The 33rd Youth Academic Annual Conference of Chinese Association of Automation (YAC)
May 18-20, 2018, Nanjing, China

Robust Supervised Learning Based on Tensor Network Method

1st YW Chen
Institute of Cyber-Systems and Control
Zhejiang University
Hangzhou, China
ewell@zju.edu.cn

2nd K Guo
College of Control Science and Engineering
Zhejiang University
Hangzhou, China
kguo@zju.edu.cn

3rd Y Pan
Institute of Cyber-Systems and Control
Zhejiang University
Hangzhou, China
ypan@zju.edu.cn

***Abstract*—The formalism of Tensor Network (TN) provides a compact way to approximate many-body quantum states with 1D chain of tensors. The 1D chain of tensors is found to be efficient in capturing the local correlations between neighboring subsystems, and machine learning approaches have been proposed using artificial neural networks (NN) of similar structure. However, a long chain of tensors is difficult to train due to exploding and vanishing gradients. In this paper, we propose methods to decompose the long-chain TN into short chains, which could improve the convergence property of the training algorithm by allowing stable stochastic gradient descent (SGD). In addition, the short-chain methods are robust to network initializations. Numerical experiments show that the short-chain TN achieves almost the same classification accuracy on MNIST dataset as LeNet-5 with less trainable network parameters and connections.**

***Index Terms*—supervised learning, tensor network, matrix product state, tensor train.**

I. Introduction

In the past few years the field of machine learning has been boosted by the evolution of computation hardware and deep learning. Artificial deep neural networks (NN), such as convolutional neural network (CNN) [1] [2] and recurrent neural network (RNN) [3], lie at the heart of this revolution. NN based machine learning has been widely applied in computer vision, natural language processing, etc. Particularly, a lot of efforts have been devoted to apply machine learning methods to aid the research in physics, material science and chemistry [3], [6]. On the contrary, the design of neural network could benefit from the research in fundamental science. For example, there exist NN structure designs inspired by the formalism of Matrix Product State(MPS) which has been extensively studied in condensed matter physics [7], [9].

MPS represents a large class of quantum states whose wavefunctions could be characterized by a tensor network (TN) composed of 1D chain of tensors, or tensor train (TT). Although the dimension of the tensor space on which general quantum states are defined grows exponentially with the increasing number of subsystems, there are certain manybody quantum systems mainly exhibit local correlations. TT representation is very efficient in encoding such quantum states with linearly increasing number of parameters. Recently, the 1D chain structure of the MPS has been explored in classical machine learning tasks. For example, there are attempts to use the MPS method to classify images [10], compress NN layers [11] and learn generative models [12]. The integration between MPS network and supervised learning has been studied for the classification task on MNIST dataset [13], [14], where a long chain TN with 196 nodes of tensors is considered.

However, the long-chain TN is hard to train and sometimes the loss function will freeze at certain point without converging. This is because the nodes of TN are connected by tensor contraction, which involves a large number of matrix multiplications if the TT is long. The product of a large number of matrices could easily lead to explosion or vanishing gradients, and the same issue usually causes the training of conventional deep NN to collapse. In addition, numerical experiments have shown that stochastic gradient descent (SGD) is not applicable to a long-chain TN [13]. In order to address these problems, we consider using short-chain TN to construct the network, and integrate the short-chain TN with conventional linear transformation layer and SGD to reduce the number of trainable parameters and at the same time improve the robustness of the training algorithm. In the next section, we will introduce the framework of supervised learning based on TN. In Section III, we will present two short-chain methods. Numerical experiments and conclusion are given in Section IV and V.

II. Preliminaries

A. Tensor Network and Tensor Train

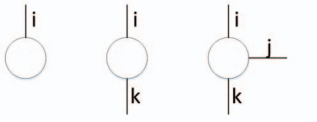

Fig. 1. Diagrammatic notation for a vector, matrix and order-3 tensor (from left to right).

