

Tensor Network for Supervised Learning at Finite Temperature

Haoxiang Lin¹, Shuqian Ye¹, Xi Zhu^{1*}

¹Shenzhen Institute of Artificial Intelligence and Robotics for Society,
the Chinese University of Hong Kong, Shenzhen
{haoxianglin, shuqianye}@link.cuhk.edu.cn, zhuxi@cuhk.edu.cn

Abstract

The large variation of datasets is a huge barrier for image classification tasks. In this paper, we embraced this observation and introduce the finite temperature tensor network (FTTN), which imports the thermal perturbation into the matrix product states framework by placing all images in an environment with constant temperature, in analog to energy-based learning. Tensor network is chosen since it is the best platform to introduce thermal fluctuation. Different from traditional network structure which directly takes the summation of individual losses as its loss function, FTTN regards it as thermal average loss computed from the entanglement with the environment. The temperature-like parameter can be automatically optimized, which gives each database an individual temperature. FTTN obtains improvement in both test accuracy and convergence speed in several datasets. The non-zero temperature automatically separates similar features, avoiding the wrong classification in previous architecture. The thermal fluctuation may give a better improvement in other frameworks, and we may also implement the temperature of database to improve the training effect.

1 Introduction

Originated from quantum many-body physics and quantum information sciences, tensor networks (TNs) states are building blocks designed for the manipulation of very high-dimensional data, representing the complex quantum states [Kuhn and Richter, 2019]. Key information of the overall wave function is encoded in individual block, and the insights into the entire framework enable the powerful numerical simulation approaches [Orús, 2019; Orús, 2014]. In the 2000s, TNs are applied in quantum information theory to explore many-body entanglement in the low-energy eigenstates, like the ground state quantum spin chain in 1D [Fannes *et al.*, 1992], a lattice model on a thin 2D torus [Milsted *et al.*, 2019], and also excited states like exciton and bi-exciton [Kuhn and Richter, 2019]. Anywhere there is a correlation,

there is also room to apply TNs' ability of estimating many-body entanglement.

Now, TNs are active in interdisciplinary areas besides physics. In recent years, great progress has been made in applying tensor networks to machine learning for both supervised and unsupervised tasks: the matrix product state (MPS) can be used as classifier [Stoudenmire and Schwab, 2016], the tree tensor network (TTN) is equivalent to a deep convolutional arithmetic circuit (ConvAC) [Levine *et al.*, 2017], and both MPS and TNN can be designed as generative models [Han *et al.*, 2018; Cheng *et al.*, 2019]. Meanwhile, TN has been investigated for other machine learning applications like the compressing of neural network weight layers [Novikov *et al.*, 2015], the compressed sensing [Ran *et al.*, 2019] and data completion [Wang *et al.*, 2016]. TN is proven to be one of the most suitable platforms in connecting physics and machine learning, which is the reason why we choose TN as the platform to insert thermal fluctuation.

One excellent optimization method for MPS classifier is adapted from density matrix renormalization group (DMRG) algorithm in physics. [Stoudenmire and Schwab, 2016] Besides this DMRG-like optimization algorithm, automatic gradient can also be implied to network training [Efthymiou *et al.*, 2019]. However, despite great achievements, they use a tensor network structure corresponding to physical system at high temperature, all states are calculated. If the variation between graphs from the same label is small and that from different labels is large, all features have large separation, so the analysis of all states is proper and the noise does not matter a lot. However, when the variation between images from the same label is relatively large and that from different labels is narrowed, the inputs are becoming complex so that we need to focus on the main feature. The proposed solution is thermal fluctuation, which requires us to find the typical wave function at finite temperature. In physics, one useful approach is the Minimally Entangled Typical Quantum States (METTS), which ensemble the average of pure states to obtain an excellent approach. A finite-T DMRG algorithm is also provided to obtain the states at finite temperature [Stoudenmire and White, 2010].

In this work, we combine the finite temperature systems with tensor network, by introducing an extra temperature layer. In Section 2, we construct the tensor network structure at finite temperature (FTTN), and its corresponding op-

*Contact Author

timization algorithm is described in Section 3. Then we give a physical interpretation in Section 4. To summarize the result as depicted in Section 5, we find that, by introducing a temperature layer into the MPS, similar features can be separated, giving improvement on the test accuracy and convergence speed. For large variation dataset, Fashion-MNIST [Xiao *et al.*, 2017], the test accuracy is increased from 87.73% to 88.72%, around one percentage increment. Even for the small variation dataset, MNIST [LeCun *et al.*, 1998], the test accuracy is increased from 98.31% to 98.43%. This method can give rise to other machine learning frameworks like graph neural network (GNN) and convolutional neural network (CNN).

