Compressing deep neural networks by matrix product operators

Ze-Feng Gao,[1] Song Cheng,[2,3,4] Rong-Qiang He,[1] Z. Y. Xie,[1,*] Hui-Hai Zhao,[5,†] Zhong-Yi Lu,[1,‡] and Tao Xiang[2,4,§]

[1]*Department of Physics, Renmin University of China, Beijing 100872, China*
[2]*Institute of Physics, Chinese Academy of Sciences, Beijing 100190, China*
[3]*Center for Quantum Computing, Peng Cheng Laboratory, Shenzhen 518055, China*
[4]*University of Chinese Academy of Sciences, Beijing, 100049, China*
[5]*RIKEN Brain Science Institute, Hirosawa, Wako-shi, Saitama, 351-0106, Japan*

(Received 8 May 2019; accepted 12 May 2020; published 8 June 2020)

A deep neural network is a parametrization of a multilayer mapping of signals in terms of many alternatively arranged linear and nonlinear transformations. The linear transformations, which are generally used in the fully connected as well as convolutional layers, contain most of the variational parameters that are trained and stored. Compressing a deep neural network to reduce its number of variational parameters but not its prediction power is an important but challenging problem toward the establishment of an optimized scheme in training efficiently these parameters and in lowering the risk of overfitting. Here we show that this problem can be effectively solved by representing linear transformations with matrix product operators (MPOs), which is a tensor network originally proposed in physics to characterize the short-range entanglement in one-dimensional quantum states. We have tested this approach in five typical neural networks, including FC2, LeNet-5, VGG, ResNet, and DenseNet on two widely used data sets, namely, MNIST and CIFAR-10, and found that this MPO representation indeed sets up a faithful and efficient mapping between input and output signals, which can keep or even improve the prediction accuracy with a dramatically reduced number of parameters. Our method greatly simplifies the representations in deep learnin, and opens a possible route toward establishing a framework of modern neural networks which might be simpler and cheaper, but more efficient.

DOI: 10.1103/PhysRevResearch.2.023300

I. INTRODUCTION

Deep neural networks [1–11] are important tools of artificial intelligence. Their applications in many computing tasks, for example, in the famous ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [12], large vocabulary continuous speech recognition [13], and natural language processing [14], have achieved great success. They have become the most popular and dominant machine-learning approaches [15] that are used in almost all recognition and detection tasks [3,16], including but not limited to language translation [17], sentiment analysis [18], segmentation and reconstruction [19], drug activity prediction [20], feature identification in big data [21], and have attracted increasing attention from almost all natural science and engineering communities, including mathematics [22,23], physics [24–28], biology [19,29], and materials science [30].

A deep feedforward neural network sets up a mapping between a set of input signals, such as images, and a set of output signals, say categories, through a multilayer transformation, $\mathcal{F}$, which is represented as a composition of many alternatively arranged linear ($\mathcal{L}$) and nonlinear ($\mathcal{N}$) mappings [31,32]. More specifically, an n-layer neural network $\mathcal{F}$ is a sequential product of alternating linear and nonlinear transformations:

$$\mathcal{F} = \mathcal{N}_n\mathcal{L}_n \cdots \mathcal{N}_2\mathcal{L}_2\mathcal{N}_1\mathcal{L}_1. \tag{1}$$

The linear mappings contain most of the variational parameters that need to be determined. The nonlinear mappings, which contain almost no free parameters, are realized by some operations known as activations, including rectified linear unit, softmax, and so on.

A linear layer maps an input vector $\mathbf{x}$ of dimension N_x to an output vector $\mathbf{y}$ of dimension N_y via a linear transformation characterized by a weight matrix W:

$$\mathbf{y} = W\mathbf{x} + \mathbf{b}. \tag{2}$$

A fully connected layer plays the role as a global linear transformation, in which each output element is a weighted summation of all input elements, and W is a full matrix. A convolutional layer [2] represents a local linear transformation, in the sense that each element in the output is a weighted summation of a small portion of the elements, which form a local cluster, in the input. The variational weights of this local cluster form a dense convolutional kernel, which is designated to extract some specific features. To maintain good performance,

*qingtaoxie@ruc.edu.cn
†huihai.zhao@riken.jp
‡zlu@ruc.edu.cn
§txiang@iphy.ac.cn

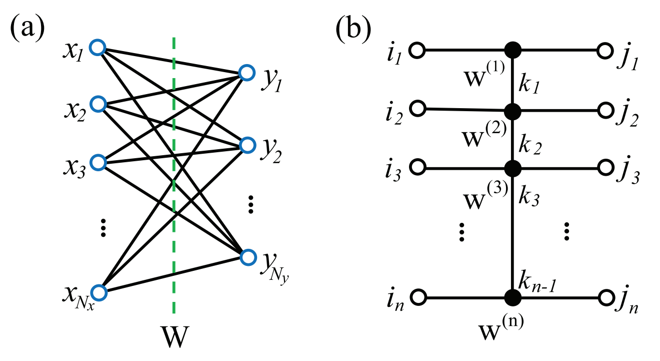

FIG. 1. (a) Graphical representation of the weight matrix W in a fully connected layer. The blue circles represent neurons, e.g., pixels. The solid line connecting an input neuron x_i with output neuron y_j represents the weight element W_{ji}. (b) MPO factorization of the weight matrix W. The local operators $w^{(k)}$ are represented by filled circles. The hollow circles denote the input and output indices, i_l and j_l, respectively. Given i_k and j_k, $w^{(k)}[j_k, i_k]$ is a matrix.

different kernels are used to extract different features. A graphical representation of W is shown in Fig. 1(a).