Tensors are multi-dimensional generalizations of matrices and vectors [7]. Tensor is a multi-dimensional array denoted by $A_{i_1 i_2 \cdots i_n}$ with i_j, $j \in \{1, 2, \cdots, n\}$ being the indices of the array. n is the order of the tensor. The diagrammatic representations of tensors are shown in "Fig. 1". Each tensor

node is drawn as a circle with edges. Circle represents the main body and edges represent the individual indices of the tensor. Therefore, a circle with one edge indicates a vector. If the circle has two edges, the node corresponds to a matrix.

One of the most important tensor operations is contraction, which refers to taking the inner product of two tensors along a specified index. The nodes in TN are connected by tensor contractions. An example in "Fig. 2" describes the contraction of two order-3 tensors along the edge j. Singular Value

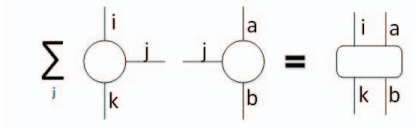

Fig. 2. Diagrammatic notation for tensor contraction between two order-3 tensors.

Decomposition (SVD) can be applied on tensors as well. "Fig. 3" shows the process of decomposing one order-4 tensor into one 2-order tensor and two 3-order tensors through SVD. More details can be found in [13]. Tensor network is a

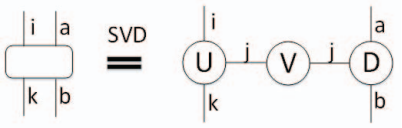

Fig. 3. Diagrammatic notation for singular-value decomposition into 3 nodes.

diagram tells us how to combine several tensors into a single composite tensor [7].MPS or TT is one of the tensor network representations, see "Fig. 4".

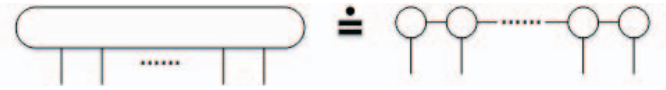

Fig. 4. The matrix product state (MPS) also called tensor train (TT) decomposition of a multi-index tensor.

B. Supervised Learning

"Fig. 5" depicts the structure of an artificial NN.

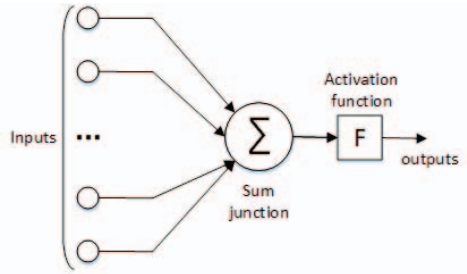

Fig. 5. From left to right, inputs are summed before the activation node.

Normally the samples will be divided into train set and test set. In this paper, we focus on supervised learning where each sample is associated with a correct label. Training is the process of fitting a transformation that maps inputs to the correct labels. During this process, the outputs of the network will be compared with the correct label. The distance between the current output and the correct label will be calculated and in each training epoch the network parameters will be adjusted through gradient descent that aims to decrease the distance.

C. Supervised Learning Method based on TN Method

Supervised learning based on MPS method has been proposed in [13] a long-chain TT with 196 neurons is used as the NN architecture to classify the handwritten numbers from MNIST dataset. Each image from the MNIST dataset is associated with a label indicating what digit it is. First, the images are compressed from 784 pixels to 196 pixels, and then each pixel value x is mapped by

$$f(x) = (\sin\frac{\pi}{2}x, \cos\frac{\pi}{2}x) \tag{1}$$

to vectors. The network is constructed by 196 tensors in series connection as shown in "Fig. 6". Each neuron has one edge receiving the input from the corresponding pixel. The training

