Residual Matrix Product State for Machine Learning

Ye-Ming Meng[1,2], Jing Zhang[3] Peng Zhang[3] Chao Gao[2⋆] and Shi-Ju Ran[4†]

1 State Key Laboratory of Precision Spectroscopy, East China Normal University, Shanghai 200062, China
2 Department of Physics, Zhejiang Normal University, Jinhua, 321004, China
3 School of Computer Science and Technology, Tianjin University, Tianjin, China
4 Department of Physics, Capital Normal University, Beijing 100048, China
⋆ gaochao@zjnu.edu.cn † sjran@cnu.edu.cn

September 9, 2022

Abstract

Tensor network, which originates from quantum physics, is emerging as an efficient tool for classical and quantum machine learning. Nevertheless, there still exists a considerable accuracy gap between tensor network and the sophisticated neural network models for classical machine learning. In this work, we combine the ideas of matrix product state (MPS), the simplest tensor network structure, and residual neural network and propose the residual matrix product state (ResMPS). The ResMPS can be treated as a network where its layers map the "hidden" features to the outputs (e.g., classifications), and the variational parameters of the layers are the functions of the features of the samples (e.g., pixels of images). This is different from neural network, where the layers map feed-forwardly the features to the output. The ResMPS can equip with the nonlinear activations and dropout layers, and outperforms the state-of-the-art tensor network models in terms of efficiency, stability, and representation power. Besides, ResMPS is interpretable from the perspective of polynomial expansion, where the factorization and exponential machines naturally emerge. Our work contributes to connecting and hybridizing neural and tensor networks, which is crucial to further enhance our understanding of the working mechanisms and improve the performance of both models.

Contents

1 Introduction

The tensor network (TN), as a mathematical model widely used to describe quantum many-body states [1–4], has been successfully applied to machine learning (ML). For instance, TN is used in supervised and unsupervised image classification, natural language processing, etc. [5–11]. Several recent works also demonstrate TN's ability of establishing the connection between physics and artificial intelligence [12, 13]. Nevertheless, despite the high interpretability of TN [14–16], there still exists a considerable performance gap between TN and neural network (NN) [7, 17].

TN itself represents a linear map between quantum states. While in machine learning, TN realizes a non-linear map from the features to the outputs, where there exists a local kernel function [5] that maps the features of the samples to the quantum states in Hilbert space. It is still an open issue to determine whether the NN techniques can enhance TN performance. Several recent works have explored different ways to combine TN and NN, which includes adopting the convolutional neural network (CNN) as a feature extractor in TN [7, 17, 18]; compressing the linear layers of deep NN by matrix product operators [19]; and implementing the convolutional operations using TN [20], etc. These attempts further motivate us to investigate the possible hybridization of TN and NN.

In this work, we incorporate the information highways (also known as shortcuts) [21,22], non-linear activations, and dropout [23] into TN (MPS in specific), and propose Residual MPS (ResMPS in short). The essential underlying idea of ResMPS is a delicate way of inputting data such that the variational parameters of the network layers are the functions of the data features. Such idea is inspired by the traditional feed-forward neural network (FNN), while in FNN, the data is input only in the initial step.

We provide two specific examples of ResMPS dubbed as simple and activated ResMPS. The simple version (sResMPS in short) is a multi-linear model that can exactly be written into a standard MPS, and the activated version (aResMPS in short) is a non-linear model equipped with NN layers. The results on Fashion-MNIST show that the simple ResMPS achieves the same accuracy as MPS while its parameter complexity is half of the MPS. For the activated ResMPS, we find a significant enhancement of efficiency and accuracy by introducing the non-linear activations and the dropout layers on the residual terms.

Furthermore, we determine the model interpretability of sResMPS by polynomial expression. The truncated model achieves a high level of accuracy with only a few low-order terms of sResMPS. Surprisingly, the factorization [24] and exponential machines [25] have naturally emerged in this expansion scheme. ResMPS shows the underlying connections between TN and NN for ML and can shed light on novel possibilities and flexibility for developing powerful ML models beyond NN or TN.

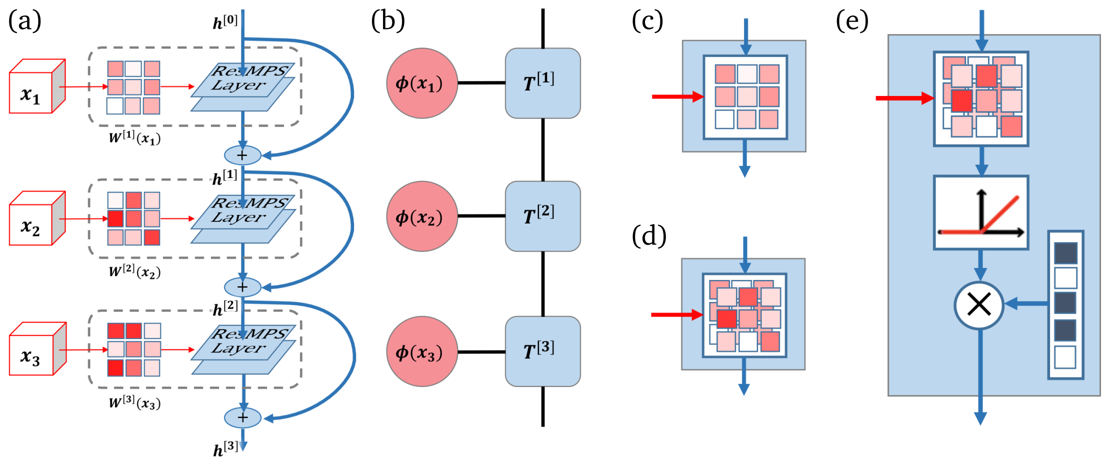

Figure 1: Illustrations of a typical ResMPS compared with a standard MPS. (a) An illustration of ResMPS containing a three-layer FNN in which the variational parameters are functions of the features, **x**. (b) An illustration of a three-tensor MPS, which is contracted with the feature vectors [see Eq. (13)]. (c) An illustration of sResMPS, which is only parameterized by a single channel weight matrix. (d) An illustration of ResMPS, which is equivalent to the standard MPS. (e) An illustration of aResMPS, where the hidden feature will pass through a two-channel linear layer, ReLU activation, and dropout layer in sequence.

The remainder of this paper is organized as follows. We define ResMPS in Sec. 2.1 and decipher its machanism in Sec. 2.2. We present two typical ResMPS architecture in Sec. 2.3. The image recognition benchmarks compared with other networks are exhibited in 2.4. Relation to recurrent neural network (RNN) and transformer is discussed in Sec. 2.5. The contribution of residual terms to training stability is discussed in Sec. 3.1. We establish equivalence between truncated ResMPS and factorization machine in Sec. 3.2. We conclude the result in Sec. 4.

2 Residual matrix product state

2.1 Definition of residual matrix product state

The traditional FNN, including the residual neural network, consists of multiple trainable layers [26]. For instance, in supervised learning, FNN maps the input sample $\mathbf{x}$ to the output l, e.g., the corresponding classification. The typical form of one layer can be written as

$$\mathbf{h}^{[n]} = \sigma\left(F^{[n]}\left(\mathbf{h}^{[n-1]}; \mathbf{W}^{[n]}\right) + \mathbf{b}^{[n]}\right), \tag{1}$$

where $\mathbf{h}^{[n-1]}$ denotes the hidden variables that are input to the n-th layer with $\mathbf{h}^{[0]} = \mathbf{x}$, $F^{[n]}$ denotes the mapping of the n-th layer (e.g., fully connected, convolution, or pooling layer). Each layer may consist some variational parameters $\mathbf{W}^{[n]}$ (weights) and $\mathbf{b}$ (bias). Furthermore, σ denotes the activation function.