2 Tensor Network and Temperature Layer

In this section, we briefly introduce the relationship between MPS for quantum physics calculation and MPS for machine learning. Then we propose our temperature-endorsed tensor network structure for machine learning. We will concretely interpret our FTTN framework later in Section 4.



Figure 1: Images with same label from different datasets. a) images with label “0” from MNIST dataset. b) images with label “ankle boot” from fashion-MNIST dataset. It can be seen that the first three images are very similar, but the latter three have large variation.

In this work, we focus on the image classifications task. One dataset that traditional neural network structure performs badly is the large variation dataset. As shown in Figure 1, the first three images are adapted from MNIST with the same label “0”. They share the same shape, a feature “circle” is enough to describe it. The traditional networks give larger than 97% [Stoudenmire and Schwab, 2016] without a complex framework. However, when it comes to large variation datasets like fashion-MNIST as shown in Figure 1b, state of the art deep learning methods obtaining about 93% test accuracy [Bhatnagar *et al.*, 2017]. A huge difference exists between these images even though they come from the same label “ankle boot”. It is hard to describe them simply using one feature. Compared with the first image, the second one is shaper and the third image has an extra heel. Even though human can notice these features, it is harder for network structure to distinguish their similarity.

Though MPS is specialized in estimating one-dimensional correlation, it can also be applied to higher-dimensional data

as well, where the two-dimension image is mapped into a one-dimensional chain. The non-linear kernel learning requires the manipulation of very large tensor, where the input data \mathbf{X} are mapped into a high dimension space by a feature map Ψ before the final decision $f(\mathbf{X}) = \mathbf{W} \cdot \Psi(\mathbf{X})$. To deal with the large tensors, the kernel trick is adapted, which only requires working with scalar products of feature tensors [Muller *et al.*, 2001]. To apply TN to machine learning, a feature map Ψ in the form of local map ψ multiplication

$$\Psi(\mathbf{X}) = \Psi^{(S)}(\mathbf{p}) = \psi^{(S_1)}(p_1) \otimes \psi^{(S_2)}(p_2) \otimes \dots \otimes \psi^{(S_N)}(p_N) \quad (1)$$

is adapted, where $S = [S_1, S_2, \dots, S_N]$ is the pixel position and $\mathbf{p} = [p_1, p_2, \dots, p_N]$ is its corresponding gray scale value ranging from 0 for black to 1 for white, \otimes is the Kronecker product. One feature map used in [Stoudenmire and Schwab, 2016] is $\psi(p) = [\cos(\pi p/2) \quad \sin(\pi p/2)]^T$, mapping the gray-scale value to a quantum spin. Another feature map is $\psi(x) = [1 - p \quad p]^T$, which is used in [Efthymiou *et al.*, 2019]. Sometimes this column vector is written as $|\psi\rangle$ and its conjugate transpose is written as $\langle\psi|$. But the selection of feature map is not important. These local map ψ correspond to individual blue circle with an edge as shown in Figure 2(a). For a certain learning task and a specified feature map, a MPS \mathbf{W} (contracted sequence of low-order tensors $A^{(S)}$, the yellow cubic shown in Figure 2b) can approximate weight tensor ,

$$\mathbf{W}^{(S)} = \sum_{\chi_1, \chi_2, \dots, \chi_N} A_{i_1 \chi_1}^{(S_1)} A_{i_2 \chi_1 \chi_2}^{(S_2)} \dots A_{i_N \chi_N}^{(S_N)} \quad (2)$$

the subscript of $A^{(S)}$ is its edges and the connecting edge between nearest tensors is the bond dimensions χ , one hyperparameter of the tensor network. Figure 2 indicates the structure. The lines represents tensor edges and the sharing edges indicates summation. For example, the contraction of left-most weight tensor $A_{i_1 \chi_1}^{(S_1)}$ and feature map tensor $\psi^{(S_1)}(p_1)$ gives $\sum_{i_1} A_{i_1 \chi_1}^{(S_1)} \psi^{(S_1)}(p_1)$, a rank 2 tensor, also known as matrix. The label index l is placed arbitrarily at one tensor and this tensor is named as label tensor. If the weight matrix and the feature map are connected, the entire tensor network represents the final decision $f(\mathbf{X}) = \mathbf{W} \cdot \Psi(\mathbf{X})$.