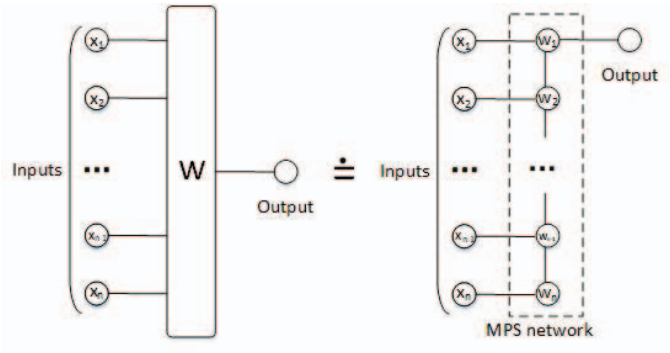

Fig. 6. MPS decomposition for weight layer, where the left is the artificial neural network and the right is the MPS framework.

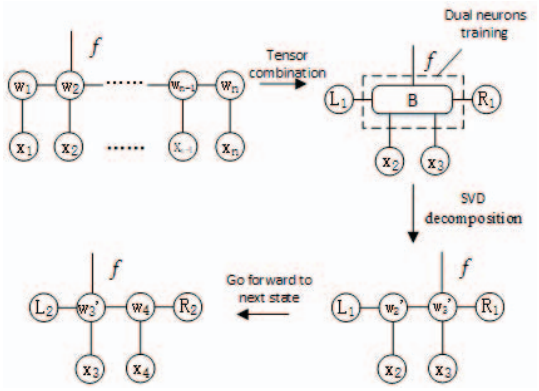

Fig. 7. The steps for training. Labels represented as f can be placed on an arbitrary neuron in TN.

algorithm can be divided into four steps, see "Fig. 7". Suppose the output edge is associate with the second neuron from the

left. First, the second and third neurons are contracted to form a bonded neuron B^l. The neurons on the left and right of the bonded neuron are contracted to form the left and right inputs L_1 and R_1 to B^l. To be more precise, R_i and L_i are calculated as

$$R_i = \sum_{\{m\}} \sum_{\{s\}} (w_{s_{i+2}}^{m_{i+1} m_{i+2}} x^{s_{i+2}}) \cdots (w_{s_N}^{m_{N-1}} x^{s_N}) \quad (2)$$

$$L_i = \sum_{\{m\}} \sum_{\{s\}} (w_{s_1}^{m_1} x^{s_1}) \cdots (w_{s_{i-1}}^{m_{i-1} m_{i-2}} x^{s_{i-1}}) \quad (3)$$

Here x^{s_j} denotes the data input at the j-th location. The training is based on gradient decent. The cost function is defined as

$$C = \frac{1}{2} \sum_{n=1}^{N} \sum_{l=1}^{L} (f - l)^2 \quad (4)$$

In this equation, N is the total number of training samples, L is the number of categories of the samples, f is the output of the TT network. Note that categorical encoding of the label is invoked here and so l takes the value either 0 or 1.

The gradient is calculated as

$$\triangle B = -\frac{\partial C}{\partial B} \quad (5)$$

$$= \sum_{n=1}^{N} \sum_{l=1}^{L} (l - f) \frac{\partial f}{\partial B} \quad (6)$$

$$= \sum_{n=1}^{N} \sum_{l=1}^{L} (l - f) \tilde{\Phi} \quad (7)$$

$\tilde{\Phi}$ is the contraction of L_{j-1}, R_{j-1}, x_j, x_{j+1}. The process of update is outlined in "Fig. 8". In the last step, an appropriate learning rate α is chosen for the the update of B^l. After the