Inspired by the matrix product state [27, 28] and residual neural network [21, 22], here we propose a novel machine learning architecture dubbed as residual matrix product state (ResMPS). Different from FNN [see, Eq. (1)], ResMPS does not explicitly map the features with a feed-forward network. Instead, it uses the features to parameterize FNN variational parameters. This enables an FNN to map the hidden features to the expected outputs (see, Fig. 1). In the ResMPS, the mapping of one layer is

$$\mathbf{h}^{[n]} = \mathbf{h}^{[n-1]} + \mathbf{v}^{[n]}\left(\mathbf{h}^{[n-1]}; \mathbf{W}^{[n]}(x_n), \mathbf{b}^{[n]}\right), \tag{2}$$

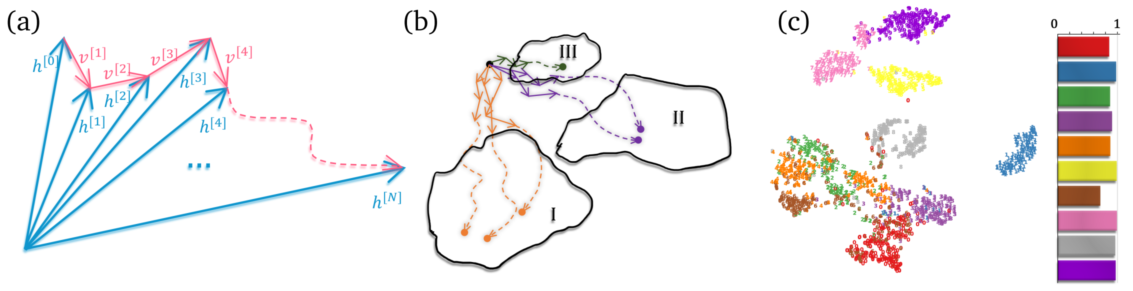

Figure 2: Encoding process of the ResMPS. (a) An illustration of a high-dimensional path of one sample. Blue arrows represent hidden features between different layers. Red arrows represent shift-vectors contributed by the residual part. (b) An illustration of the aggregation behavior of samples. The same color denotes samples belonging to the same class. (c) The two-dimensional data distribution generated by t-SNE on the endpoint dimension reduction of (b), the data points come from the Fashion-MNIST data set, and the corresponding accuracy is on the right. Note we reduce the feature dimension to 2 for illustration, which is far less than the original dimension of the hidden and weakened separations of the samples from different classes.

where the second term $\mathbf{v}^{[n]}$ is the function of $\mathbf{h}^{[n]}$ parametrized by weights $\mathbf{W}^{[n]}$ and bias $\mathbf{b}^{[n]}$, and weights $\mathbf{W}^{[n]}$ are further parameterized by the n-th feature x_n, which distinguish ResMPS from FNNs; the initial hidden $\mathbf{h}^{[0]}$ is initialized by ones for simplicity. Therefore, the depth of ResMPS depends on the input size. Eq. (2) gives the general form of $\mathbf{v}^{[n]}$, in this work we limit $\mathbf{v}^{[n]}$ in a simple form,

$$\mathbf{v}^{[n]} = \sigma\left(L^{[n]}\left(\mathbf{h}^{[n-1]}; \mathbf{W}^{[n]}(x_n)\right) + \mathbf{b}^{[n]}\right). \tag{3}$$

where $L^{[n]}$ is a linear map, and σ is the activation. Similar to ResNet, the output of one layer is the addition of the output of $\mathbf{v}^{[n]}$, and the input includes the hidden features. This is to form a shortcut of the information flow, which can avoid the vanishing/explosion of the gradients (see Sec. 3.1 for details). We further note that one obtains a standard FNN by adopting $\mathbf{h}^{[0]} = \mathbf{x}$ and removing the dependence of $\mathbf{W}$ on $\mathbf{x}$.

Note that the dimension of the hidden variable is equal to the dimension of the label index (i.e., the number of classes in the discriminative task). Therefore, one additional linear layer without bias and activation should be added to map the final hidden $\mathbf{h}^{[N]}$ to the output $f^{(l)}$ as $f^{(l)} = \sum_i L_{li} h_i^{[N]}$ where $\dim(l)$ equals to the number of classes. The linear map can be flexibly replaced by any other map as long as its output dimension matches the dimension of the label.

2.2 The working mechanism of ResMPS

We illustrate the path of the hidden state $\mathbf{h}^{[i]}$ of ResMPS in the high-dimensional vector space [as shown in Fig.2(a)]. Each layer of the ResMPS updates the state $\mathbf{h}^{[i]}$ once to make it one step forward with shift-vector $\mathbf{v}^{[i+1]} = \mathbf{h}^{[i+1]} - \mathbf{h}^{[i]}$. After passing through all layers, all shift-vectors are connected into a continuous path, namely $\sum_{i=1}^{N} \mathbf{v}^{[i]}$. For the same ResMPS, different features of the samples share the same initial point (i.e., $\mathbf{h}^{[0]}$). Since the parameter W of shift-vector $\mathbf{v}$ is a function of feature $\mathbf{x}$, the path encodes the information of samples. Besides, Similar samples have close paths in the vector space [as shown in Fig.2(b)]. After training convergence, samples of the same category will eventually gather together.

In order to show the behavior of paths of the hidden variables in the high-dimensional space, we use the Fashion-MNIST dataset to train sResMPS, and use the t-SNE algorithm [29,

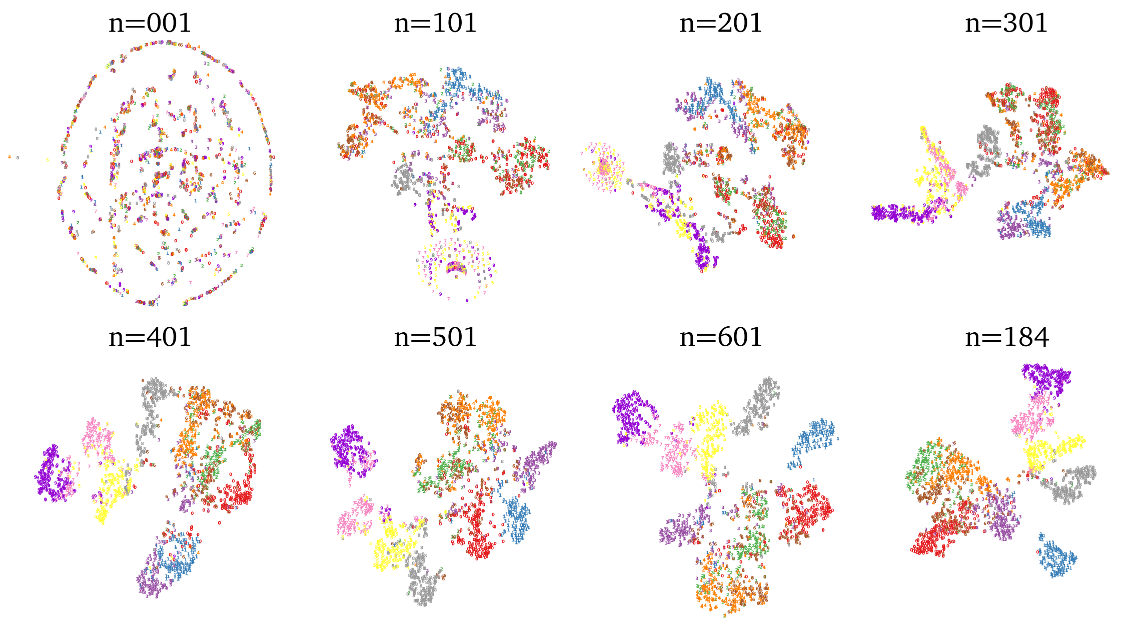

Figure 3: Intermediate hidden features $\mathbf{h}^{[n]}(1 \leq n \leq 784)$ visualized by t-SNE. It is shown that samples of the same classes gradually enter the same regions. This characterizes the encoding process of ResMPS.

