A Support Tensor Train Machine

Chen, Cong; Batselier, Kim; Ko, Ching Yun; Wong, Ngai

DOI
10.1109/IJCNN.2019.8851985

Publication date
2019

Document Version
Final published version

Published in
Proceedings of the 2019 International Joint Conference on Neural Networks (IJCNN 2019)

A Support Tensor Train Machine

Cong Chen*, Kim Batselier**, Ching-Yun Ko*, and Ngai Wong*

*Department of Electrical and Electronic Engineering, The University of Hong Kong, Hong Kong, China
Email: {chencong, cyko, nwong}@eee.hku.hk
**Delft Center for Systems and Control, Delft University of Technology, Delft, The Netherlands
Email: k.batselier@tudelft.nl

***Abstract*—There has been growing interest in extending traditional vector-based machine learning techniques to their tensor forms. Support tensor machine (STM) and support Tucker machine (STuM) are two typical tensor generalization of the conventional support vector machine (SVM). However, the expressive power of STM is restrictive due to its rank-one tensor constraint, and STuM is not scalable because of the exponentially sized Tucker core tensor. To overcome these limitations, we introduce a novel and effective support tensor train machine (STTM) by employing a general and scalable tensor train as the parameter model. Experiments validate and confirm the superiority of the STTM over SVM, STM and STuM.**

***Index Terms*—support vector machine, tensor train, classification**

I. Introduction

Classification algorithm design has been an important topic in machine learning, pattern recognition and computer vision for decades. One of the most representative and successful classifiers is the support vector machine (SVM) [1], which achieves an enormous success in pattern classification by minimizing the Vapnik-Chervonenkis dimensions and structural risk. However, a standard SVM model is based on vector inputs and cannot directly deal with matrices or higher dimensional data structures called tensors, which are very common in real-life applications. For example, a grayscale picture is stored as a matrix which is a second-order tensor, while color pictures have a color axis and are naturally third-order tensors. The common SVM realization on such high dimensional inputs is by reshaping each sample into a vector. However, when the number of training samples is relatively small compared to the feature vector dimension, it may easily result in poor classification performance due to overfitting [2]–[4]. To overcome this, researchers have focused on exploring new data structures and corresponding numerical operations. Tensors constitute a versatile data structure which has recently received much attention in the machine learning community. In particular, tensor decomposition techniques have found various applications. In [5] a tensor train based polynomial classifier is proposed that encodes the coefficients of the polynomial as a tensor train. In [6] tensor trains are used to compress the traditional fully connected layers of a neural network into tensor train layers with much fewer parameters. Tensor trains have also been exploited to represent nonlinear predictors [7] and classifiers [8]. Moreover, the canonical polyadic (CP) tensor decomposition has been used for speeding up the convolution step in convolutional neural networks [9] and the Tucker decomposition for the classification of tensor data [10] etc.

Not surprisingly, standard SVMs have also been extended to tensor formulations yielding significant performance enhancements [11], [12]. Ref. [11] proposes a supervised tensor learning (STL) scheme by replacing the vector inputs with tensor inputs and decomposing the corresponding weight vector into a rank-1 tensor, which is trained by the alternating projection optimization method. Based on this learning scheme, [13] extends the standard linear SVM to a general tensor form called the support tensor machine (STM). Although STM lifts the overfitting problem in traditional SVMs, the expressive power of a rank-1 weight tensor is limited, which translates into an often poor classification accuracy. In [14], the rank-1 weight tensor of STM is generalized to CP forms for stronger model expressive power. However, the determination of a good CP-rank is NP-complete [15]. In [12], an STM is generalized to a support Tucker machine (STuM), which replaces the rank-1 tensor in STM with Tucker format tensor. Nevertheless, the number of parameters in the Tucker form is exponentially large, which still suffers from the curse of dimensionality. The idea of combining the tensor train decomposition and SVM is first proposed in [16]. However, the parameter tensor $\mathcal{W}$ in [16] is trained in full tensor format and a tensor train decomposition is then implemented to obtain its tensor train format. As such, the curse of dimensionality still prevails since the model parameter is still represented and trained in full tensor format.