Usually, the number of elements or neurons, N_x and N_y, are very large, and thus there are a huge number of parameters to be determined in a fully connected layer [9]. The convolutional layer reduces the variational parameters by grouping the input elements into many partially overlapped kernels, and one output element is connected to one kernel. The number of variational parameters in a convolutional layer is determined by the number of kernels and the size of each kernel. It could be much less than that in a fully connected layer. However, the total number of parameters in all the convolutional layers can still be very large in a deep neural network which contains many convolutional layers [10]. To train and store these parameters raises a big challenge in this field. First, it is time consuming to train and optimize these parameters, and may even increase the probability of overfitting. This would limit the generalization power of deep neural networks. Second, it needs a big memory space to store these parameters. This would limit its applications where the space of hard disk is strongly confined; for example, on mobile terminals.

There are similar situations in the context of quantum information and condensed-matter physics. In a quantum many-body system, the Hamiltonian or any other physical operator can be expressed as a higher-order tensor in the space spanned by the local basis states [33]. To represent exactly a quantum many-body system, the total number of parameters that need to be introduced can be extremely huge, and should in principle grow exponentially with the system size (or the size of each "image" in the language of neural network). The matrix product operator (MPO) was originally proposed in physics to characterize the short-range entanglement in one-dimensional quantum systems [34,35], and is now a commonly used approach to represent effectively a higher-order tensor or Hamiltonian with short-range interactions. Mathematically, it is simply a tensor-train approximation [36,37] that is used to factorize a higher-order tensor into a sequential product of the so-called local tensors. Using the MPO representation, the number of variational parameters needed is greatly reduced since the number of parameters contained in an MPO just grows linearly with the system size. Nevertheless, it turns out that to provide an efficient and faithful representation of the systems with short-range interactions whose entanglement entropies are upper bounded [38,39] or, equivalently, the systems with finite excitation gaps in the ground states. The application of MPOs in condensed-matter physics and quantum information science has achieved great successes [40,41] in the past decade.

In this paper, we propose to solve the parameter problem in neural networks by employing the MPO representation, which is illustrated in Fig. 1(b) and expressed in Eq. (5). The starting point is the observation that the linear transformations in a commonly used deep neural network have a number of similar features as the quantum operators, which may allow us to simplify their representations. In a fully connected layer, for example, it is well known that the rank of the weight matrix is strongly restricted [42–44] due to short-range correlations or entanglements among the input pixels. This suggests that we can safely use a lower-rank matrix to represent this layer without affecting its prediction power. In a convolutional layer, the correlations of images are embedded in the kernels, whose sizes are generally very small in comparison with the whole image size. This implies that the "extracted features" from this convolution can be obtained from very local clusters. In both cases, a dense weight matrix is not absolutely necessary to perform a faithful linear transformation. This peculiar feature of linear transformations results from the fact that the information hidden in a data set is just short-range correlated. Thus, to accurately reveal the intrinsic features of a data set, it is sufficient to use a simplified representation that catches more accurately the key features of local correlations. This motivates us to adopt MPOs to represent linear transformation matrices in deep neural networks.

There have been several applications of tensor network structures in neural networks [37,45–50]. Our approach differs from them by the following aspects: (1) It is physically motivated, emphasizes more on the local structure of the relevant information, and helps to understand the success of deep neural networks. (2) It works in the framework of neural networks, in the sense that the multiple-layer structure and activation functions are still retained and the parameters are entirely optimized through algorithms developed in neural networks. (3) It is a one-dimensional representation, and is flexible to represent the linear transformations including both the fully connected layers and the entire convolutional layers. (4) It is also used to characterize the complexity of image data sets. (5) A systematic study has been done. These issues will become clear in the following sections.

The rest of the paper is structured as follows. In Sec. II, we present the way the linear layers can be represented by MPO and the training algorithm of the resulting network. In Sec. III, we apply our method systematically to five main neural networks, including FC2, LeNet-5, VGG, ResNet, and DenseNet on two widely used data sets, namely, MNIST and CIFAR-10. Experiments on more data sets can be found in Sec. II. A in the Supplemental Material (SM) [51]. Finally, in Sec. IV, we discuss the relation with previous efforts and the possibility to construct a framework of neural networks based on the matrix product representations in the future. In the SM

[51], we give the detailed structure of the neural networks used in this work, and provide extra information details about the MPO representations.

II. METHOD

In this paper, the weight matrices W appearing in Eq. (2) and representing linear mappings in the most parameter-consuming layers—to be precise, all the fully connected layers and some of the heaviest convolutional layers—are expressed as MPOs. To construct the MPO representation of a weight matrix W, we first reshape it into a $2n$-indexed tensor:

$$W_{yx} = W_{j_1 j_2 \cdots j_n, i_1 i_2 \cdots i_n}. \tag{3}$$

Here, the one-dimensional coordinate x of the input signal $\mathbf{x}$ with dimension N_x is reshaped into a coordinate in an n-dimensional space, labeled $(i_1 i_2 \cdots i_n)$. Hence, there is a one-to-one mapping between x and $(i_1 i_2 \cdots i_n)$. Similarly, the one-dimensional coordinate y of the output signal $\mathbf{y}$ with dimension N_y is also reshaped into a coordinate in an n-dimensional space, and there is a one-to-one correspondence between y and $(j_1 j_2 \cdots j_n)$. If I_k and J_k are the dimensions of i_k and j_k, respectively, then

$$\prod_{k=1}^{n} I_k = N_x, \quad \prod_{k=1}^{n} J_k = N_y. \tag{4}$$

The index decomposition in Eq. (3) is not unique. One should in principle decompose the input and output vectors such that the test accuracy is the highest. However, to test all possible decompositions is time consuming. For the results presented in this paper, we have done the decomposition just by convenience, i.e., simply reshaping the single dimension N_x as n parts $\{I_1, I_2, ..., I_n\}$ in order. In fact, it can be argued and verified by examples that when the network is away from underfitting, different factorization manners should always produce almost the same result. More details can be found in Sec. II. B in the SM [51].