30] to embed the endpoints of the ResMPS to a two-dimensional plane after the network converges. Note that before we apply t-SNE for dimensionality reduction, the original virtual feature has 100 components, which is sufficient large for accuracy yet economical. Fig. 2(c) illustrates the visualization of final hidden features $h_i^{[N]}$ in the two-dimensional space. Samples with better classification accuracy are relatively separated, while those with poor classification accuracy overlap with other classifications. We also demonstrate the visualizations by t-SNE for the intermediate hidden features in Fig. 3 to show how the hidden features gradually cluster into different sub-regions. It shows that the initial hidden features are uniformly distributed, and then gradually separated during the encoding process.

2.3 The Architecture of ResMPS

In the following, we examine two instances of ResMPS, called simple ResMPS [sResMPS, see Fig.1(c)] and activated ResMPS [aResMPS, see Fig.1(f)]. The sResMPS is a multi-linear model that is equivalent to MPS. It achieves the same accuracy with only half of the parameter complexity of the MPS. The aResMPS is a generalized version of sResMPS, in which the generalization efficiency is enhanced by introducing non-linear activation functions and dropout in the FNN part. The map of one layer in the sResMPS is written as

$$h_j^{[n]} = h_j^{[n-1]} + \sum_i x_n W_{ij}^{[n]} h_i^{[n-1]}. \tag{4}$$

The weights of the layers in the FNN are linearly dependent on the features $\mathbf{x}$. The bias terms are also disabled in this example.

sResMPS is equivalent to a restricted version of MPS and can achieve identical performance with only a half parameter complexity of standard MPS. See Sec. 2.4.2 for details.

It is seen that MPS has a remarkable representation power. The training error is less than 1% [31]. However, a gap between the training and testing accuracy suggests an over-fitting issue. To address the over-fitting issue, we propose the aResMPS by incorporating the non-linear activation functions and dropout. This modification also enhances the generalization power [32]. The map of each layer in the FNN of the aResMPS is more-or-less a fully-connected layer with a shortcut, which reads

$$\mathbf{h}^{[n]} = \mathbf{h}^{[n-1]} + \sigma\left(L^{[n]}(\mathbf{h}^{[n-1]}) + \mathbf{b}^{[n]}\right), \tag{5}$$

where σ is an activation function. The map $L^{[n]}$ relies on the feature x_n in a non-linear fashion

$$L^{[n]}(\mathbf{h}^{[n-1]})_j = \sum_{c=1,2}\left[\xi^{[c]}(x_n)\sum_i W_{ij}^{[n,c]} h_i^{[n-1]}\right]. \tag{6}$$

The architecture of ResMPS is flexible, due to the choice of $\xi^{[c]}(x_n)$ and the number of channels dim(c). Here we choose $\xi^{[1]}(x_n) = x_n$ and $\xi^{[2]}(x_n) = 1 - x_n$. We introduce $\xi^{[c]}$ to enhance the non-linearity of the aResMPS. It is worth mentioning that even sResMPS represents a non-linear map on the features $\mathbf{x}$ (but a linear map on the hidden features).

For the aResMPS, the map on either the features or the hidden features is non-linear. Indeed, the FNN embedded inside the aResMPS can be replaced by any NN. Here, we choose a standard fully-connected network with two channels labeled by c.

Throughout this paper, we choose the ReLU activation function that can screen the negative inputs [33, 34]. Due to its piecewise linear characteristics, the gradient can directly pass through without attenuation or enhancement. Therefore, the ReLU function is suitable for enhancing the non-linearity of the deep networks, which can improve its representation power and avoid the vanishing/explosion of the gradient. Furthermore, we use a dropout layer combined with the residual structure to improve the generalization ability of ResMPS. The dropout layer effectively creates an ensemble of networks while avoiding the co-adaptation of intermediate variables [23, 35, 36]. We impose dropout on the residual terms, i.e. $\mathbf{h}^{[n]} = \mathbf{h}^{[n-1]} + \text{dropout}(\sigma(\cdots))$.

If we discard the activation and the dropout layers of aResMPS [see Fig.1(e)], we will get a standard two-channel MPS. For a standard MPS with physical bond dimension $d = 2$, the map given by a local-tensor construction is [31]

$$h_j^{[n]} = \sum_{c=1,2}\left[\xi^{[c]}(x_n)\sum_i T_{ij}^{[n,c]} h_i^{[n-1]}\right]. \tag{7}$$

If we introduce transformation $T_{ij}^{[n,c]} = W_{ij}^{[n,c]} + \delta_{ij}$, we can simply get

$$h_j^{[n]} = h_j^{[n-1]}\left(\sum_{c=1,2}\xi^{[c]}(x_n)\right) + \sum_{c=1,2}\left[\xi^{[c]}(x_n)\sum_i W_{ij}^{[n,c]} h_i^{[n-1]}\right]. \tag{8}$$

By taking feature map with norm-1 normalization [31], i.e. $\sum_{c=1,2}\xi^{[c]}(x_n) = 1$, we get a ResMPS with map

$$h_j^{[n]} = h_j^{[n-1]} + \sum_{c=1,2}\left[\xi^{[c]}(x_n)\sum_i W_{ij}^{[n,c]} h_i^{[n-1]}\right]. \tag{9}$$

2.4 Benchmarking results

2.4.1 Classification accuracy

For the MNIST and Fashion-MNIST datasets, Table 1 shows the accuracy of the sResMPS and aResMPS, compared with several established NN [39] and TN models [7, 9, 15, 17, 31, 38].

Table 1: Experimental results on MNIST [37] and Fashion-MNIST dataset. The first 6 models are pure TN architectures, which means they are multi-linear, and no neural structures like pooling, activation and convolution are introduced. AlexNet, ResNet, and CNN-PEPS are NN or TN-NN hybrid models. For aResMPS, we use ReLU as activation, and the dropout probability is set to be 0.6.

Model	MPS Train	MPS Test	Fashion-MNIST Train	Fashion-MNIST Test
MPS machine [31]	1.0000	0.9880	0.9988	0.8970
Unitary tree TN [9]	0.98	0.95	-	-
Tree curtain model [38]	-	-	0.9538	0.8897
Bayesian TN [15]	-	-	0.8950	0.8692
EPS-SBS [7]	-	0.9885	-	0.886
PEPS [17]	-	-	-	0.883
CNN-PEPS [17]	-	-	-	0.912
AlexNet [39]	-	-	-	0.8882
ResNet [39]	-	-	-	0.9339
sResMPS	1.0000	0.9873	0.9987	0.8909
aResMPS	1.0000	0.9907	0.9999	0.9142

The MPS and ResMPS models represent a high level of representation power, as indicated by their high training accuracy. The aResMPS also surpasses the probabilistically interpretable Bayesian [15] and other TN models, including the two-dimensional TN known as projected-entangled pair state (PEPS) [17]. It also achieves a (slightly) better accuracy than that of CNN-PEPS model, in which CNN is adopted as the feature extractor. This accuracy surpasses the CNN without the stacking architecture, such as AlexNet [39]. The aResMPS still does not overperform the ResNet which is formed by stacking multiple convolution layers. It seems that the ResMPS models eventually surpass ResNet by replacing the fully-connected network with more sophisticated ones or staking multiple ResMPSs.