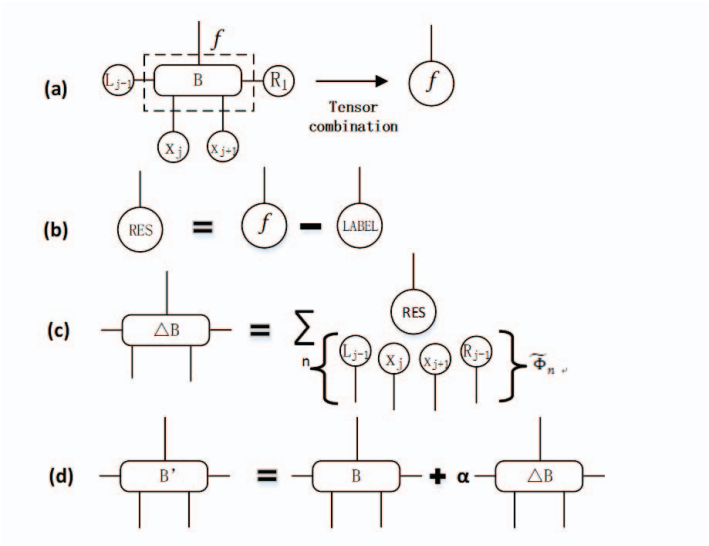

Fig. 8. Graphical expression of updating B^l. The L and R neurons are left contraction and right contraction, respectively.

update, B^l should be decomposed into two nodes in order to keep TT network in the original form. Here SVD is employed and B^l are decomposed into three neurons. Then the middle and the right neurons are contracted to form a new neuron W'_{j+1} which replaces W_{j+1}. W_j is replaced by the left node W'_j. This procedure is repeated until the end of the chain has been reached, which finishes one sweeping. The next sweeping then starts from the rightmost neurons.

III. Short-chian Method

The long-chain TT network has two major drawbacks:

- The training process is difficult to converge. The gradient is either too big or too small due to the large number of contractions.
- SGD does not perform well for long-chain TT and so all training samples must be used in the calculation of gradient per update.

In this section, we circumvent these issues by reducing the length of TT in the network. Two different proposals are discussed as follows.

A. Each Neuron Covers More Than One Pixel

In principle, each neuron in the TT could associate with more than one pixel from the image. This can be easily done by increasing the bond of the edge associated with the input. Here bond refers to upper bound on the edge index. For example, $\{x^1, \cdots, x^j\}$ can be grouped together as a single input to the first neuron with bond strength 14. By this way we can shorten the length of chain from 196 to 28, i.e. every neuron receives the input from seven pixels simultaneously. After this reduction, SGD becomes applicable. In each update

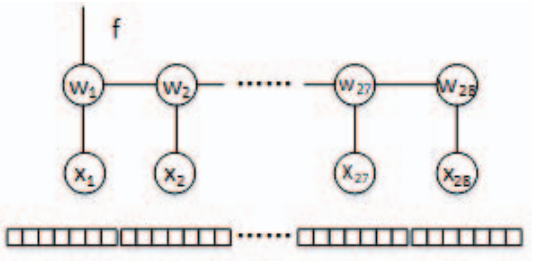

Fig. 9. Each neuron covers more than one pixel by increasing the input bond dimension.

only a batch of samples will be used to calculate the gradient, which greatly accelerates the training speed and at the same time enhances the performance of the optimization algorithm.

B. Short-chains Combined with Linear Transformation Layer

TT can be combined with other type of component. Here we combine the TT layers with a linear transformation layer. The structure of this network is shown in "Fig. 10". The entire network is composed of 14 TT components which means the image is divided into 14 parts. Each TT covers 14 pixels with 14 neurons. The output of each TT layer Y_i is sent to the linear transformation layer, i.e. the final output is a linear transformation of the outputs of the TT layers. The gradients