The MPO representation of W is obtained by factorizing it into a product of n local tensors,

$$W_{j_1\cdots j_n, i_1\cdots i_n} = \mathrm{Tr}(w^{(1)}[j_1, i_1] w^{(2)}[j_2, i_2] \cdots w^{(n)}[j_n, i_n]), \tag{5}$$

where $w^{(k)}[j_k, i_k]$ is a $D_{k-1} \times D_k$ matrix with D_k the dimension of the bond linking $w^{(k)}$ and $w^{(k+1)}$. In this case, $D_0 = D_n = 1$. For convenience in the discussion below, we assume $D_k = D$ for all k except $k = 0$ or n. A graphical representation of this MPO is shown in Fig. 1(b).

In this MPO representation, the tensor elements of $w^{(k)}$ are variational parameters. The number of parameters increases with the increase of the bond dimension D. Hence D serves as a tunable parameter that controls the expressive power. In quantum many-body systems, D also controls the expressive accuracy of a target state variationally.

The tensor elements of $w^{(k)}$ in Eq. (5), instead of the elements of W in Eq. (2), are the variational parameters that need to be determined in the training procedure of deep neural networks. For an MPO whose structure is defined by Eq. (8), the total number of these variational parameters equals

$$N_{\mathrm{mpo}} = \sum_{k=2}^{n-1} I_k J_k D^2 + I_1 J_1 D + I_n J_n D, \tag{6}$$

which will be a great reduction of the number $N_x N_y$ in the original fully connected layers (when N_x and N_y are large) and of the number $N_k N_0$ in the original convolutional layers (when the kernel size N_0 and the number of kernels N_k are large).

The strategy of training is to find a set of optimal w's so the following cost function is minimized,

$$L = -\sum_m t_m^T \log y_m + \frac{\alpha}{2} \sum_i |w^{(i)}|^2, \tag{7}$$

where m is the label of images, i is the label of all the parameters, including the local tensors in the MPO representations and the kernels in the untouched convolutional layers. $|w|$ represents the norm of parameter w, and α is an empirical parameter that is fixed prior to the training. The first term measures the cross entropy between prediction vectors y and target label vectors t. The second term is a constraint, called the L2 regularization [52], which is a widely adopted technique in deep learning to alleviate overfitting, and thus it is also used in all the networks mentioned in this paper, including both the normal neural networks, such as FC2, LeNet-5, VGG, ResNet, as well as DenseNet, and the corresponding MPO-Net counterparts. It should be mentioned that the usage of the L2 regularization has little to do with the validity of the MPO representations, i.e., without L2 regularization, the MPO-Nets should still work as well as the normal networks, as shown in Sec. II. D in the SM [51].

To implement a training step, L is evaluated using the known w's, which are randomly initialized, and the input data set. The gradients of the cost function with respect to the variational parameters are determined by the standard back propagation [53]. All the parameters w are treated equally and are updated by the stochastic gradient descent with momentum algorithm [54] in parallel. This is different from the previous effort [37] and is more suitable for deep learning. This training step is terminated when the cost function stops to drop.

The detailed structures of the neural networks we have studied, as well as the performance on more data sets, different factorization manners, entanglement entropy grasped, the influence of L2 regularization, convergence of training, and so on, are appended systematically in the SM [51]. The specific setting of the hyperparameters for training can be found in the source code [55].

III. RESULTS

Here we show the results obtained with the MPO representation in five kinds of typical neural networks on two data sets, i.e., FC2 [56] and LeNet-5 [2] on the MNIST data set [57]; VGG [9], ResNet [10], and DenseNet [11] on the CIFAR-10 data set [58]. Among them, FC2 and LeNet-5 are relatively shallow in the depth of network, while VGG, Residual CNN (ResNet), and Dense CNN (DenseNet) are deeper neural networks.

For convenience, we use MPO-Net to represent a deep neural network with all or partial linear layers being represented by MPOs. Moreover, we denote an MPO, defined by Eq. (5), as

$$M^{J_1,J_2,\ldots,J_n}_{I_1,I_2,\ldots,I_n}(D). \tag{8}$$

To quantify the compressibility of MPO-net with respect to a neural network, we define its compression ratio ρ as

$$\rho = \frac{\sum_l N^{(l)}_{\text{mpo}}}{\sum_l N^{(l)}_{\text{ori}}}, \tag{9}$$

where $\sum_l$ is to sum over the linear layers whose transformation tensors are replaced by MPO. $N^{(l)}_{\text{ori}}$ and $N^{(l)}_{\text{mpo}}$ are the number of parameters in the lth layer in the original and MPO representations, respectively. The smaller is the compression ratio, the fewer number of parameters is used in the MPO representation.

Furthermore, to examine the performance of a given neural network, we train the network m times independently to obtain a test accuracy a with a standard deviation σ defined by

$$a = \bar{a} \pm \sigma, \tag{10}$$

$$\sigma = \frac{1}{\sqrt{m-1}}\left[\sum_{i=1}^{m}(a_i - \bar{a})^2\right]^{1/2}, \tag{11}$$

where a_i is the test accuracy of the i-th training procedure. $\bar{a}$ is the average of $\{a_i\}$. The results presented in this paper are obtained with $m = 5$.

A. MNIST data set

We start from the identification of handwritten digits in the MNIST data set [57], which consists of 60 000 digits for training and 10 000 digits for testing. Each image is a square of 28×28 grayscale pixels, and all the images are divided into ten classes corresponding to numbers $0 \sim 9$, respectively.