The MPS framework in Figure 2(b) is in analog to calculating the average of an observable when the feature map only consists of real number. Originally in physics, the expectation of an observable is $\langle\psi|H|\psi\rangle$, where the Hamiltonian H is a transpose conjugate matrix standing for the observable, similar to the weight layer. Indices connected to the wave functions $\langle\psi|$ and $|\psi\rangle$ are generally the same, or sometimes conjugated. [Fannes *et al.*, 1992; Klümper *et al.*, 1993; Orús, 2014]. As for the machine learning task, one of these two indices is truncated as shown in Figure 2(b). This is the same MPS structure as [Stoudenmire and Schwab, 2016; Efthymiou *et al.*, 2019], if we treat the observable H as a modified weight tensor.

Inspired by the very successful Minimally Entangled Typical Quantum States (METTS) algorithm [Stoudenmire and White, 2010], we propose an MPS framework at finite temperature. First, we need to recover the original tensor network structure with two input wave functions as shown in the left

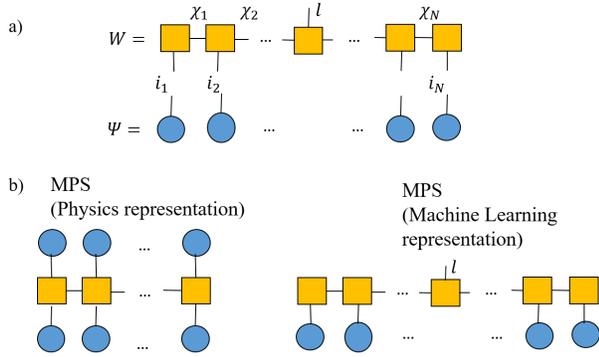


Figure 2: a) The tensor network representation of weight tensor W and feature map Ψ . Each pixel of an image is encoded in a rank-1 tensor, also known as vector, corresponding to Equation 1. The entire weight tensor W is approximated by contracted sequence of low-order tensors A , corresponding to Equation 2. b) MPS for quantum physics and machine learning. The left sub-figure stands for the expected value of an observable, while the right sub-figure is the tensor network architecture for machine learning.

sub-figure of Figure 2b. The diagonal element of $A'_{:,ij,:}$ is set as the element of $A_{:,ij}$, where i and j are the inter-layer edges, this procedure adds an extra edge. Notice that now the weight tensor $A^{(s)}$ has the rank of 4 except the boundary tensor which has the rank of 3. Then the temperature layer T is constructed by taking the exponential of recovered weight tensors $-\beta A'$ where β is a temperature-like constant, which is $T = \exp(-\beta A')$. The upper and lower edges of the temperature layer are connected to the input feature map and the weight layer. This recovery step is necessary since in machine learning representation, the rank of weight matrix is 3 except the boundary tensor which has the rank of 2. If we simply take exponential of these weight matrices, the number of edges is not enough to connect both the weight layer and the feature map. From this step, the tensor network structure is formulated as shown in the left sub-figure of Figure 3. Then we use the initial weight tensor to replace this modified weight tensor to truncate the upper part of this framework, providing a simplified version as shown in the right sub-figure of Figure 3. Now, FTTN is constructed.

In this work, we are interested in classifying data with given hidden labels. The contraction of the entire FTTN framework gives a vector, whose largest element corresponds to the output class.

3 Optimization Algorithm

In this section, we focus on the optimization algorithm of FTTN. It is a slightly modified version based on gradient descent.

As for the Bayesian neural network, there exists another way to incorporate temperature effect. [Baldock and Marzari, 2019] This method defines the potential energy of the network according to the calculated loss, and then the posterior distribution is temperature-adjusted based on input batches and the potential energy. However, in the FTTN, the potential energy is based on the weight tensor A instead of the output

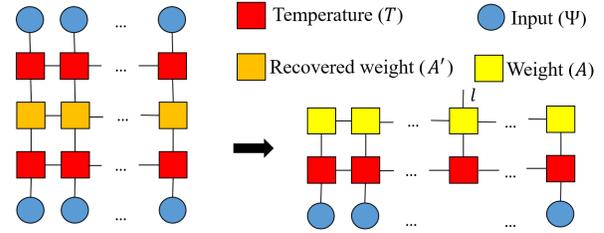


Figure 3: The evolution of our tensor network for supervised learning at finite temperature. The left sub-figure is the physical representation of calculating observables with the endorsement of the temperature layer. The middle sub-figure is the simplified version, where we truncate one edge to reduce the number of tensors. The right sub-figure contract the weight tensor with its corresponding temperature tensor to obtain a simplified version, which is in analog to the previous MPS framework.

from the entire network. Such a method can be useful, but in this work, we use another algorithm. Here the loss function is set as multi-class cross-entropy defined as:

$$\text{Loss} = - \sum_{(\mathbf{x}_i, \mathbf{y}_i)} \log \left(\text{Softmax} \left(f^{(\mathbf{y}_i)}(\mathbf{x}_i) \right) \right)$$

The gradient of each individual tensor $A^{(S)}$ can be computed through a “sweeping” optimization algorithm [Stoudenmire and Schwab, 2016] or automatic gradients which is built-in to TensorFlow [Roberts *et al.*, 2019; Efthymiou *et al.*, 2019]. By adding the coefficient to the obtained gradient, FTTN can be optimized easily without great change. After revising the weight layer, the temperature layer is simultaneously adjusted. Other optimization methods can also be applied to FTTN as long as its gradient can be computed and adjusted based on our proposed method.