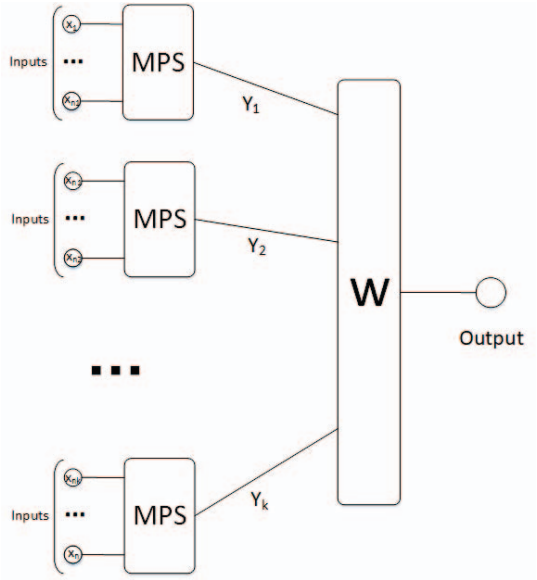

Fig. 10. Short-chains combined with linear transformation layer.

on the TT component can be calculated by back propagation (BP) as

$$\triangle B_k = -\frac{\partial C}{\partial Y_k}\frac{\partial Y_k}{\partial B_k} \tag{8}$$

$$= W_k \sum_{n=1}^{N_k}\sum_{l=1}^{L}(l-f)\frac{\partial Y_k}{\partial B_k} \tag{9}$$

$$= W_k \sum_{n=1}^{N_k}\sum_{l=1}^{L}(l-f)\tilde{\Phi}_k \tag{10}$$

Here N_k is the number of neurons in each TT component. W_k is the weighted array that defines the linear transformation layer. Similarly, the gradient of W_k can be calculated by

$$\triangle W = -\frac{\partial C}{\partial W} \tag{11}$$

$$= \sum_{n=1}^{N}\sum_{l=1}^{L}(l-f)\tilde{\Phi}_{y_k} \tag{12}$$

where $\tilde{\Phi}_{y_k}$ is the contraction of Y_k.

The training process is a bit different from single TT. In each step, two neurons are updated in each TT component and then the linear transformation layer is optimized. Each TT component has the same number of neurons so that they will finish the sweeping at the same time.

IV. Numerical Result

The two short-chain methods are tested on MNIST dataset. Here we use the same data mapping as in "(1)". The size of train set and test set are 60000 and 10000, respectively.

A. Each Neuron Covers More Than One Pixel

The numerical result is shown in "Fig. 11". Each iteration includes two sweeping, namely, left to right and then right to left. The weights of each tensor are updated four times in one iteration. As we can see, bond strength between the neurons is a crucial factor. When the bond strength is 10, the error rate is above 0.2 and the classification accuracy on the test set is only around 80%. This may be explained by the unbalanced bonds for the data input and nearby neurons. In this case the bond of the input edge is 14 but the bond strength between the neurons is only 10 which cannot forward all of features through the chain. When we improve the bond strength, the accuracy could be significantly improved. We achieve 9% and 4.4% error rates for the bond strength being 20 and 50.

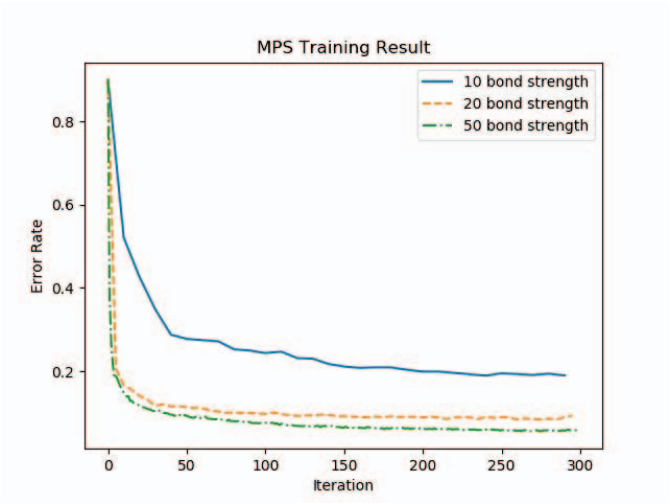

Fig. 11. Bond strength vs number of iterations in training of STN.