1. FC2

We first test the MPO representation in the simplest textbook structure of neural network, i.e., FC2 [56]. FC2 consists of only two fully connected layers whose weight matrices have 784×256 and 256×10 elements, respectively. We replace these two weight matrices, respectively, by $M^{4,4,4,4}_{4,7,7,4}(D)$ and $M^{1,1,10,1}_{4,4,4,4}(4)$ in the corresponding MPO representation. Here we fix the bond dimension in the second layer to 4, and only allow the bond dimension to vary in the first layer.

Figure 2 compares the results obtained with FC2 and the corresponding MPO-Net. The test accuracy of MPO-Net increases when the bond dimension D is increased. It reaches the accuracy of the normal FC2 when $D = 16$. Even for the $D = 2$ MPO-Net, which has only 1024 parameters, about 200 times less than the original FC2, the test accuracy is already very good. This shows that the linear transformations in FC2 are very local and can indeed be effectively represented by MPOs. The compression ratio of MPO-Net decreases with increasing D. But even for $D = 16$, the compression ratio is still below 8%, which indicates that the number of parameters to be trained can be significantly reduced without any accuracy loss.

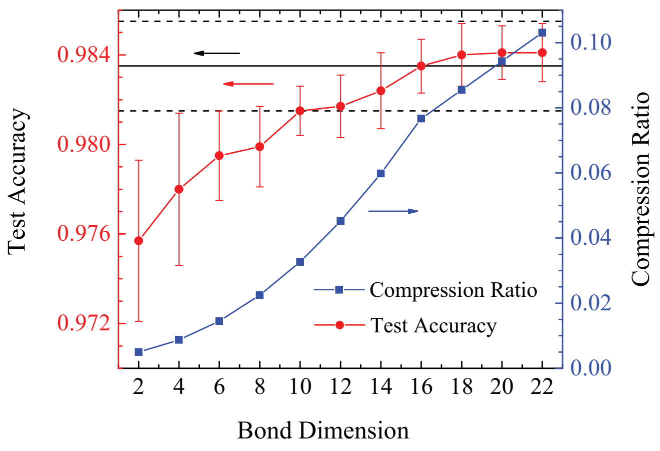

FIG. 2. Performance of the MPO representations in FC2 on MNIST. The solid straight line denotes the test accuracy obtained by the normal FC2, 98.35% ± 0.2%, and the dashed straight lines are plotted to indicate its error bar.

2. LeNet-5

We further test MPO-Net with the famous LeNet-5 network [2], which is the first instance of convolutional neural networks. LeNet-5 has five linear layers. Among them, the last convolutional layer and the two fully connected layers contain the most parameters. We represent these three layers by three MPOs, which are structured as $M^{2,5,6,2}_{2,10,10,2}(4)$, $M^{2,3,7,2}_{2,5,6,2}(4)$, and $M^{1,5,2,1}_{2,3,7,2}(2)$, respectively. The compression ratio is $\rho \sim 0.05$.

Table I shows the results obtained with the original and MPO representations of LeNet-5. We find that the test accuracy of LeNet-5 can be faithfully reproduced by MPO-Net. Since LeNet-5 is the first and prototypical convolutional neural network, this success gives us confidence in using the MPO presentation in deeper neural networks.

B. CIFAR-10 data set

CIFAR-10 is a more complex data set [58]. It consists of 50 000 images for training and 10 000 images for testing. Each image is a square of 32×32 RGB pixels. All the images in this data set are divided into ten classes corresponding to airplane, automobile, ship, truck, bird, cat, deer, dog, frog, and horse, respectively. To have a good classification accuracy, deeper neural networks with many convolutional layers are used. To show the effectiveness of MPO representation, as a preliminary test, we use MPOs only on the fully connected

TABLE I. Test accuracy a and compression ratios ρ obtained in the original and MPO representations of LeNet-5 on MNIST and VGG on CIFAR-10.

		Original Rep	MPO-Net	
Data set	Network	a (%)	a (%)	ρ
MNIST	LeNet-5	99.17 ± 0.04	99.17 ± 0.08	0.05
CIFAR-10	VGG-16	93.13 ± 0.39	93.76 ± 0.16	$\sim$0.0005
	VGG-19	93.36 ± 0.26	93.80 ± 0.09	$\sim$0.0005

layers and on some heavily parameter-consuming convolutional layers.

1. VGG

VGG [9] is the first *very* deep neural network contrusted. It won first place in the localization task of the ILSVRC competition 2014. We have tested two well-established VGG structures, which have 16 and 19 layers, respectively. In both cases, there are many convolutional layers and three fully connected layers. We represent the last two heaviest convolutional layers and all the three fully connected layers, respectively, by MPO with the structures: $M^{2,8,8,8,2}_{2,8,8,8,2}(4)$, $M^{2,8,8,8,2}_{2,8,8,8,2}(4)$, $M^{4,4,8,8,4}_{4,4,4,4,2}(4)$, $M^{4,4,8,8,4}_{4,4,8,8,4}(4)$, and $M^{1,10,1,1,1}_{4,4,8,8,4}(4)$. The result is summarized in Table I.

For both structures, the compression ratio of MPO-Net is about 0.0005. Hence the number of parameters used is much less than in the original representation. However, we find that the prediction accuracy of MPO-Net is even better than those obtained from the original networks. This is consistent with the results reported by Novikov [37] for the ImageNet data set [59]. It results from two facts of MPOs: First, since the number of variational parameters is greatly reduced in MPO-Net, the representation is more economical and it is easier to train the parameters. Second, the local correlations between input and output elements are more accurately represented by MPO. This can reduce the probability of overfitting.