Due to the endorsement of temperature layer, the back propagation process is modified. The exact solution is complex, and we propose an approximation. Notice that the temperature layer (T) and the weight layer (W) are strongly related by the formula $T = \exp(-\beta W)$. We first contracted the temperature layer with its corresponding weight layer. This contraction produces a T -dependent weight layer \hat{O} :

$$\hat{O} = A \cdot \exp(-\beta \cdot A) \quad (3)$$

Therefore, in the back-propagation process, an coefficient C need to be added:

$$C = \frac{\partial \hat{O}}{\partial A} = (1 - \beta A) \cdot \exp(-\beta A) \quad (4)$$

Compared with the traditional back-propagation method for TN, the DMRG-like optimization method, the adjustment is the change of gradient. This is in analogue to modify the gradient matrix by multiplying the coefficient c_{ijk} on each element of the matrix. For the other built-in optimizer of TensorFlow, we can also directly change the gradient matrix to optimize FTTN.

Even though the sweeping algorithms like the DMRG are preferred in physical applications due to its fast convergence,

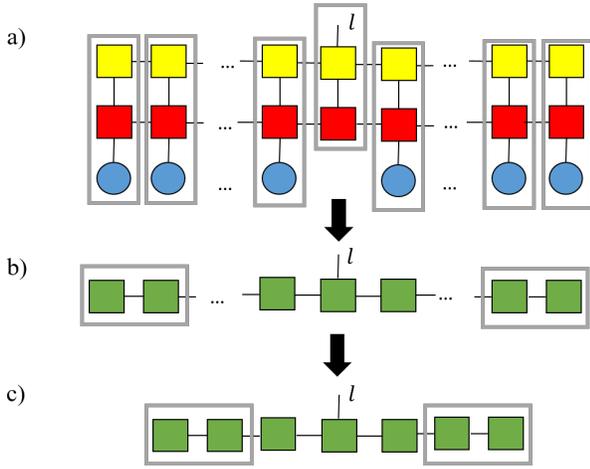


Figure 4: The three steps contraction order of FTTN. (a) Step 1, contract temperature tensor with its corresponding weight tensor and local feature map. This creates new effective tensors depicted as green squares. (b) Step 2, contract the generated tensors in pairs. This step is conducted independently for each pair of them in parallel. (c) Step 3, repeat step 2 until the tensor chain is fully contracted.

the simplicity of gradient-based optimization are more attractive to machine learning applications, which is the method we implied. In this FTTN framework, after calculating the gradient matrix, we calculate the coefficient matrix based on equation 4. Then the coefficient matrix is used to adjust the gradient. The temperature layer is adjusted based on the weight layer.

The contraction order matters the computation complexity. The DMRG-like order gives the total time complexity of order $O(NLd^2\chi^2)$ [Stoudenmire and Schwab, 2016], but it can not be forwarded to parallelized computation. Here we use a parallelized contraction order. The first step contracts the temperature layer with its corresponding weight layers and the feature map. This step gives a tensor chain. In the second step, we contract the nearest tensors in pair, then this step is repeated until the tensor chain is fully contracted. The total time complexity of this contraction is $O(Nd^2\chi^2 + \chi^3 \log(N))$. Even though the total cost is near the same as the previous one, this contraction order has the advantage that each step is parallelized as matrix multiplication without requiring the calculating result from the neighboring calculation. We noticed that the parallelized computation gives great enhancement on calculation speed.

4 Physical Interpretation

In this section, we will briefly investigate the MPS framework and the effect of the endorsed temperature layer. This layer gives a thermal perturbation to the feature map, flatten the noise. The dataset with large variation contains large noise, and the endorsement of the temperature layer flatten the noise to obtain a better performance.

