance begin to saturate, or until generalization starts to degrade.

We conclude that our algorithm provides an interesting platform for classification and regression, with unique capabilities. Further work is needed to improve the model accuracy for the classification of continuous labels (i.e. regression). It is also worth investigating the effect that different wavelet transformations may have on the model; determining how much is gained by introducing optimizable parameters into the coarse-graining layers; and investigating adaptive learning schemes for the sizes of indices in the MERA layers. We emphasize that our MERA realizes daubechie wavelet transformations corresponding to linear features, and further work must be done to extend this approach to the space of non-linear features.

Data availability statement

The data that support the findings of this study are available upon reasonable request from the authors.

Acknowledgments

J R was partially supported by NSF Grant Nos. CCF-1525943 and CCF-1844434. E M S is supported by the Flatiron Institute, a division of the Simons Foundation.

ORCID iD

J A Reyes https://orcid.org/0000-0002-6330-9801

References

[1] Marcello D C, Caccin M, Baireuther P, Hyart T and Fruchart M 2019 Machine learning assisted measurement of local topological invariants (arXiv:1906.03346)
[2] Chinni C, Kulkarni A, Pai D M, Mitra K and Sarvepalli P K 2019 Neural decoder for topological codes using pseudo-inverse of parity check matrix (arXiv:1901.07535)
[3] Rodriguez-Nieva J F and Scheurer M S 2019 Identifying topological order through unsupervised machine learning *Nat. Phys.* **15** 790–5
[4] Nautrup H P, Delfosse N, Dunjko V, Briegel H J and Friis N 2018 Optimizing quantum error correction codes with reinforcement learning (arXiv:1812.08451)
[5] Melkinov A, Fedichkin L and Alodjants A 2019 Detecting quantum speedup by quantum walk with convolutional neural networks (arXiv:1901.10632)
[6] Samaniego E, Anitescu C, Goswami S, Nguyen-Thanh V M, Guo H, Zhuang X and Rabczuk T 2020 An energy approach to the solution of partial differential equations in computational mechanics via machine learning: concepts, implementation and applications *Comput. Methods Appl. Mech. Eng.* **362** 112790
[7] Novikov A, Podoprikhin D, Osokin A and Vetrov D 2015 Tensorizing neural networks (arXiv:1509.06569)
[8] Novikov A, Trofimov M and Oseledets I 2016 Exponential machines (arXiv:1605.03795)
[9] Stoudenmire E M and Schwab D J 2017 Supervised learning with quantum inspired tensor networks *Adv. Neural Inf. Process. Syst.* **29** 4799
[10] Stoudenmire E M 2018 Learning relevant features of data with multi-scale tensor networks *Quant. Sci. Tech.* **3** 034003
[11] Glasser I, Pancotti N and Cirac J I 2018 Supervised learning with generalized tensor networks (arXiv:1806.05964)
[12] Yu R, Zheng S, Anandkumar A and Yue Y 2017 Long term forecasting using tensor-train RNNs (arXiv:1711.00073)
[13] Han Z-Y, Wang J, Fan H, Wang L and Zhang P 2018 Unsupervised generative modeling using matrix product states *Phys. Rev.* X **8** 031012
[14] Liu D, Ran S-J, Wittek P, Peng C, Garcia R B, Su G and Lewenstein M 2019 Machine learning by unitary tensor network of hierarchical tree structure *New J. Phys.* **21** 073059
[15] Cheng S, Wang L, Xiang T and Zhang P 2019 Tree tensor networks for generative modeling *Phys. Rev.* B **99** 155131
[16] Levine Y, Yakira D, Cohen N and Shashua A 2017 Deep learning and quantum entanglement: fundamental connections with implications to network design (arXiv:1704.01552)
[17] Glasser I, Sweke R, Pancotti N, Eisert J and Ignacio Cirac J 2019 Expressive power of tensor-network factorizations for probabilistic modeling, with applications from hidden Markov models to quantum machine learning (arXiv:1907.03741)