2. ResNet

ResNet [10] is commonly used to address the degradation problem with deep convolutional neural networks. It won first place on the detection task in ILSVRC in 2015, and differs from the ordinary convolutional neural network by the so-called ResUnit structure, in which identity mappings are added to connect some of the input and output signals. The ResNet structure used in our calculation has a fully connected layer realized by a weight matrix of $64k \times 10$. Here k controls the width of the network. We represent this layer by an MPO of $M^{1,5,2,1}_{4,4,4,k}(3)$. In our calculation, $k = 4$ is use and the corresponding compression ratio is about 0.11.

Figure 3 shows the test accuracy as a function of the depth of layers of ResNet with $k = 4$. We find that MPO-Net has the same accuracy as the normal ResNet for all the cases we have studied. We also find that even the ResUnit can be compressed by MPO. For example, for the 56-layer ResNet, by representing the last heaviest ResUnit and the fully connected layer with two $M^{2,4,4,4,4,4,4,4,2}_{2,4,4,4,4,4,4,4,2}(4)$ and one $M^{1,5,2,1}_{4,4,4,k}(3)$, we obtain the same accuracy as the normal ResNet. Similar observations are obtained for other k values.

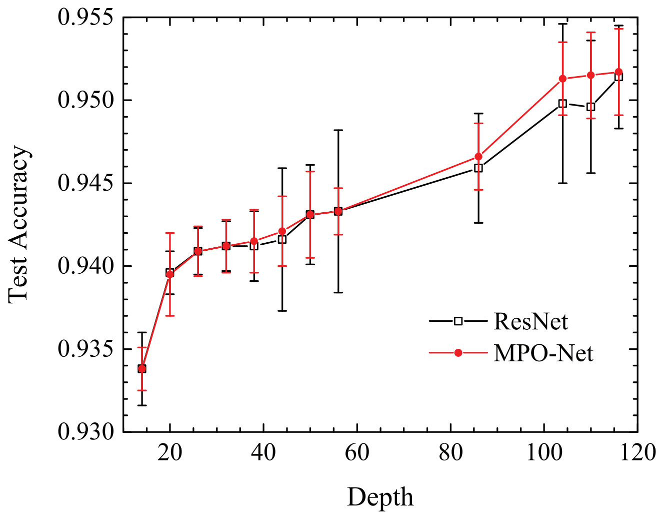

FIG. 3. Comparison of the test accuracy a between the original and MPO representations of ResNet on CIFAR-10 with $k = 4$. The compression ratio of MPO-Net $\rho \sim 0.11$.

3. DenseNet

The last deep neural network we have tested is DenseNet [11]. Constructed in the framework of ResNet, DenseNet modifies ResUnit to DenseUnit by adding more shortcuts in the units. This forms a wider neural network, allowing the extracted information to be more efficiently recycled. It also achieved great success in the ILSVRC competition, and drew much attention in the CVPR conference in 2017.

The DenseNet used in this work has a fully connected layer with a weight matrix of $(n + 3km) \times 10$, where m controls the total depth L of the network, $L = 3m + 4$, n and k are the other two parameters that specify the network. There is only one fully connected layer in DenseNet, and we use MPO to reduce the parameter number in this layer. Corresponding to different m, k, and n, we use different MPO representations.

Our results are summarized in Table II. For the four DenseNet structures we have studied, the fully connected layer is compressed by more than seven to 21 times. The corresponding compression ratios vary from 0.044 to 0.129. In the first three cases, we find that the test precisions obtained with MPO-Net agree with the DenseNet results within numerical errors. For the fourth case, the test accuracy obtained with MPO-Net is even slightly higher than that obtained with DenseNet. Further more, in the first structure listed in Table II, we have also tried to replace the last heaviest

TABLE II. Performance of MPO representations in DenseNet on CIFAR-10.

		Test accuracy (%)			
Depth	(n, m, k)	DenseNet	MPO-Net	MPO structure	ρ
40	(16, 12, 12)	93.56 ± 0.26	93.59 ± 0.13	$M^{1,5,2,1}_{4,4,7,4}(4)$	0.129
40	(16, 12, 24)	95.12 ± 0.15	95.13 ± 0.13	$M^{1,5,2,1}_{4,5,11,4}(4)$	0.089
100	(24, 32, 12)	95.36 ± 0.15	95.58 ± 0.07	$M^{1,5,2,1}_{4,7,7,6}(4)$	0.070
100	(96, 32, 24)	95.74 ± 0.09	96.09 ± 0.07	$M^{1,5,2,1}_{5,8,12,5}(4)$	0.044

convolutional layer by $M^{2,4,4,3,4,2}_{4,5,8,11,4,4}(20)$ while keeping the last fully connected layer represented by $M^{1,5,2,1}_{4,4,7,4}(4)$. The obtained accuracy is about 93.52 ± 0.40, which is still consistent with the original DenseNet result 93.56 ± 0.26. The corresponding compression ratio is also considerable, i.e., $\rho \sim 0.497$.

Applications to more data sets, such as the Fashion-MNIST [60] and Street View House Number data sets [61], can be found in Sec. II. A in the SM [51]. We found that the faithful representation and the effective compression capability of MPO are also valid there.

IV. DISCUSSION

Motivated by the success of MPOs in the study of quantum many-body systems with short-range interactions, we propose to use MPOs to represent linear transformation matrices in deep neural networks. This is based on the assumption that the correlations between pixels, or the inherent structures of information hidden in "images," are essentially localized [62,63], which enables us to make an analogy between an image and a quantum state and further between a linear mapping in neural network and a quantum operator in physics. We have tested our approach with five different kinds of typical neural networks on two data sets, and found that MPOs cannot only improve the efficiency in training and reduce the memory space, as originally expected, but also slightly improve the test accuracy using much fewer number of parameters than in the original networks. This, as already mentioned, may result from the fact that the variational parameters can be more accurately and efficiently trained due to the dramatic reduction of parameters in MPO-Net. The MPO representation emphasizes more on the local correlations of input signals. It puts a strong constraint on the linear transformation matrix and avoids the training data being trapped at certain local minima. We believe this can reduce the risk of overfitting.