To summarize our tensor network framework, we embed data into space in the form of tensor, then the inner product between data and weight gives the final prediction. It should be noticed that the MPS tensor A can be interpreted as an

other vector in the tensor space, intuitively, a combination of all images in a given class. Ideally, for a specific class, an image not belonging to this class should be orthogonal to the MPS tensor, while an image belonging to this class has a large overlap with the MPS tensor. In physics, this is in analog to calculate the probability for any outcome of well-defined observables, which is calculated through the density matrix ρ . One may define the loss function as potential energy, and now the optimization is in analog to obtain the ground states. Different from traditional network structure which directly takes the summation of individual losses as its loss function, the input feature goes through a temperature layer to be entangled with environment, in the other words, FTTN regards the loss as thermal average loss computed from the entanglement with the environment.

The insertion of the temperature layer is inspired by the METTS algorithm in physics applications [White, 2009]. The fundamental proposition of statistical mechanics indicates that the density matrix of a system at inverse temperature β with Hamiltonian H is $\rho = \exp(-\beta H)$. One can regard this formula as arising from the quantum mechanical entanglement with a heat bath which produces mixed states. In other words, FTTN regards it as thermal average loss computed from the entanglement with the environment instead of the direct summation. As for this classification task, the MPS tensor can be treated as the heat bath with a certain Fermi energy, and the individual local feature map entangled with the MPS tensor to produce mixed states. We idealized the statistical mechanism, the physical system has a specific history and environment. We equilibrate the system with weak coupling to a heat bath (MPS tensor), then moving coupling between the heat bath and the system.

Here a typical set of states $\{|\psi(i)\rangle\}$ with probabilities $P(i)$ need to be chosen, it satisfies the fundamental proposition:

$$\sum_i P(i) |\psi(i)\rangle \langle \psi(i)| = \exp(-\beta H) \quad (5)$$

From this equation, the expectation value of any Hermitian operator H can be determined from average of $\langle \psi(i) | H | \psi(i) \rangle$, where each $|\psi(i)\rangle$ is chosen randomly according to $P(i)$. Let $|i\rangle$ to be a complete orthonormal basis, then one specific states satisfies Equation 5 can be:

$$\begin{aligned} |\psi(i)\rangle &= P(i)^{-1/2} \exp(-\beta H/2) |i\rangle \\ P(i) &= \langle i | \exp(-\beta H) | i \rangle = \text{trace}(\rho |i\rangle \langle i|) \end{aligned} \quad (6)$$

Notice these results do not require the states $\{|i\rangle\}$ to be the orthonormal basis. From the physical aspect, the final result should be divided by a partition function $Z = \text{Tr}(\rho) = \sum_i P(i)$, but here it is omitted since the final result is determined by the largest one.

The insertion of our proposed temperature layer is in analog to this METTS algorithm. The contraction of the temperature layer and the feature map gives a thermal perturbation to inputs. Here we give an intuitive explanation, if a picture contains a perfectly straight line, both humans and computers can easily distinguish it. However, if the line is slightly bent, humans can still treat it as a straight line. But for computers, by accurate calculation, the line is treated as a curve rather

than a line. For thermal fluctuation, the bent line is treated as the superposition state of line and curve, which makes the network structure treat it as line by ignoring the small perturbation. In other words, the temperature-like parameter β is an analog to a threshold. If the difference between input and feature is smaller than the threshold, they are treated as the same; however, for input and features with difference larger than the threshold, they are separated.

Mathematically, for a small $\beta \rightarrow 0$, this temperature layer approaches the identity matrix, corresponding to the MPS machine learning structure without the temperature layer. However, for a large β , due to the exponential decay, the variation between features is exponentially flattened, resulting in an entirely black image. A proper β emphasizes the main feature and by choosing a proper temperature, the image contrast is adjusted to produce easy-to-train images.

5 Experiment

In this section, we implement FTTN using TensorNetwork [Roberts *et al.*, 2019] with TensorFlow [Abadi *et al.*, 2016] backend. Here we focus on the result on Fashion-MNIST due to its large variance, while FTTN also demonstrates improvement on other small-variance datasets like MNIST.

For TN without thermal fluctuation, it is observed that training with automatic gradients is nearly independent of the used bond dimension [Efthymiou *et al.*, 2019], while the prior bond dimension matters when using the DMRG-like sweeping algorithm [Stoudenmire and Schwab, 2016]. One possible reason is that the latter one changes the bond dimension during the optimization process, while the previous one does not. Therefore, we adapt the automatic gradient descent method rather than the DMRG-like sweeping algorithm, since we want to focus on the temperature-like parameter β instead of other parameters like bond dimension.