2.4.2 Redundancy of regular MPS

To see the equivalence between the standard MPS and sResMPS mentioned in Sec.2.3, we introduce the third-order tensors $\mathbf{T}^{[n]}$ satisfying

$$T^{[n]}_{1,:,:} = \mathbf{I}, \quad T^{[n]}_{2,:,:} = \mathbf{W}^{[n]}. \tag{10}$$

The feature vectors $\phi(x_n)$ are obtained by the feature map as $\phi(x_n) = (1, x_n)$ [5, 9, 31]. Summing their joint index gives a linear mapping represented by a matrix

$$\begin{aligned} A^{[n]}_{a_{n-1},a_n} =& \sum_{p_n} T^{[n]}_{p_n,a_{n-1},a_n} \phi(x_{n-1})_{p_n} \\ =& \delta_{a_{n-1},a_n} + x_n W^{[n]}_{a_{n-1},a_n}. \end{aligned} \tag{11}$$

Applying this mapping to $\mathbf{h}^{[n-1]}$, one gets

$$\begin{aligned} h^{[n]} =& \sum_{a_{n-1}} h^{[n-1]}_{a_{n-1}} A^{[n]}_{a_{n-1},a_n} \\ =& h^{[n-1]}_{a_n} + \sum_{a_{n-1}} h^{[n-1]}_{a_{n-1}} x_n W^{[n]}_{a_{n-1},a_n}. \end{aligned} \tag{12}$$

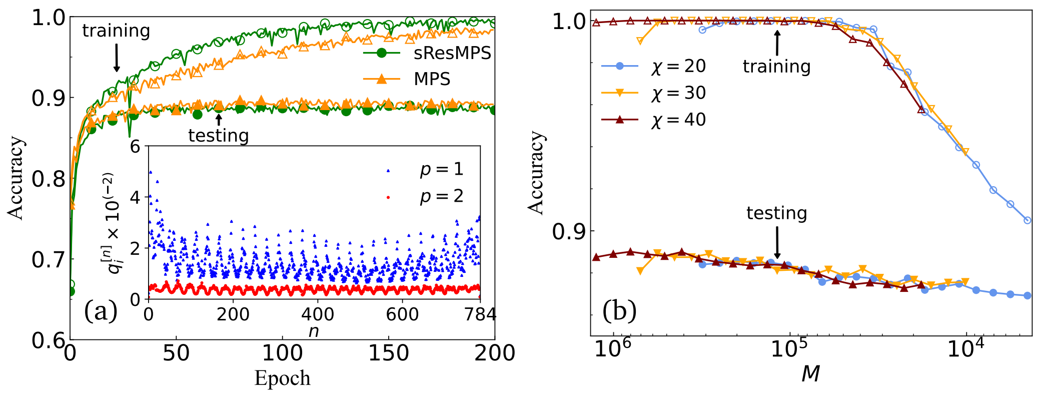

Figure 4: Numerical results of the simple ResMPS. (a) Training and testing accuracy of sResMPS (without dropout) and MPS versus epochs on the Fashion-MNIST dataset. The inset shows the average norm [Eq. (14)] of the two channels in the MPS for different tensors n. (b) Training and testing accuracy of the sResMPS versus the total number of unmasked weights in the sResMPS. The left end of each curve corresponds to the un-pruned result. It is also seen that the first few steps of pruning improve the accuracy. The total number of parameters of sResMPS with $\chi = 20, 30$, and 40 equals to about 3×10^5, 7×10^5 and 13×10^5, respectively.

This form is exactly the definition of sResMPS, shown in Eq. (4). Therefore, the sResMPS is equivalent to the standard MPS formed by the following tensors

$$\mathcal{T} = \sum_{\{a_n\}} \prod_n T^{[n]}_{p_n a_n a_{n+1}} \tag{13}$$

as its tensor-train cores [40] [Fig. 1 (b)]. The numbers of the input and output hidden features for different layers provide the two virtual bond dimensions of the MPS, i.e., $\{\dim(a_n)\}$. In this work, we fix $\dim(a_n) = \chi$, $\forall n$. The physical dimension of the MPS should also match the dimension of the feature vector, i.e. $\dim(\phi(x_n)) = \dim(p_n)$.

For $\dim(p_n) = 2$, the number of variational parameters in sResMPS is $\sim O(N\chi^2)$ where N is total number of features. This is only half of that in the MPS which is $\sim O(2N\chi^2)$. Our numerical simulations show that the accuracy of both models is almost the same. See the training and testing accuracy versus epochs on Fashion-MNIST dataset [41] in Fig. 4 (a) with $\chi = 40$. This is because one of the two channels of each tensor in the MPS is much less "activated". The inset of Fig. 4 (a) shows the average norm of the two channels of different tensors

$$q_p^{[n]} = \frac{1}{\chi^2} \sum_{j=1}^{\chi} \sum_{k=1}^{\chi} \left| T^{[n]}_{pjk} - \delta_{jk} \right|, \tag{14}$$

with $p = 1, 2$ representing the channels. The main contribution to the output is from the second channel. Therefore, one channel is sufficient to pass the information to the output.

2.4.3 Representation power of ResMPS

In physics, the virtual bond dimension, χ, characterizes the representation power of the MPS. This is because it determines the total number of variational parameters and the upper bound of the entanglement entropy the MPS can carry [1]. This scenario may not be the case for machine learning. We show this by adding masks on the variational parameters, i.e., pruning [14]. Each parameter is multiplied by a factor that is either zero or one. The parameters multiplied by zeros are masked. To mask a certain number of parameters, we choose to mask those with relatively small absolute values. We then optimize the unmasked parameters after the masks take effect.

Fig. 4 (b) shows the accuracy values versus the number of unmasked parameters M. For different virtual bond dimensions, $\chi = 20$, 30, and 40, the results are similar if the number of the unmasked parameters are the same. This suggests that the parameter which characterizes the representation and generalization power, is in fact, M, not χ. For a given χ, it is possible to further reduce the complexity of MPS (and sResMPS) without harming the accuracy. Our results also indicate that the sResMPS achieves its maximal representation power for $M \sim 5 \times 10^4$ (the training accuracy $\simeq 99.98\%$).

2.5 Comparison with the recurrent neural network and transformer

In the first sight the ResMPS is quite similar to the RNN and transformer, which are intensively explored nowadays. Here we compare the ResMPS with the later two respectively.

The layer mapping of ResMPS is similar to recurrent neural network (RNN). However, there are two essential differences between ResMPS and RNN. Firstly, ResMPS is non-uniform, and can be reduced to the translation-invariant MPS named as uniform MPS [42, 43]. From this point of view, RNN, in which the layers share the same variational parameters, is closer to the uniform MPS. Secondly, the recurrent mapping (also called a unit) is constructed by several linear operations and non-linear activations. Different constructions of the units define different RNN, such as the long short-term memory model. For the ResMPS, it is formed by tensor units, which are simply multi-linear mappings and can be analyzed by the established methods such as tensor decompositions [2, 27, 40].

ResMPS is different from transformer networks. Firstly, the transformer network is established on the attention mechanism, which is given by the inner product of “querys” and “keys”, while ResMPS contains higher order interaction of input features (see its connection to exponential machine in Sec. 3.2 for detail). Hence, the working mechanism of ResMPS, as explained in Sec. 2.2, is different to transformer networks. Secondly, A general transformer network consists of an encoder and a decoder. The encoder transforms a data sequence into a vector, while the decoder transforms the vector into another sequence. In contrast, ResMPS, and in general the MPS-based networks output a vector. Moreover, ResMPS does not need positional encoding, see appendix B for more discussions.