Consequently, this work proposes a support tensor train machine (STTM) wherein the rank-1 weight tensor of an STM is replaced by a tensor train that can approximate any tensor with a scalable number of parameters. We highlight that the main difference between [16] and our work is that we train the parameter tensor $\mathcal{W}$ in tensor train format directly while [16] trains the parameter tensor $\mathcal{W}$ in full tensor format before decomposing the latter into a tensor train. Therefore, the approach in [16] still suffers from the curse of dimensionality. Our proposed STTM exhibits the following advantages:

1) With a small sample size, an STTM has comparable or better classification accuracies than the standard SVM.
2) The expressive power of a tensor train increases with its tensor train ranks. This means an STTM can capture much

This work is supported by the Hong Kong Research Grants Council under General Research Fund (GRF) Project 17246416, and the University Research Committee of The University of Hong Kong.

978-1-7281-2009-6/$31.00 paper N-20155.pdf

richer structural information than an STM and lead to an improved classification accuracy.

3) The tensor train in STTMs is more scalable than the Tucker tensor in STuMs, and results in a more efficient training especially when the number of training samples is large.
4) A tensor train mixed-canonical form can be readily exploited to further speed up algorithmic convergence.

In the following, Section II introduces some tensor basics and the key ideas of the SVM and STM. The proposed STTM is presented in Section III. Experiments are given in Section IV to show the advantages of an STTM over SVM, STM and STuM. Finally, Section V draws the conclusions.

II. Preliminaries

A. Tensor Basics

Tensors are multi-dimensional arrays that are higher order generalization of vectors (first-order tensors) and matrices (second-order tensors). A dth-order or d-way tensor is denoted as $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ and the element of $\boldsymbol{\mathcal{A}}$ by $a_{i_1 i_2 \ldots i_d}$, where $1 \leq i_k \leq n_k$, $k = 1, 2, \ldots, d$. The numbers $n_1, n_2, \ldots, n_d$ are called the dimensions of the tensor $\boldsymbol{\mathcal{A}}$. We use boldface capital calligraphic letters $\boldsymbol{\mathcal{A}}$, $\boldsymbol{\mathcal{B}}$, ...to denote tensors, boldface capital letters $\boldsymbol{A}$, $\boldsymbol{B}$, ...to denote matrices, boldface letters $\boldsymbol{a}$, $\boldsymbol{b}$, ...to denote vectors, and roman letters a, b, ...to denote scalars. $\boldsymbol{A}^T$ and $\boldsymbol{a}^T$ are the transpose of a matrix $\boldsymbol{A}$ and a vector $\boldsymbol{a}$. The unit matrix of order n is denoted $\boldsymbol{I}_n$. An intuitive and useful graphical representation of scalars, vectors, matrices and tensors is depicted in Figure 1. The unconnected edges, also called free legs, are the indices of the array. Therefore scalars have no unconnected edge, while matrices have 2 unconnected edges. We will mainly employ these graphical representations to visualize the tensor networks and operations in the following sections whenever possible and refer to [17] for more details. We now briefly introduce some important tensor operations.

Definition 1: (Tensor k-mode product): The k-mode product of a tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{n_1 \times \cdots \times n_k \times \cdots \times n_d}$ with a matrix $\boldsymbol{U} \in \mathbb{R}^{p_k \times n_k}$ is denoted as $\boldsymbol{\mathcal{B}} = \boldsymbol{\mathcal{A}} \times_k \boldsymbol{U}$ and defined by

$$\boldsymbol{\mathcal{B}}(i_1, \ldots, i_{k-1}, j, i_{k+1}, \ldots, i_d) = \sum_{i_k=1}^{n_k} \boldsymbol{U}(j, i_k)\, \boldsymbol{\mathcal{A}}(i_1, \ldots, i_k, \ldots, i_d),$$

where $\boldsymbol{\mathcal{B}} \in \mathbb{R}^{n_1 \times \cdots \times n_{k-1} \times p_k \times n_{k+1} \times \cdots \times n_d}$.