In fact, by using the canonical form [64] of an MPO representation obtained from training, we can introduce the entanglement entropy, initially defined for a quantum state in physics, for a data set in deep learning to quantify the expressive ability of the network and the complexity of the data set. This can also help to understand the relation between local correlations in the input data sets and the performance of the MPO-Nets. More details about this topic can be found in Sec. II. C in the SM [51].

MPOs can be used to represent both fully connected and convolutional layers. In our paper, they are not distinguished from each other at all, and are regarded as the same thing, i.e., linear mappings appeared in Eq. (1). One can also use it just to represent the kernels in convolutional layers, which is a substantially different approach to use MPOs since the convolutional structure is still retained, as suggested by Garipov *et al.* [45]. However, it is more efficient in representing a fully connected layer where the weight matrix is a fully dense matrix. This representation can greatly reduce the memory cost and shorten the training time in a deep neural network where all or most of the linear layers are fully connected ones, such as in a recurrent neural network [46,65,66], which is used to dispose of video data.

Tensor-network representation of deep neural networks is actually not new. Inspired by the locality assumption about the correlations between pixels, matrix product representation has been already successfully used to characterize and compress images [62] and to determine the underlying generative models [67]. Novikov *et al.* [37] also used MPOs to represent some fully connected layers, not including the classifiers, in FC2 and VGG. Our work, however, demonstrates that all fully connected layers, including the classifiers especially, as well as convolutional layers, can be effectively represented by MPOs no matter how deep a neural network is. In other words, in our approach there are no concepts of fully connected layers or convolutional layers but only linear mappings expressed as sparse MPO and parameter-free nonlinear activations. We think this is a great simplification for both concepts and applications and is of great potential due to the much less required memory space and relatively mathematical structure to study. Our work will greatly help the application of neural networks and especially may help to get rid of connection to the cloud.

There are some previous efforts which aim to establish the entire mapping from the input data to the output label, e.g., Stoudenmire and Schwab tried to represent the mapping in terms of a single MPO [50]. Our proposal differs from it since we are still working in the framework of neural networks, in the sense that the multiple-layer structure and activation functions are still retained. There are also other mathematical structures that have been used to represent deep neural networks due to entanglement consideration from physics. For example, Kossaifi *et al.* [47] used a Tucker-structure representation, which is a lower-rank approximation of a high-dimensional tensor, to represent a fully connected layer and its input feature. Hallam *et al.* [48] used a tensor network called a multiscale entangled renormalization ansatz [68] and Liu *et al.* [49] used an unitary tree tensor network [69] to represent the entire mapping from the input to the output labels. Comparing with these works, our approach is a one-dimensional representation which emphasizes more on local entanglement in physics, and it is more efficient and flexible to represent some intermediate layers.

It is valuable to mention two aspects about the MPO representation. One is about its application scope. Due to the locality assumption, it is expected to work efficiently in the data sets where locality can be appropriately defined; otherwise, a large bond dimension would be necessary, which might lead to the loss of efficiency and advantage. The other is about the manner how the input data is ordered when it is fed to the MPO-Net. It is instructive to notice that different orderings of an input vector are related by elementary transformations; therefore, they should lead to the same prediction, in principle, as long as the bond dimension is sufficiently large; while given a small bond dimension, the ordering which keeps better the locality may lead to higher prediction accuracy. A coarse-grained ordering which can better characterize the locality of the original image was proposed in Ref. [62] and is worth being studied systematically in the future.

In this paper, we have proposed to use MPOs to compress the transformation matrices in deep neural networks. Similar ideas can be used to compress complex data sets, for example, the data set called ImageNet [59], in which each image contains about 224×224 pixels. In this case, it is matrix product states [70], instead of MPOs, that should be used. We believe this can reduce the cost in decoding each "image" in a data set, and by combining with the MPO representation of the linear transformation matrices, can further compress deep neural networks and enhance prediction power. A preliminary example is shown in Sec. II. E in the SM [51]. Another possible advance in the future is about the analysis of optimization in a neural network. In this work, in most cases, MPO-Nets converge faster than the original networks in the training procedure, and this is probably due to the far fewer parameters. However, in deep learning, due to the strong nonlinearity of the cost function, e.g., Eq. (7), more parameters means higher-dimensional variational space and might have more local minima, thus it is difficult for the current optimization approach, e.g., stochastic gradient descent method, to guarantee a faster convergence speed in a model with fewer parameters. The counterexamples can be found in both normal networks and MPO-Nets, as discussed in Sec. II. E in the SM [51]. By using MPO representations, as a complementary tool, we can study this optimization problem in deep learning from the viewpoint of entanglement entropy developed in quantum physics, as we did preliminarily in the SM [51]. Therefore, based on these matrix product representations stemming from quantum many-body physics, it is possible to establish a framework of modern neural networks which might be simpler, cheaper, but more efficient and better understood. It is also expected that this bridge between quantum many-body physics and deep learning can eventually provide some useful feedback and insight to physics, and we would like to leave the extensive study as a future pursuit.

ACKNOWLEDGMENTS

We thank Andrzej Cichocki, Pan Zhang, and Lei Wang for valuable discussions. This work was supported by the National Natural Science Foundation of China (Grants No. 11774420, No. 11774422, No. 11874421, and No. 11888101), and by the National R&D Program of China (Grants No. 2016YFA0300503 and No. 2017YFA0302900) and by the Research Funds of Renmin University of China (Grants No. 20XNLG19).