B. Short-chains Combined with Linear Transformation Layer

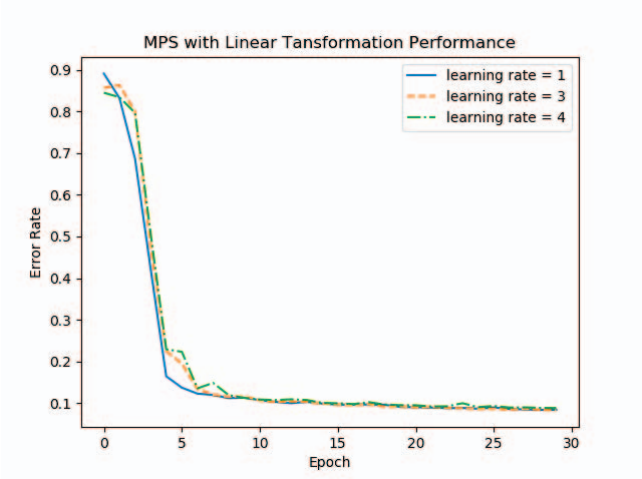

Fig. 12. The results of short-chains combined with linear transformation layer for different learning rate.

The numerical result is shown in "Fig. 12". In each epoch we update the weights twice by gradient decent. As can be seen in "Fig. 12", the best learning rate is 4 and with 80 sweepings we achieve an error rate of 7.24% using bond

strength 10. Compared to the method in the last subsection, this one uses lower bond strength and yet achieves better performance.

C. Compared to CNN

TABLE I
COMPARISON BETWEEN SHORT-CHAIN TN AND CNN

Type of neural network	*LeNet-5*	*STN(50)*	*STN(10) with LT*
Trainable parameters	59968	924000	40980
Connection numbers	330068	72892	21818
accuracy	0.981	0.956	0.9276

We consider the convolutional neural network proposed by LeCun [1] called LeNet-5. As shown in "Table. I", LeNet-5 has the maximum number of connections and short-chain TN (STN) with bond strength 50 has the maximum number of trainable parameters. Although the accuracy of STN with linear transformation layer is lower than other two networks, it has the minimum number of trainable parameters and connections. In STN training, one sweeping incurs two updates of each weight parameter. CNN takes two epochs to update each weight parameter twice. The performance of STN and CNN are compared on this basis and the result is shown in "Fig. 13" and "Fig. 14".

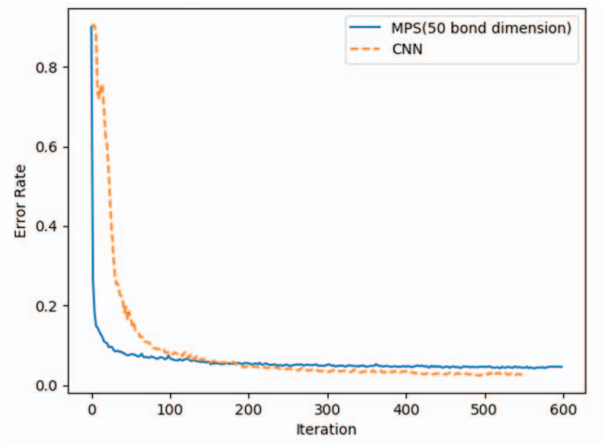

Fig. 13. Error rate comparison between STN(50) and LeNet-5.

Both STN(50) and LeNet-5 are trained with stochastic gradient descent. Interestingly, the result from "Fig. 12" shows that STN(50) is faster in converging. "Fig. 13" compares STN(10) with a linear transformation layer and CNN. Similarly, after 15 epochs the training process is completed for STN.

V. CONCLUSION

In this paper, we shorten the length of TT to improve the robustness and convergence property of supervised learning based on TN method. In contrast to long-chain case, the stochastic gradient descent algorithm can be used in the training of short-chain TN which greatly improves the performance. We test our proposals on the MNIST dataset and achieve over 95% classification accuracy. We compare the numerical results with LeNet-5 and find that short-chain TN converges faster in training.