The graphical representation of a 3-mode product between a third-order tensor $\boldsymbol{\mathcal{A}}$ and a matrix $\boldsymbol{U}$ is shown in Figure 2, where the summation over the i_3 index is indicated by the connected edge.

Definition 2: (Reshaping) Reshaping is another often used tensor operation. Employing $MATLAB$ notation, "reshape($\boldsymbol{\mathcal{A}}, [m_1, m_2, \ldots, m_d]$)" reshapes the tensor $\boldsymbol{\mathcal{A}}$ into another tensor with dimensions m_1, m_2, ..., m_d. The total number of elements of the tensor $\boldsymbol{\mathcal{A}}$ must be $\prod_{k=1}^{d} m_k$.

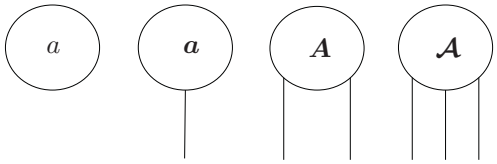

Fig. 1: Graphical representation of a scalar a, vector $\boldsymbol{a}$, matrix $\boldsymbol{A}$, and third-order tensor $\boldsymbol{\mathcal{A}}$.

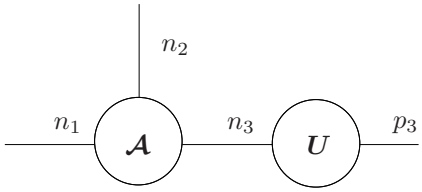

Fig. 2: 3-mode product between a 3-way tensor $\boldsymbol{\mathcal{A}}$ and matrix $\boldsymbol{U}$.

Definition 3: (Vectorization) Vectorization is a special reshaping operation that reshapes a tensor $\boldsymbol{\mathcal{A}}$ into a column vector, denoted as vec($\boldsymbol{\mathcal{A}}$).

Definition 4: (Tensor inner product) For two tensors $\boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$, their inner product $\langle \boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}} \rangle$ is defined as

$$\langle \boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}} \rangle = \sum_{i_1=1}^{n_1} \sum_{i_2=1}^{n_2} \cdots \sum_{i_d=1}^{n_d} a_{i_1, i_2, \cdots, i_d} b_{i_1, i_2, \cdots, i_d}.$$

Definition 5: (Frobenius norm) The Frobenius norm of a tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ is defined as $||\boldsymbol{\mathcal{A}}||_F = \sqrt{\langle \boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{A}} \rangle}$.

B. Tensor Decompositions

Here we introduce two related tensor decomposition methods, namely, the rank-1 tensor decomposition used in STM and the tensor train (TT) decomposition used in STTM.

1) Tensor Rank-1 Decomposition: A d-way tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ is rank-1 if it can be written as the outer product of d vectors

$$\boldsymbol{\mathcal{A}} = \boldsymbol{a}^{(1)} \circ \boldsymbol{a}^{(2)} \circ \cdots \circ \boldsymbol{a}^{(d)}, \tag{1}$$

where $\circ$ denotes the vector outer product, and each element in $\boldsymbol{\mathcal{A}}$ is the product of the corresponding vector elements:

$$\boldsymbol{\mathcal{A}}(i_1, \ldots, i_d) = \boldsymbol{a}^{(1)}(i_1) \boldsymbol{a}^{(2)}(i_2) \cdots \boldsymbol{a}^{(d)}(i_d).$$

Storing the component vectors $\boldsymbol{a}^{(1)}, \ldots, \boldsymbol{a}^{(d)}$ instead of the whole tensor $\boldsymbol{\mathcal{A}}$ significantly reduces the required number of storage elements. However, a rank-1 tensor is rare in real-world applications, so that a rank-1 approximation to a general tensor usually results in unacceptably large approximation errors. This calls for a more general and powerful tensor approximation, for which the TT decomposition serves as a particularly suitable choice.