Ze-Feng Gao and Song Cheng contributed equally to this work.

[1] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, Backpropagation applied to handwritten zip code recognition, Neural Comput. **1**, 541 (1989).

[2] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE **86**, 2278 (1998).

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, ImageNet classification with deep convolutional neural networks, Adv. NIPS **25**, 1097 (2012).

[4] M. Lin, Q. Chen, and S. Yan, Network in network, arXiv:1312.4400.

[5] R. K. Srivastava, K. Greff, and J. Schmidhuber, Training very deep networks, Adv. NIPS **28**, 2377 (2015).

[6] J. Schmidhuber, Deep learning in neural networks: An overview, Neural Networks **61**, 85 (2015).

[7] G. Larsson, M. Maire, and G. Shakhnarovich, Fractalnet: Ultra-deep neural networks without residuals, arXiv:1605.07648.

[8] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, Going deeper with convolutions, *CVPR* (2015), pp. 1–9, https://www.cv-foundation.org/openaccess/content_cvpr_2015/html/Szegedy_Going_Deeper_With_2015_CVPR_paper.html.

[9] K. Simonyan and A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv:1409.1556.

[10] K. M. He, X. Y. Zhang, S. Q. Ren, and J. Sun, Deep residual learning for image recognition, *CVPR* (2017), pp. 4700–4708, see http://openaccess.thecvf.com/content_cvpr_2016/html/He_Deep_Residual_Learning_CVPR_2016_paper.html.

[11] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberge, Densely connected convolutional networks, *CVPR* (2017), http://openaccess.thecvf.com/content_cvpr_2017/html/Huang_Densely_Connected_Convolutional_CVPR_2017_paper.html.

[12] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and F. F. Li, ImageNet Large Scale Visual Recognition Challenge, Int. J. Comput Vis. **115**, 211 (2015).

[13] T. Sainath, A.-R. Mohamed, B. Kingsbury, and B. Ramabhadran, Deep convolutional neural networks for LVCSR, in *IEEE International Conference on Acoustics, Speech and Signal Processing* (IEEE, Piscataway, NJ, 2013), pp. 8614–8618.

[14] L. Deng and Y. Liu, *Deep Learning in Natural Language Processing* (Springer, Singapore, 2018).

[15] Y. LeCun, Y. Bengio, and G. Hinton, Deep learning, Nature **521**, 436 (2015).

[16] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-R. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, Deep neural networks for acoustic modeling in speech recognition, IEEE Signal Processing Mag. **29**, 82 (2012).

[17] I. Sutskever, O. Vinyals, and Q. V. Le, Sequence to sequence learning with neural networks, Adv. NIPS **27**, 3104 (2014).

[18] H. H. Do, P. Prasad, A. Maag, and A. Alsadoon, Deep learning for aspect-based sentiment analysis: A comparative review, Expert Sys. Appl. **118**, 272 (2019).

[19] M. Helmstaedter, K. L. Briggman, S. C. Turaga, V. Jain, H. S. Seung, and W. DenkHelmstaedter, Connectomic reconstruction of the inner plexiform layer in the mouse retina, Nature **500**, 168 (2013).

[20] J. Ma, R. P. Sheridan, A. Liaw, G. E. Dahl, and V. Svetnik, Deep neural nets as a method for quantitative structure-activity relationships, J. Chem. Inf. Model. **55**, 263 (2015).

[21] P. Baldi, P. Sadowski, and D. Whiteson, Searching for exotic particles in high-energy physics with deep learning, Nat. Commun. **5**, 4308 (2014).

[22] N. Lei, Z. Luo, S. T. Yau, and D. X. Gu, Geometric understanding of deep learning, arXiv:1805.10451.