We train on the total fashion-MNIST training set consisting of 60,000 images of size 28×28 . The test accuracy is calculated on the whole test dataset consisting of 10,000 images. To illustrate the effect of the temperature layer, we use the same setup as [Efthymiou *et al.*, 2019], where the typical setting that performs well is to use Adam optimizer [Kingma and Ba, 2014] with a learning rate of 10^{-4} and the batch size of 50. We adapt the multi-class cross-entropy as loss function, which is defined in Equation 3. The feature map is $\psi(x) = [1 - p \quad p]^T$. We also evaluate our method to MNIST dataset, which has relatively small variation.

Ideally, the performance of FTTN achieves best when the given temperature is the same as that of the dataset. Here we adopt a simulated annealing algorithm to do the automatically optimization since they share a similar physical process. FTTN is given a randomly selected temperature. Then for each step, the parameter β close to the current one is selected and its corresponding accuracy is measured, then this algorithm decides whether move to it or not depending on the temperature and the fact whether the new solution is better or worse than the previous one. The optimization process of this temperature-like parameter β is shown in Figure 5. It can be seen that the temperature-like hyper-parameter β converges to 0.4 for fashion-MNIST, independent of the bond di-

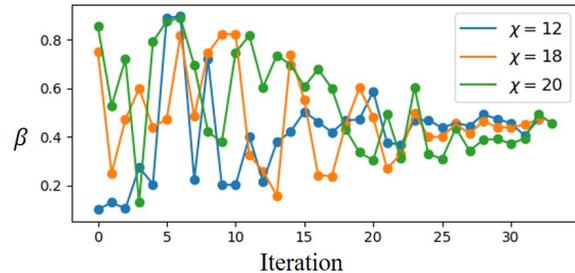


Figure 5: The optimization process of the temperature-like parameter β .

mension. Such parameter might be the intrinsic property of a dataset, and obtaining such property can be useful in training machine-learning-based tensor network frameworks.

Experiments observe that FTTN has following advantages: **Test Accuracy** As shown in Figure 6, we analyze the effect of thermal perturbation with different bond dimensions χ as shown in Figure 6. Without thermal perturbation, the test accuracy increase with bond dimension χ at first, but it goes through a decrease after a critical number. The introduced thermal perturbation does not change this tendency, but it can result in an overall improvement in test accuracy. Similar to what observed without thermal fluctuation, FTTN has tiny dependence on the bond dimension. We have seen that the thermal perturbation gives rise to the training accuracy and the convergence speed. The endorsement of the temperature layer gives an accuracy rising from 87.73% to 88.72%, a one percentage increment for the database Fashion-MNIST. This means FTTN is suitable for this database since it has a large variation. However, even for the small variation database MNIST, FTTN also gives an accuracy increment from 98.31% to 98.43%.

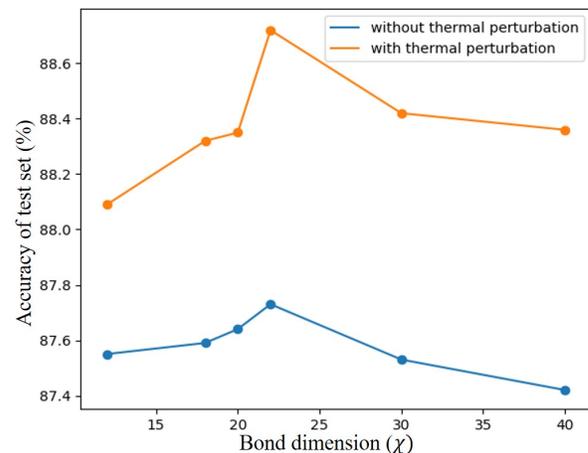


Figure 6: The χ dependent bond accuracy. Orange and blue lines correspond to the testing accuracy for Fashion-MNIST using tensor network with/without thermal perturbation respectively. Here the model of Figure 2a refer to the case without thermal perturbation.

Speed of Convergence We observed that the temperature layer can improve the speed of convergence. The speed of convergence is significantly improved in the first several epochs as shown in Figure 7. Here the bond dimension χ is set as 12. We can see that the proper β gives a larger than 2% increment on the testing accuracy during the first several epoches. From our experiment, even though the improvement will decrease with increasing epoches, the improvement still possesses a considerable quantity.

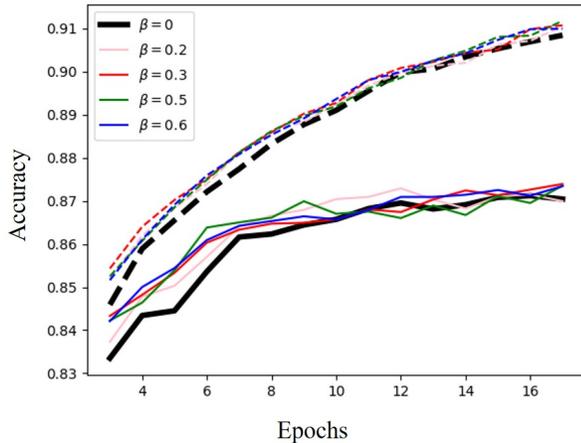


Figure 7: The first 3-18 epoch of FTTN with different β . Here we use $\chi = 12$ as an example, while similar situation happens for other χ value. The solid and dashed line represent the accuracy for test and training set respectively.