3 Properties of the residual structure

3.1 Avoiding the gradient problems by residual terms

A typical MPS architecture designed for pattern recognition contains hundreds of tensor cores. Such an architecture probably encounters gradient vanishing/exploding problems. To avoid the gradient problems, some existing MPS schemes apply a DMRG-like algorithm where the MPS takes the canonical form [5,6,10,44]. In these attempts, however, the accuracy is sensitive to the hidden features’ dimensions (virtual bonds). Recently, an MPS algorithm was proposed based on the automatic gradient technique [31] that can achieve higher accuracy than the previous ones, while its performance is not sensitive to the virtual dimensions. To find out why such a deep network avoids the gradient problems, here we construct the tensor cores to satisfy a specific form given by Eq. (10). The identity in $T^{[n]}_{1,:,:}$ plays the role of “highway” to pass the information from the previous tensor core directly to the latter ones. The components $T^{[n]}_{2,:,:}$ represent the residual terms, which is $\ll O(1)$. The application of residual conditions implies that each layer of ResMPS can easily express identity mapping. In other words, the architecture

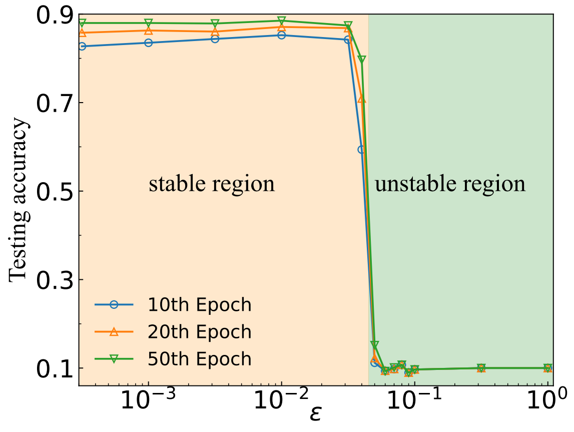

Figure 5: The testing accuracy of the sResMPS versus ε on Fashion-MNIST dataset. Here ε is the standard deviation of the initial residual part. We fix the number of epochs to be 10, 20, and 50. The network can be trained stably for small values of ε. Otherwise, the training process encounters gradient vanishing (or explosion) problems. The stable and unstable regions are illustrated by orange and green colors, respectively. Note that for the unstable region, the value of the network elements diverges. The network will give classifications randomly, thus, the accuracy tends to be 0.1.

of ResMPS satisfies the identity parameterization [21, 22, 45].

To further demonstrate the role of identity parameterization in ResMPS, we use Gaussian distributions with zero mean and standard deviation ε to randomly initialize the elements of $T^{[n]}_{2,:,:}$. Fig. 5 shows the testing accuracy at the 10-th, 20-th, and 50-th epochs. For a sufficiently small ε, the accuracy is quickly and stably converged. However, for relatively large ε [e.g., $O(10^{-1})$] as illustrated by the green region, the gradients become unstable. Consequently, the accuracy stays around 0.1 and cannot improve further through the training process. Note that this may be unstable in most cases if instead of the identity parameterization, the entire T is randomly initialized.

3.2 Relations to polynomial expansion

The forward propagation of the sResMPS (4) is fully linear on the hidden features. Applying the maps to the initial hidden features $\mathbf{h}_0$ in sequence, we can then rewrite the output hidden features in an expansive form [Fig. 6 (b)] as

$$\begin{aligned}\mathbf{h}^{[N]} &= \left(\mathbf{I}+x_N\mathbf{W}^{[N]}\right)\dots\left(\mathbf{I}+x_2\mathbf{W}^{[2]}\right)\left(\mathbf{I}+x_1\mathbf{W}^{[1]}\right)\mathbf{h}^{[\mathbf{0}]}\\ &=\sum_{k=0}^{N}\mathbf{M}^{[k]}\mathbf{h}^{[0]},\end{aligned} \tag{15}$$

where N is the total number of features $\mathbf{x}$. The output $\mathbf{h}^{[N]}$ is the stack of $N+1$ terms. The zeroth term satisfies $\mathbf{M}^{[0]}=\mathbf{I}$, which is the result of the information highway from the first input hidden features to the output. The term $\mathbf{M}^{[1]}=\sum_{\alpha=1}^{N}x_\alpha\mathbf{W}^{[\alpha]}$ is the part in ResMPS which is linear on the features $\mathbf{x}$. The k-th term contains the k-th order contributions from $\mathbf{x}$,

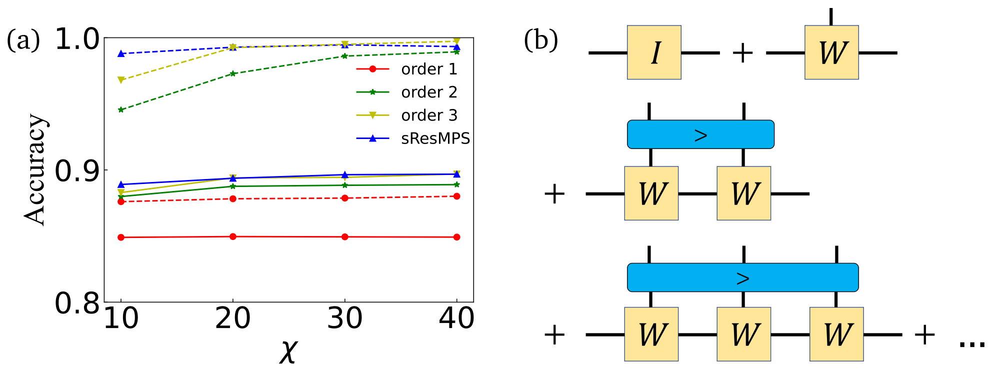

Figure 6: Polynomial expansion based on ResMPS. (a) Training (dashed) and testing (solid) accuracy versus χ by taking different orders in the expansion form; (b) An illustration of the polynomial expansion picture of the sResMPS, see Eq. (16).

i.e.,

$$\mathbf{M}^{[k]} = \sum_{\alpha_1 \ldots \alpha_k = 1}^{N} G_{\alpha_1 \ldots \alpha_k} x_{\alpha_1} \ldots x_{\alpha_k} \mathbf{W}^{[\alpha_1]} \ldots \mathbf{W}^{[\alpha_k]}, \tag{16}$$

$$G_{a_1, a_2, \ldots, a_n} = \begin{cases} 1, & a_1 > a_2 > \ldots > a_n \\ 0, & \text{otherwise.} \end{cases} \tag{17}$$

This formula is actually a specific form of the Exponential Machines [25]. Due to their essential similarity, the algebraic properties of Exponential Machines are also valid for sResMPS. For instance, the output feature $\mathbf{h}^{[N]}$ is a linear mapping concerning the initial hidden feature h_0, and a multi-linear mapping concerning the feature $\mathbf{x}$.

Due to the residual condition [see Eq. (10) with $|\mathbf{W}^{[n]}| \ll O(10^{-1})$], the contributions from the higher-order terms of Eq. (16) should decay exponentially with k. Therefore, we can define a set of lower-order effective models by retaining the first few terms. For instance, by keeping the zeroth- and first-order terms in Eq. (16), we obtain a model in which the output features are linear to both hidden and sample features. Keeping the zeroth, linear, and quadratic terms gets a model

$$\mathbf{h}^{[N](2)} = \Big(\mathbf{I} + \sum_{\alpha=1}^{N} x_\alpha \mathbf{W}^{[\alpha]} + \sum_{\alpha,\beta=1}^{N} G_{\alpha,\beta} x_\alpha x_\beta \mathbf{W}^{[\alpha]} \mathbf{W}^{[\beta]}\Big) \mathbf{h}^{[0]}. \tag{18}$$

This model is similar to Factorization Machines [24] and polynomial NN [46].