2) Tensor Train Decomposition: A TT decomposition [18] represents a d-way tensor $\boldsymbol{\mathcal{A}}$ as d third-order tensors $\boldsymbol{\mathcal{A}}^{(1)}$, $\boldsymbol{\mathcal{A}}^{(2)}$, ..., $\boldsymbol{\mathcal{A}}^{(d)}$ such that a particular entry of $\boldsymbol{\mathcal{A}}$ is written as the following matrix product

$$\boldsymbol{\mathcal{A}}(i_1, \ldots, i_d) = \boldsymbol{\mathcal{A}}^{(1)}(:, i_1, :) \cdots \boldsymbol{\mathcal{A}}^{(d)}(:, i_d, :). \tag{2}$$

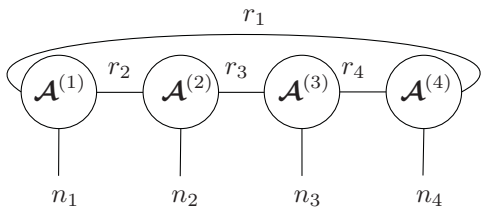

Fig. 3: Tensor train decomposition of a 4-way tensor $\boldsymbol{\mathcal{A}}$ into 3-way tensors $\boldsymbol{\mathcal{A}}^{(1)}, \ldots, \boldsymbol{\mathcal{A}}^{(4)}$.

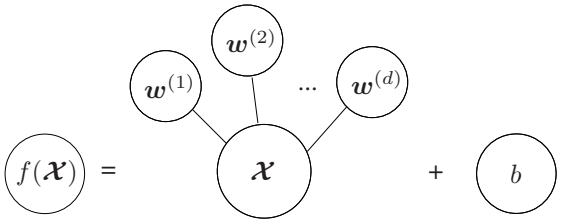

Fig. 4: Graphical representation of an STM hyperplane function.

Each tensor $\boldsymbol{\mathcal{A}}^{(k)}$, $k = 1, \ldots, d$, is called a TT-core and has dimensions $r_k \times n_k \times r_{k+1}$. Storage of a tensor as a TT therefore reduces from $\prod_{i=1}^{d} n_i$ down to $\sum_{i=1}^{d} r_i n_i r_{i+1}$. In order for the left-hand-side of (2) to be a scalar we require that $r_1 = r_{d+1} = 1$. The remaining r_k values are called the TT-ranks. Figure 3 illustrates the TT-decomposition of a 4-way tensor $\boldsymbol{\mathcal{A}}$, where the edges connecting the different circles indicate the matrix-matrix products of (2).

Definition 6: (Left orthogonal and right orthogonal TT-cores) A TT-core $\boldsymbol{\mathcal{A}}^{(k)} (1 \leq k \leq d)$ is left orthogonal when reshaped into an $r_k n_k \times r_{k+1}$ matrix $\boldsymbol{A}$ we have that

$$\boldsymbol{A}^T \boldsymbol{A} = \boldsymbol{I}_{r_{k+1}}.$$

Similarly, a TT-core $\boldsymbol{\mathcal{A}}^{(k)}$ is right orthogonal when reshaped into an $r_k \times n_k r_{k+1}$ matrix $\boldsymbol{A}$ we have that

$$\boldsymbol{A}\boldsymbol{A}^T = \boldsymbol{I}_{r_k}.$$

Definition 7: (Site-k-mixed-canonical tensor train) A tensor train is in site-k-mixed-canonical form [19] when all TT-cores $\{\boldsymbol{\mathcal{A}}^{(l)} \,|\, l = 1, \ldots, k-1\}$ are left orthogonal and $\{\boldsymbol{\mathcal{A}}^{(l)} \,|\, l = k+1, \ldots, d\}$ are right orthogonal.
Turning a TT into its site-k-mixed-canonical form requires $d-1$ QR decompositions of the reshaped TT-cores. Changing k in a site-k-mixed-canonical form to either $k-1$ or $k+1$ requires one QR factorization of $\boldsymbol{\mathcal{A}}^{(k)}$. It can be shown that the Frobenius norm of a tensor $\boldsymbol{\mathcal{A}}$ in a site-k-mixed-canonical form is easily computed from

$$||\boldsymbol{\mathcal{A}}||_F^2 = ||\boldsymbol{\mathcal{A}}^{(k)}||_F^2 = \text{vec}(\boldsymbol{\mathcal{A}}^{(k)})^T \text{vec}(\boldsymbol{\mathcal{A}}^{(k)}).$$