The thermal perturbation effect is illustrated in Figure 8. The photo shown in Figure 8 is inaccurately classified as “coat” without perturbation, while the thermal fluctuation gives it a correct label “dress”. We tried the traditional CNN structure using two convolutional layers and two pooling layers [Xiao *et al.*, 2017], despite its higher accuracy, this image can not be accurately classified. One of the differences is that the middle section is separated for “coat” while for “dress”, the entire middle part is treated as a whole. Initially at $\beta = 0$, the feature of dress and coat degenerated since they share similar features, here a line in the middle is added to emphasize their differences. However, these two features are separated with the increasing β value, which is the effect of the temperature layer. A proper β gives enough separation to similar features.

6 Conclusions

In this work, we introduce a thermal perturbation into the tensor network structure by insertion of a temperature layer and then we analyze the effect of this thermal perturbation. The constructed structure is in analogue to that from METTS algorithm [Stoudenmire and White, 2010]. Such modification leads to around one percentage increment of accuracy in Fashion-MNIST dataset. Meanwhile, the convergence speed is accelerated, which is significant in the first several epoches. We hope that this technique we proposed here will be taken up by the wider machine learning community.

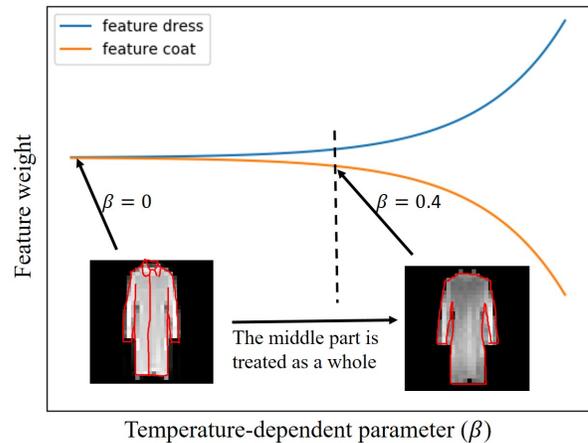


Figure 8: The schematic diagram of the thermal fluctuation effect. The left figure is adapt from fashion-MNIST, while the right one is this image with thermal fluctuation. The perturbation separates similar features of “coat” and “dress” to make a correct classification.

While using an one-dimension MPS ansatz for the classification works well even for two-dimensional data as shown in this work, there still exist some other feasible tensor network structures, like projected entangled pair states (PEPS) [Verstraete and Cirac, 2004], multiscale entanglement renormalization ansatz (MERA) [Vidal, 2007], comb tensor networks [Chepiga and White, 2019]. Our method of introducing thermal fluctuation may also be applied to these frameworks and gives an enhancement. There is also much room to improve the temperature-endorsed optimization algorithm by incorporating with other frameworks like convolutional neural network and graph neural network. Besides the thermal fluctuation, symmetry is excellent in improving tensor network [Vanhecke *et al.*, 2019]. The recurrent neural network has been proven as an analogy to wave physics [Hughes *et al.*, 2019], and we believe the physics-endorsed frameworks will greatly contribute to the machine learning community.

References

- [Abadi *et al.*, 2016] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pages 265–283, 2016.
- [Baldock and Marzari, 2019] Robert J. N. Baldock and Nicola Marzari. Bayesian neural networks at finite temperature, 2019.
- [Bhatnagar *et al.*, 2017] Shobhit Bhatnagar, Deepanway Ghosal, and Maheshkumar H Kolekar. Classification of fashion article images using convolutional neural networks. In *2017 Fourth International Conference on Image Information Processing (ICIIP)*, pages 1–6. IEEE, 2017.