Fig. 6 (a) shows the difference between the accuracy of several lower-order models and the sResMPS. This implies that the significant improvement achieved by the sResMPS has its root in a few lower-order terms, especially the linear term. As the order increases, the cost of directly computing Eq. (15) is also exponentially increased. Therefore, truncating the order of expansion is not economical. ResMPS adopts a different and efficient scheme for retaining all higher-order interactions.

4 Conclusion

We propose ResMPS by incorporating MPS with the information highways, non-linear activations, and dropout. In contrast to FNN, the variational parameters in ResMPS are replaced by adjustable functions. For FNN, features are input at the first layer of the network. For ResMPS, however, features are divided and input into the weight matrices of each layer, which is inherited from MPS. Furthermore, the introduction of neural network structures in the ResMPS brings more vital representation power than the usual MPS. For concreteness, we present two specific versions of ResMPS.

The first derived architecture, sResMPS, is simply a linear version of ResMPS. By comparing MPS' learning performance on the Fashion-MNIST dataset, we reveal the channel redundancy of MPS. sResMPS thus discards the redundant channel. Consequently, it achieves consistent accuracy while the parameter complexity is reduced by half.

The second derived architecture, aResMPS, is equipped with activation and dropout layers. We compare aResMPS with several TN and NN models on the Fashion-MNIST dataset. The activation and dropout layer enhance the non-linearity and generalization ability of the model, respectively. Therefore, aResMPS surpasses the state-of-the-art TN methods and AlexNet in terms of accuracy, although it is still inferior to ResNet formed by stacking multiple convolution layers. Going beyond present aResMPS to achieve higher accuracy, e.g., replacing the weight matrices with convolution layers, is a valuable improvement direction of ResMPS.

The perspectives of the residual network derive the polynomial expansion of ResMPS. The benefits are two-fold. Firstly, we give the condition of vanishing/explosion of the gradients of ResMPS. This helps the feature design of MPS and ResMPS algorithms with stable convergence. Secondly, it establishes the equivalence between MPS and polynomial networks such as Factorization Machines and Exponential Machines. Further numerical evidence suggests that the contribution of high-order terms is insignificant. This helps to better understand the MPS and ResMPS.

Are other NN structures (e.g., convolution and pooling layers) compatible with ResMPS? Is it possible to propose a ResMPS structure based on general NN structures (e.g., Tree TN or Projected Entangled-Pair States)? These problems are worthy of investigation in the future.

Acknowledgements

Funding information Y.-M.M. and C.G. are supported by National Natural Science Foundation of China (NSFC, Grant No. 1183501 and No. 12074342) and Zhejiang Provincial Natural Science Foundation of China (Grant No. LR22A040001 and No. LY21A040004). S.-J.R. is supported by NSFC (Grant No. 12004266 and No. 11834014), Beijing Natural Science Foundation (No. 1192005 and No. Z180013), Foundation of Beijing Education Committees (No. KM202010028013), and the Academy for Multidisciplinary Studies, Capital Normal University. J.Z. and P.Z. are supported by NSFC (Grant No. 61772363).

A Training details

All benchmarks of this paper are implemented on MNIST and Fashion-MNIST datasets, each of which includes 60,000 training and 10,000 testing grayscale images with $L = 10$ labels. To fit inputs of ResMPS, we choose a specific path to reorder 2D pixels into a 1D sqeuence $\{(x_1, x_2, \ldots x_n; n = 784) | x_i \in [0,1]\}$, where 784 is the pixel number of one image. For single channel ResMPS, e.g., the sResMPS, there is no need to feature map the original data. And for double channel ResMPS, a linear feature map $(x, 1-x)$ is adapted to match the channel dimension. The dimension of the hidden feature χ is set to 100, which is sufficient large for accuracy yet economical. The predicted classification is given by the largest component

$$f(\mathbf{x}) = \mathrm{argmax}_l f^{(l)}(\mathbf{x}), \tag{19}$$

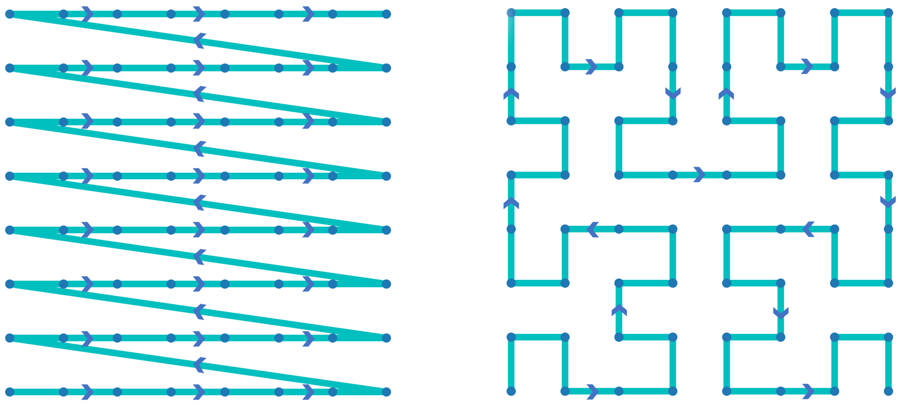

Figure 7: The zigzag path (left) and the Hilbert path (right).

where $f^{(l)}(\mathbf{x})$ is the overall mapping of ResMPS with $l = 1, \ldots L$ denotes classification. To train ResMPS, we use the Stochastic Gradient Descent (SGD) method with Adam optimizer [47] and learning rate 10^{-4}. Data are divided into mini-batches of size 1000. We choose cross-entropy as the loss function,

$$CE = -\sum_{(\mathbf{x}_i, y_i)\in\mathcal{D}} \log \mathrm{softmax} f^{(y_i)}(\mathbf{x}_i) \tag{20}$$

where

$$\mathrm{softmax} f^{(y_i)}(\mathbf{x}_i) = \frac{e^{f^{(y_i)}(\mathbf{x}_i)}}{\sum_{l=0}^{L-1} e^{f^{(y_l)}(\mathbf{x}_i)}}. \tag{21}$$

Here $\mathcal{D}$ is the data set of a mini-batch, and $\mathbf{x}_i$ and y_i are the i-th input and classification respectively. If dropout is taken, the corresponding probability is set to 0.6. We take ReLU for the non-linear case. The whole implementation is based on the PyTorch library, and the relevant code is available on GitHub.

B Path independency

ResMPS naturally deals with sequential data, but general data like images are usually high-dimensional. Therefore, one needs to choose a specific path to unfold high-dimensional data. To study how path choice effects the performance, we here compare three typical paths — zigzag path, Hilbert path, and random path. The first two are illustrated in Fig. (7). The zigzag path, as the most popular one, arranges data row by row; the Hilbert path preserves more neighboring information than others; as for the random path, the original position information is completely dropped, i.e., all points are collected together, shuffled completely, and then reorderd in a random way. Note that the numbers of rows and columns of the Hilbert path need to fit 2^n with n a positive integer, so we need to extend the original image by using borders consisting of zeros. For the case of MNIST and Fashion-MNIST, image size is extended from 28×28 to 32×32.

We run benchmarks for sResMPS without activation and dropout on the Fashion-MNIST dataset. The hidden dimension is set to be 40 and other training details are the same as appendix A. The result is shown in Fig. 8. To our surprise, there is no significant performance difference observed, even the random path still performs well.