[23] N. Lei, K. Su, S. T. Yau, and D. X. Gu, A Geometric view of optimal transportation and generative model, Comput. Aided Geometric Design **68**, 1 (2019).
[24] X. Gao and L. M. Duan, Efficient representation of quantum many-body states with deep neural networks, Nat. Commun. **8**, 662 (2017).
[25] J. Carrasquilla and R. G. Melko, Machine learning phases of matter, Nat. Phys. **13**, 431 (2017).
[26] S. H. Li and L. Wang, Neural Network Renormalization Group, Phys. Rev. Lett. **121**, 260601 (2018).
[27] D. Wu, L. Wang, and P. Zhang, Solving Statistical Mechanics Using Variational Autoregressive Networks, Phys. Rev. Lett. **122**, 080602 (2019).
[28] G. Carleo, I. Cirac, K. Cranmer, L. Daudet, M. Schuld, N. Tishby, L. Vogt-Maranto, and L. Zdeborová, Machine learning and the physical sciences, Rev. Mod. Phys. **91**, 045002 (2019).
[29] S. Webb, Deep learning for biology, Nature **554**, 555 (2018).
[30] V. Stanev, C. Oses, A. G. Kusne, E. Rodriguez, J. Paglione, S. Curtarolo, and I. Takeuchi, Machine learning modeling of superconducting critical temperature, npj Comput. Materials **4**, 29 (2018).
[31] K. Hornik, M. Stinchcombe, and H. White, Multilayer feedforward networks are universal approximators, Neural Networks **2**, 359 (1989).
[32] G. Cybenko, Approximation by superpositions of a sigmoidal function, Math. Control Signals Sys. **2**, 303 (1989).
[33] P. A. M. Dirac, *The Principles of Quantum Mechanics*, 4th ed. (Clarendon Press, 1982).
[34] F. Verstraete, J. J. García-Ripoll, and J. I. Cirac, Matrix Product Density Operators: Simulation of Finite-Temperature and Dissipative Systems, Phys. Rev. Lett. **93**, 207204 (2004).
[35] B. Pirvu, V. Murg, J. I. Cirac and F. Verstraete, Matrix product operator representations, New J. of Phys. **12**, 025012 (2010).
[36] I. V. Oseledets, Tensor-Train decomposition, SIAM J. Sci. Comput. **33**, 2295 (2011).
[37] A. Novikov, D. Podoprikhin, A. Osokin, and D. Vetrov, Tensorizing neural networks, Adv. NIPS **28**, 442 (2015).
[38] D. Poulin, A. Qarry, R. Somma, and F. Verstraete, Quantum Simulation of Time-Dependent Hamiltonians and the Convenient Illusion of Hilbert Space, Phys. Rev. Lett. **106**, 170501 (2011).
[39] J. Eisert, M. Cramer, and M. B. Plenio, Colloquium: Area laws for the entanglement entropy, Rev. Mod. Phys. **82**, 277 (2010).
[40] S. R. White, Density Matrix Formulation for Quantum Renormalization Groups, Phys. Rev. Lett. **69**, 2863 (1992).
[41] G. Vidal, Efficient Classical Simulation of Slightly Entangled Quantum Computations, Phys. Rev. Lett. **91**, 147902 (2003).
[42] M. Denil, B. Shakibi, L. Dinh, M. Ranzato, and N. de Freitas, Predicting parameters in deep learning, Adv. NIPS **26**, 2148 (2013).
[43] T. N. Sainath, B. Kingsbury, V. Sindhwani, E. Arisoy, and B. Ramabhadran, Low-rank matrix factorization for deep neural network training with high-dimensional output targets, in *IEEE International Conference on Acoustics, Speech and Signal Processing* (IEEE, Piscataway, NJ, 2013), pp. 6655–6659.
[44] J. Xue, J. Li, and Y. Gong, Restructuring of deep neural network acoustic models with singular value decomposition, Interspeech 2365 (2013).
[45] T. Garipov, D. Podoprikhin, A. Novikov, and D. Vetrov, Ultimate tensorization: Compressing convolutional and fully-connected layers alike, arXiv:1611.03214.
[46] Y. Yang, D. Krompass, and V. Tresp, Tensor-Train recurrent neural networks for video classification, ICML **70**, 3891 (2017).
[47] J. Kossaifi, Z. Lipton, A. Khanna, T. Furlanello, and A. Anandkumar, Tensor regression networks, arXiv:1707.08308.
[48] A. Hallam, E. Grant, V. Stojevic, S. Severini, and A. G. Green, Compact neural networks based on the multiscale entanglement renormalization ansatz, arXiv:1711.03357.
[49] D. Liu, S. J. Ran, P. Wittek, C. Peng, R. B. Garcia, G. Su, and M. Lewenstein, Machine learning by two-dimensional hierarchical tensor networks: A quantum information theoretic perspective on deep architectures, New J. Phys. **21**, 073059 (2019).
[50] E. Stoudenmire, and D. J. Schwab, Supervised learning with tensor networks, Adv. NIPS **29**, 4799 (2016).
[51] See Supplemental Material at http://link.aps.org/supplemental/10.1103/PhysRevResearch.2.023300 for additional information about this work, including the detailed structure of the neural networks used in this work and extra information about the MPO representations.
[52] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (MIT press, 2016).
[53] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, Nature, Learning representations by back-propagating errors, Nature **323**, 533 (1986).
[54] K. P. Murphy, *Machine Learning: A Probabilistic Perspective* (MIT press, 2012).
[55] https://github.com/zfgao66/deeplearning-mpo.
[56] See, e.g., http://www.tensorfly.cn/tfdoc/tutorials/mnist_beginners.html.
[57] The official website of MNIST is available at http://yann.lecun.com/exdb/mnist.
[58] The official website of CIFAR is available at https://www.cs.toronto.edu/~kriz/cifar.html.
[59] The official website of ImageNet is available at http://www.image-net.org.
[60] See, e.g., http://www.worldlink.com.cn/en/osdir/fashion-mnist.html.
[61] The official website of SVHN is available at http://ufldl.stanford.edu/housenumbers/.
[62] J. I. Latorre, Image compression and entanglement, arXiv:quant-ph/0510031.
[63] Some numerical analysis on the MNIST data set based on the entropy calculation has been devoted to the topic, e.g., see Song Cheng, Jing Chen, and Lei Wang, Information perspective to probabilistic modeling: Boltzmann machines versus Born machines, Entropy **20**, 583 (2018).
[64] D. Perez-Garcia, F. Verstraete, M. M. Wolf, J. I. Cirac, Matrix product state representations, Quantum Inf. Comput. **7**, 401 (2007).

[65] S. Hochreiter and J. Schmidhuber, Long short-term memory, Neural Comput. **9**, 1735 (1997).

[66] C. Guo, Z. Jie, W. Lu, and D. Poletti, Matrix product operators for sequence-to-sequence learning, Phys. Rev. E **98**, 042114 (2018).

[67] Z. Y. Han, J. Wang, H. Fan, L. Wang, and P. Zhang, Unsupervised Generative Modeling using Matrix Product States, Phys. Rev. X **8**, 031012 (2018).

[68] G. Vidal, Entanglement Renormalization, Phys. Rev. Lett. **99**, 220405 (2007).

[69] Y. Shi, L. M. Duan, and G. Vidal, Classical simulation of quantum many-body systems with a tree tensor network, Phys. Rev. A. **74**, 022320 (2006).

[70] M. Fannes and B. Nachtergaele, Finitely correlated states on quantum spin chains, Comm. Math. Phys. **144**, 443 (1992).