- [Cheng *et al.*, 2019] Song Cheng, Lei Wang, Tao Xiang, and Pan Zhang. Tree tensor networks for generative modeling. *Physical Review B*, 99(15):155131, 2019.
- [Chepiga and White, 2019] Natalia Chepiga and Steven R White. Comb tensor networks. *Physical Review B*, 99(23):235426, 2019.
- [Efthymiou *et al.*, 2019] Stavros Efthymiou, Jack Hidary, and Stefan Leichenauer. Tensor network for machine learning. *arXiv preprint arXiv:1906.06329*, 2019.
- [Fannes *et al.*, 1992] Mark Fannes, Bruno Nachtergaele, and Reinhard F Werner. Finitely correlated states on quantum spin chains. *Communications in mathematical physics*, 144(3):443–490, 1992.
- [Han *et al.*, 2018] Zhao-Yu Han, Jun Wang, Heng Fan, Lei Wang, and Pan Zhang. Unsupervised generative modeling using matrix product states. *Physical Review X*, 8(3):031012, 2018.
- [Hughes *et al.*, 2019] Tyler W Hughes, Ian AD Williamson, Momchil Minkov, and Shanhui Fan. Wave physics as an analog recurrent neural network. *arXiv preprint arXiv:1904.12831*, 2019.
- [Kingma and Ba, 2014] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.
- [Klümper *et al.*, 1993] A Klümper, A Schadschneider, and J Zittartz. Matrix product ground states for one-dimensional spin-1 quantum antiferromagnets. *EPL (Europhysics Letters)*, 24(4):293, 1993.
- [Kuhn and Richter, 2019] Sandra C Kuhn and Marten Richter. Combined tensor network/cluster expansion method using logic gates: Illustrated for (bi) excitons by a single-layer mos 2 model system. *Physical Review B*, 99(24):241301, 2019.
- [LeCun *et al.*, 1998] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- [Levine *et al.*, 2017] Yoav Levine, David Yakira, Nadav Cohen, and Amnon Shashua. Deep learning and quantum entanglement: Fundamental connections with implications to network design. *arXiv preprint arXiv:1704.01552*, 2017.
- [Milsted *et al.*, 2019] Ashley Milsted, Martin Ganahl, Stefan Leichenauer, Jack Hidary, and Guifre Vidal. Tensor network on tensorflow: A spin chain application using tree tensor networks. *arXiv preprint arXiv:1905.01331*, 2019.
- [Muller *et al.*, 2001] K-R Muller, Sebastian Mika, Gunnar Ratsch, Koji Tsuda, and Bernhard Scholkopf. An introduction to kernel-based learning algorithms. *IEEE transactions on neural networks*, 12(2):181–201, 2001.
- [Novikov *et al.*, 2015] Alexander Novikov, Dmitrii Podoprikin, Anton Osokin, and Dmitry P Vetrov. Tensorizing neural networks. In *Advances in neural information processing systems*, pages 442–450, 2015.
- [Orús, 2014] Román Orús. A practical introduction to tensor networks: Matrix product states and projected entangled pair states. *Annals of Physics*, 349:117–158, 2014.
- [Orús, 2019] Román Orús. Tensor networks for complex quantum systems. *Nature Reviews Physics*, 1(9):538–550, 2019.
- [Ran *et al.*, 2019] Shi-Ju Ran, Zheng-Zhi Sun, Shao-Ming Fei, Gang Su, and Maciej Lewenstein. Quantum compressed sensing with unsupervised tensor-network machine learning, 2019.
- [Roberts *et al.*, 2019] Chase Roberts, Ashley Milsted, Martin Ganahl, Adam Zalcman, Bruce Fontaine, Yijian Zou, Jack Hidary, Guifre Vidal, and Stefan Leichenauer. Tensor network: A library for physics and machine learning. *arXiv preprint arXiv:1905.01330*, 2019.
- [Stoudenmire and Schwab, 2016] Edwin Stoudenmire and David J Schwab. Supervised learning with tensor networks. In *Advances in Neural Information Processing Systems*, pages 4799–4807, 2016.
- [Stoudenmire and White, 2010] EM Stoudenmire and Steven R White. Minimally entangled typical thermal state algorithms. *New Journal of Physics*, 12(5):055026, 2010.
- [Vanhecke *et al.*, 2019] Bram Vanhecke, Laurens Vanderstraeten, and Frank Verstraete. Symmetric cluster expansions with tensor networks, 2019.
- [Verstraete and Cirac, 2004] Frank Verstraete and J Ignacio Cirac. Renormalization algorithms for quantum-many body systems in two and higher dimensions. *arXiv preprint cond-mat/0407066*, 2004.
- [Vidal, 2007] Guifre Vidal. Entanglement renormalization. *Physical review letters*, 99(22):220405, 2007.
- [Wang *et al.*, 2016] Wenqi Wang, Vaneet Aggarwal, and Shuchin Aeron. Tensor completion by alternating minimization under the tensor train (tt) model, 2016.
- [White, 2009] Steven R White. Minimally entangled typical quantum states at finite temperature. *Physical review letters*, 102(19):190601, 2009.
- [Xiao *et al.*, 2017] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.