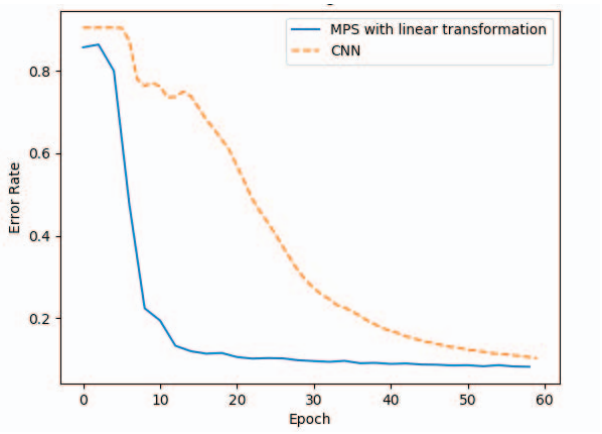

Fig. 14. Comparison between STN(10) with LT and LeNet-5 for 60 Epochs.

REFERENCES

[1] Y. LeCun, Y. Bengio, "Convolutional Networks for Images, Speech, and Time-Series," Handbook of Brain Theory & Neural Networks,1995.

[2] C. Szegedy, W. Liu, Y. Jia, et al, "Going deeper with convolutions," ArXiv:1409.4842v1[cs.CV] 17 Sep 2014.

[3] T. Mikolov, M. Karafit, L. Burget, J. Cernock, S. Khudanpur, "Recurrent neural network based language model," Interspeech, Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September, 2010 :1045-1048.

[4] G. Pilania, C. Wang, X. Jiang, S. Rajasekaran, and R. Ramprasad, "Accelerating materials property predictions using machine learning," Scientic Reports 3, 2810 EP (2013).

[5] Y. Saad, D. Gao, T. Ngo, S. Bobbitt, J.R. Chelikowsky, and W. Andreoni, "Data mining for materials: Computational experiments with AB compounds," Phys. Rev. B 85, 104104 (2012).

[6] B. Karlk, "Machine Learning Algorithms for Characterization of EMG Signals," The 6th International Conference on Computer Research and Development (2014).

[7] G. Eason, B. Noble, and I.N. Sneddon, "Hand-waving and Interpretive Dance:An Introductory Course on Tensor Networks," Jacob C. Bridgeman, Christopher T. Chubb, arXiv:1603.03039v4 [quant-ph] 16 May 2017.

[8] U. Schollwock, "The density-matrix renormalizationgroup in the age of matrix product states," Annals of Physics 326, 96-192 (2011).

[9] G. Evenbly and G. Vidal, "Tensor network states and geometry," Journal of Statistical Physics 145, 891-918 (2011).

[10] J.A. Bengua, H.N. Phien, and H.D. Tuan, "Optimal feature extraction and classification of tensors via matrix product state decomposition," 2015 IEEE Intl. Congress on Big Data (BigData Congress) (2015) pp.669-672.

[11] A. Novikov, D. Podoprikhin, A. Osokin, et al, "Tensorizing neural networks," arxiv:1509.06569, (2015).

[12] Z. Han, J. Wang, H. Fan, L. Wang, P. Zhang, "Unsupervised Generative Modeling Using Matrix Product States," arxiv:1709.01662, (2017).

[13] E. M. Stoudenmire and D. J. Schwab, "Supervised Learning with Quantum-Inspired Tensor Networks," Advances in Neural Information Processing Systems 29, 4799 (2016), May 2017.

[14] A. Khan, B. Baharudin, L. H. Lee, and K. Khan, "A Review of Machine Learning Algorithms for Text-Documents Classification," Journal of Advances in Information Technology, vol. 1, no. 1, pp. 420, 2010.