[18] Batselier K, Cichocki A and Wong N 2019 MERACLE: constructive layer-wise conversion of a tensor train into a MERA (arXiv:1912.09775)
[19] Kossaifi J, Lipton Z C, Koleinsson A, Khanna A, Furlanello T and Anandkumar A 2020 Tensor regression networks *JMLR* **21** 1–21
[20] Bradley T-D, Stoudenmire E M and Terilla J 2019 Modeling sequences with quantum states: a look under the hood (arXiv:1910.07425)
[21] Orus R 2014 A practical introduction to tensor network states: matrix product states and projected entangled pair states *Ann. Phys., NY* **349** 117–58
[22] Cichocki A 2014 Tensor networks for big data analytics and large-scale optimization problems. (arXiv:1407.3124)
[23] Vidal G 2003 Efficient classical simulation of slightly entangled quantum computations *Phys. Rev. Lett.* **91** 147902
[24] Perez-Garcia D, Verstraete F, Wolf M M and Cirac J I 2007 Matrix product state representations *Quantum Info. Comput.* **7** 401–30
[25] Schollwöck U 2011 The density matrix renormalization group in the age of matrix product states *Ann. Phys., NY* **326** 96–192
[26] Oseledets I 2011 Tensor-train decomposition *SIAM J. Sci. Comput.* **33** 2295–317
[27] Vidal G 2008 A class of quantum many body systems that can be efficiently simulated *Phys. Rev. Lett.* **101** 110501
[28] Evenbly G and Vidal G 2009 Algorithms for entanglement renormalization *Phys. Rev.* B **79** 144108
[29] Vidal G 2007 Algorithms for entanglement renormalization (arXiv version 2) (arXiv:0707.1454v2)
[30] Dolfi M, Bauer B, Troyer M and Ristivojevic Z 2012 Multigrid algorithms for tensor network states *Phys. Rev. Lett.* **109** 020604
[31] Walker J 1999 *A Primer of Wavelets and Their Scientific Applications* (Boca Raton, FL: CRC Press)
[32] Evenbly G and Vidal G 2011 Quantum criticality with the multi-scale entanglement renormalization ansatz (arXiv:1109.5334)
[33] White S 1992 Density matrix formulation for quantum renormalization groups *Phys. Rev. Lett.* **69** 2863–6
[34] Wilson K G 1979 Problems in physics with many scales of length *Sci. Am.* **241** 158–79
[35] Vidal G 2007 Entanglement renormalization *Phys. Rev. Lett.* **99** 220405
[36] Evenbly G and White S R 2016 Entanglement renormalization and wavelets *Phys. Rev. Lett.* **116** 140403
[37] Evenbly G and White S R 2018 Representation and design of wavelets using unitary circuits *Phys. Rev.* A **97** 052314
[38] Paeckel S, Köhler T, Andreas Swoboda S R Manmana U S and Hubig C 2019 Time-evolution methods for matrix-product states *Ann. Phys., NY* **411** 167998
[39] Vidal G 2004 Efficient simulation of one-dimensional quantum many-body systems *Phys. Rev. Lett.* **93** 040502

[40] IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events 2017 (available at: http://www.cs.tut.fi/sgn/arg/dcase2017/challenge/)
[41] Mun S, Park S, Han D K and Ko H 2017 Generative adversarial network based acoustic scene training set augmentation and selection using SVM-hyperplane (available at: http://dcase.community/documents/challenge2017/technical-reports/DCASE2017-Mun-213.pdf)
[42] Weiping Z, Jiantao Y, Xiaotao X and Shaohu P 2017 Acoustic scene classification using deep convolutional neural network and multiple spectrograms fusion (available at: www.cs.tut.fi/sgn/arg/dcase2017/documents/challenge-technical-reports/DCASE2017-Xing-158.pdf)
[43] Kukanov I, Hautamaki V and Lee K A 2017 Recurrent neural network and maximal figure of merit for acoustic event detection (available at: www.cs.tut.fi/sgn/arg/dcase2017/documents/challenge-technical-reports/DCASE2017-Kukanov-196.pdf)
[44] Schmidhuber J 2015 Deep learning in neural networks: an overview *Neural Netw.* **61** 85–117