The unexpected result seems contrary to the exponential decay behavior of entanglement entropy of MPS [48], and hints the existence of long-range information. We suppose that this inconsistency is mainly due to the introduction of residual connection, which preserve the

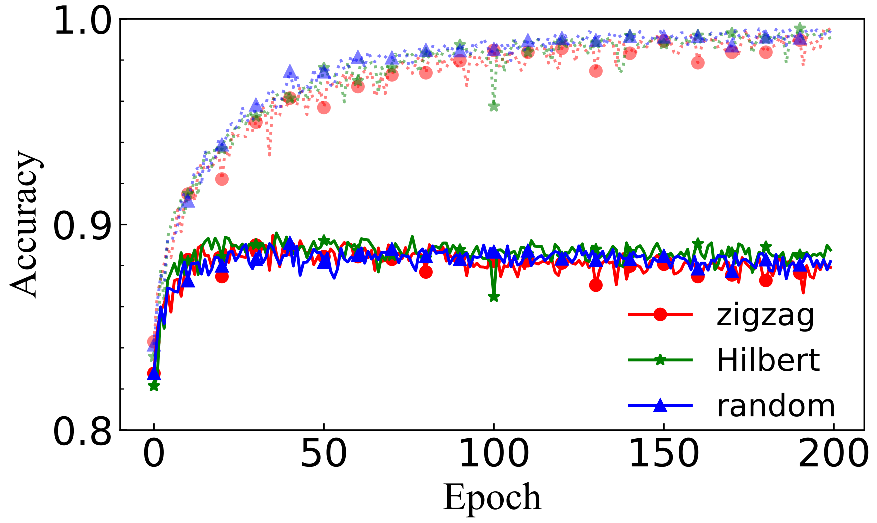

Figure 8: Convergence history of different paths on Fashion-MNIST dataset.

Table 2: Benchmarks for fine partitioned models. sResMPS and aResMPS corresponds to the first and the last row respectively. Other models filled the gap between sResMPS and aResMPS.

Model	MPS Train	MPS Test	Fashion-MNIST Train	Fashion-MNIST Test
1-channel, -ReLU, -dropout	1.0000	0.9873	0.9987	0.8909
1-channel, -ReLU, +dropout	1.0000	0.9889	0.9618	0.9022
1-channel, +ReLU, -dropout	1.0000	0.9864	1.0000	0.8957
1-channel, +ReLU, +dropout	1.0000	0.9885	0.9904	0.9108
2-channel, -ReLU, -dropout	1.0000	0.9880	0.9988	0.8970
2-channel, -ReLU, +dropout	1.0000	0.9896	0.9899	0.9081
2-channel, +ReLU, -dropout	1.0000	0.9873	1.0000	0.9102
2-channel, +ReLU, +dropout	1.0000	0.9907	0.9999	0.9142

sensitivity of gradient in the long-range, and presumably maintains the long-range correlation.

C Additional benchmarks

Note that aResMPS can achieve better performance than sResMPS, while the former differs to the later in three aspects: the channel number is doubled, the nonlinear action and dropout are added. To explore the contribution of each aspect to the final performance, we use different combinations to give more fine partitioned models. The benchmark results are shown in Table 2. From the experimental results, we can see that these components more or less improve the performance, and aResMPS as the most complex one achieves maximum performance.

The result of the pruning test shows that the hidden dimension χ does not affect the accuracy, but this statement is only valid for sufficiently large χ. If we gradually reduce the value of χ, a predictable result is that the dimensions of hidden space are too small, so that different classes become hard to distinguish. We show convergence procedure of sResMPS for small χs, see Fig. 9. The results show that increasing the bond dimension indeed helps to

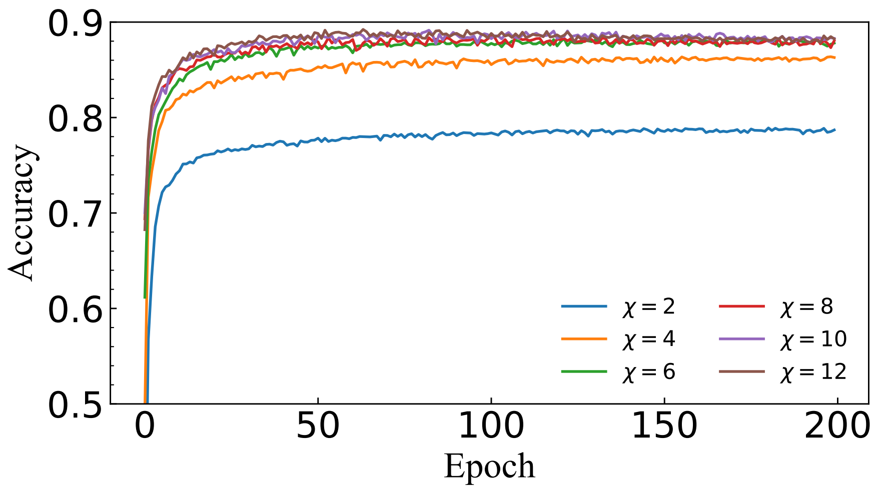

Figure 9: Testing accuracy for sResMPS on Fashion-MNIST dataset for various hidden dimension χ. The performance will increase as χ increase for small χ, and achieves saturation after a critical point about $\chi_c = 6$, which is the same as the target space dimension.

improve the accuracy, but only for a small enough χ. Once a certain threshold is exceeded, increasing χ has no benefit to accuracy growth. Another fact is that even a really small value of hidden dimension $\chi = 2$ has already achieved about 78% accuracy.

References

[1] U. Schollwöck, *The density-matrix renormalization group in the age of matrix product states*, Annals of Physics **326**(1), 96 (2011), doi:10.1016/j.aop.2010.09.012.

[2] S.-J. Ran, E. Tirrito, C. Peng, X. Chen, L. Tagliacozzo, G. Su and M. Lewenstein, *Tensor Network Contractions*, Springer International Publishing, doi:10.1007/978-3-030-34489-4 (2020).

[3] F. Verstraete, V. Murg and J. Cirac, *Matrix product states, projected entangled pair states, and variational renormalization group methods for quantum spin systems*, Advances in Physics **57**(2), 143 (2008), doi:10.1080/14789940801912366.

[4] G. Evenbly and G. Vidal, *Tensor network states and geometry*, Journal of Statistical Physics **145**(4), 891 (2011), doi:10.1007/s10955-011-0237-4.

[5] E. Stoudenmire and D. J. Schwab, *Supervised learning with tensor networks*, In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon and R. Garnett, eds., *Advances in Neural Information Processing Systems 29*, pp. 4799–4807. Curran Associates, Inc. (2016).

[6] Z.-Y. Han, J. Wang, H. Fan, L. Wang and P. Zhang, *Unsupervised generative modeling using matrix product states*, Physical Review X **8**, 031012 (2018), doi:10.1103/PhysRevX.8.031012.

[7] I. Glasser, N. Pancotti and J. I. Cirac, *From probabilistic graphical models to generalized tensor networks for supervised learning*, IEEE Access **8**, 68169 (2020), doi:10.1109/access.2020.2986279.

[8] S. Cheng, L. Wang, T. Xiang and P. Zhang, *Tree tensor networks for generative modeling*, Physical Review B **99**, 155131 (2019), doi:10.1103/PhysRevB.99.155131.

[9] D. Liu, S.-J. Ran, P. Wittek, C. Peng, R. B. García, G. Su and M. Lewenstein, *Machine learning by unitary tensor network of hierarchical tree structure*, New Journal of Physics **21**(7), 073059 (2019), doi:10.1088/1367-2630/ab31ef.

[10] Z.-Z. Sun, S.-J. Ran and G. Su, *Tangent-space gradient optimization of tensor network for machine learning*, Physical Review E **102**, 012152 (2020), doi:10.1103/PhysRevE.102.012152.

[11] P. Zhang, Z. Su, L. Zhang, B. Wang and D. Song, *A quantum many-body wave function inspired language modeling approach*, In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. ACM, doi:10.1145/3269206.3271723 (2018).

[12] J. Chen, S. Cheng, H. Xie, L. Wang and T. Xiang, *Equivalence of restricted boltzmann machines and tensor network states*, Physical Review B **97**(8), 085104 (2018), doi:10.1103/physrevb.97.085104.

[13] V. Khrulkov, A. Novikov and I. Oseledets, *Expressive power of recurrent neural networks*, In *International Conference on Learning Representations* (2018).

[14] Y. Levine, O. Sharir, N. Cohen and A. Shashua, *Quantum entanglement in deep learning architectures*, Physical Review Letters **122**, 065301 (2019), doi:10.1103/PhysRevLett.122.065301.

[15] S.-J. Ran, *Bayesian tensor network with polynomial complexity for probabilistic machine learning* (2019), 1912.12923v2.

[16] J. Martyn, G. Vidal, C. Roberts and S. Leichenauer, *Entanglement and tensor networks for supervised image classification* (2020), 2007.06082v1.

[17] S. Cheng, L. Wang and P. Zhang, *Supervised learning with projected entangled pair states*, Physical Review B **103**(12), 125117 (2021), doi:10.1103/physrevb.103.125117.

[18] D. Liu, Z. Yao and Q. Zhang, *Quantum-classical machine learning by hybrid tensor networks* (2020), 2005.09428v1.

[19] Z.-F. Gao, S. Cheng, R.-Q. He, Z. Y. Xie, H.-H. Zhao, Z.-Y. Lu and T. Xiang, *Compressing deep neural networks by matrix product operators*, Physical Review Research **2**, 023300 (2020), doi:10.1103/PhysRevResearch.2.023300.

[20] P. Blagoveschensky and A. H. Phan, *Deep convolutional tensor network* (2020), 2005.14506v1.

[21] K. He, X. Zhang, S. Ren and J. Sun, *Identity mappings in deep residual networks*, In *Computer Vision – ECCV 2016*, pp. 630–645. Springer International Publishing, doi:10.1007/978-3-319-46493-0_38 (2016).

[22] K. He, X. Zhang, S. Ren and J. Sun, *Deep residual learning for image recognition*, In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778. IEEE, doi:10.1109/CVPR.2016.90 (2016).

[23] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, *Dropout: A simple way to prevent neural networks from overfitting*, Journal of Machine Learning Research **15**(1), 1929–1958 (2014).

[24] S. Rendle, *Factorization machines*, In *2010 IEEE International Conference on Data Mining*. IEEE, doi:10.1109/icdm.2010.127 (2010).

[25] A. Novikov, M. Trofimov and I. V. Oseledets, *Exponential machines*, In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*. OpenReview.net (2017).

[26] H. B. Demuth, M. H. Beale, O. De Jess and M. T. Hagan, *Neural Network Design*, Martin Hagan, Stillwater, OK, USA, 2nd edn., ISBN 0971732116 (2014).

[27] I. V. Oseledets, *Tensor-train decomposition*, SIAM Journal on Scientific Computing **33**(5), 2295 (2011), doi:10.1137/090752286.

[28] D. Perez-García, F. Verstraete, M. M. Wolf and J. I. Cirac, *Matrix product state representations*, Quantum Information and Computation **7**(5-6), 401 (2007), doi:10.5555/2011832.2011833.

[29] L. van der Maaten and G. Hinton, *Visualizing data using t-sne*, Journal of Machine Learning Research **9**(86), 2579 (2008).

[30] L. van der Maaten, *Learning a parametric embedding by preserving local structure*, In D. van Dyk and M. Welling, eds., *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, vol. 5 of *Proceedings of Machine Learning Research*, pp. 384–391. PMLR, Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA (2009).

[31] S. Efthymiou, J. Hidary and S. Leichenauer, *Tensor network for machine learning* (2019), 1906.06329v1.

[32] J. Gao, L.-F. Qiao, Z.-Q. Jiao, Y.-C. Ma, C.-Q. Hu, R.-J. Ren, A.-L. Yang, H. Tang, M.-H. Yung and X.-M. Jin, *Experimental machine learning of quantum states*, Physical Review Letters **120**, 240501 (2018), doi:10.1103/PhysRevLett.120.240501.

[33] X. Glorot, A. Bordes and Y. Bengio, *Deep sparse rectifier neural networks*, In *14th International Conference on Artificial Intelligence and Statistics*, vol. 15, pp. 315–323 (2011).

[34] A. F. Agarap, *Deep learning using rectified linear units (relu)* (2018), 1803.08375v2.

[35] P. Baldi and P. J. Sadowski, *Understanding dropout*, In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani and K. Q. Weinberger, eds., *Advances in Neural Information Processing Systems*, vol. 26, pp. 2814–2822. Curran Associates, Inc. (2013).

[36] W. Zaremba, I. Sutskever and O. Vinyals, *Recurrent neural network regularization* 1409.2329v5.

[37] Y. LECUN, *The mnist database of handwritten digits*, http://yann.lecun.com/exdb/mnist/ .

[38] E. M. Stoudenmire, *Learning relevant features of data with multi-scale tensor networks*, Quantum Science and Technology **3**(3), 034003 (2018), doi:10.1088/2058-9565/aaba1a.

[39] K. Meshkini, J. Platos and H. Ghassemain, *An analysis of convolutional neural network for fashion images classification (fashion-mnist)*, In S. Kovalev, V. Tarassov, V. Snasel and A. Sukhanov, eds., *Proceedings of the Fourth International Scientific Conference "Intelligent Information Technologies for Industry" (IITI'19)*, pp. 85–95. Springer International Publishing, Cham, ISBN 978-3-030-50097-9, doi:10.1007/978-3-030-50097-9_10 (2020).

[40] T. G. Kolda and B. W. Bader, *Tensor decompositions and applications*, SIAM Review **51**(3), 455 (2009), doi:10.1137/07070111x.

[41] H. Xiao, K. Rasul and R. Vollgraf, *Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms* (2017), 1708.07747v2.

[42] H. N. Phien, G. Vidal and I. P. McCulloch, *Infinite boundary conditions for matrix product state calculations*, Physical Review B **86**(24), 245107 (2012), doi:10.1103/physrevb.86.245107.

[43] J. Miller, G. Rabusseau and J. Terilla, *Tensor networks for probabilistic sequence modeling*, In A. Banerjee and K. Fukumizu, eds., *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, vol. 130 of *Proceedings of Machine Learning Research*, pp. 3079–3087. PMLR (2021).

[44] S. R. White, *Density matrix formulation for quantum renormalization groups*, Physical Review Letters **69**, 2863 (1992), doi:10.1103/PhysRevLett.69.2863.

[45] M. Hardt and T. Ma, *Identity matters in deep learning* (2016), 1611.04231v3.

[46] L.-L. Huang, A. Shimizu, Y. Hagihara and H. Kobatake, *Face detection from cluttered images using a polynomial neural network*, Neurocomputing **51**, 197 (2003), doi:10.1016/s0925-2312(02)00616-1.

[47] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, doi:10.48550/ARXIV.1412.6980 (2014).

[48] J. Eisert, *Entanglement and tensor network states*, Modeling and Simulation 3, 520 (2013) (2013), 1308.3318.