C. Support Vector Machines

We briefly introduce linear SVMs before discussing STMs. Assume we have a dataset D=$\{\boldsymbol{x_i}, y_i\}_{i=1}^{M}$ of M labeled samples, where $\boldsymbol{x}_i \in \mathbb{R}^n$ are the samples or feature vectors with labels $y_i \in \{-1, 1\}$. Learning a linear SVM is finding a discriminant hyperplane

$$f(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + b \tag{3}$$

that maximizes the margin between the two classes where $\boldsymbol{w}$ and b are the weight vector and bias, respectively. In practice, the data are seldom linearly separable due to measurement noise. A more robust classifier can then be found by introducing the slack variables $\xi_1, \ldots, \xi_M$ and writing the learning problem as an optimization problem

$$\begin{aligned} \min_{\boldsymbol{w}, b, \xi} \quad & \frac{1}{2}||\boldsymbol{w}||_F^2 + C\sum_{i=1}^{M} \xi_i \\ \text{subject to} \quad & y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0,\ i = 1, \ldots, M. \end{aligned} \tag{4}$$

The parameter C controls the trade-off between the size of the weight vector $\boldsymbol{w}$ and the size of the slack variables. It is common to solve the dual problem of (4) with quadratic programming, especially when the feature size n is larger than the sample size M.

D. Support Tensor Machines

Suppose the input samples in the dataset D=$\{\boldsymbol{\mathcal{X}}_i, y_i\}_{i=1}^{M}$ are tensors $\boldsymbol{\mathcal{X}}_i \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$. A linear STM extends a linear SVM by defining d weight vectors $\boldsymbol{w}^{(i)} \in \mathbb{R}^{n_i}$ $(i = 1, \ldots, d)$ and rewriting (3) as

$$f(\boldsymbol{\mathcal{X}}) = \boldsymbol{\mathcal{X}} \times_1 \boldsymbol{w}^{(1)} \times_2 \cdots \times_d \boldsymbol{w}^{(d)} + b. \tag{5}$$

The graphical representation of (5) is shown in Figure 4. The tensor $\boldsymbol{\mathcal{X}}$ is contracted along each of its modes with the weight vectors $\boldsymbol{w}^{(1)}, \ldots, \boldsymbol{w}^{(d)}$, resulting in a scalar that is added to the bias b. The weight vectors of the STM are computed by the alternating projection optimization procedure, which comprises d optimization problems. The main idea is to optimize each $\boldsymbol{w}^{(k)}$ in turn by fixing all weight vectors but $\boldsymbol{w}^{(k)}$. The kth optimization problem is

$$\begin{aligned} \min_{\boldsymbol{w}^{(k)}, b, \xi} \quad & \frac{1}{2}\,\beta\,||\boldsymbol{w}^{(k)}||_F^2 + C\sum_{i=1}^{M} \xi_i \\ \text{subject to} \quad & y_i((\boldsymbol{w}^{(k)})^T \hat{\boldsymbol{x}}_i + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0,\ i = 1, \ldots, M, \end{aligned} \tag{6}$$

where

$$\beta = \prod_{1 \leq l \leq d}^{l \neq k} ||\boldsymbol{w}^{(l)}||_F^2 \quad \text{and} \quad \hat{\boldsymbol{x}}_i = \boldsymbol{\mathcal{X}}_i \prod_{1 \leq l \leq d}^{l \neq k} \times_l \boldsymbol{w}^{(l)}.$$

The optimization problem (6) is equivalent to (4) for the linear SVM problem. This implies that any SVM learning algorithm can also be used for the linear STM. Each of the weight vectors of the linear STM is updated consecutively until the loss function of (6) converges. The convergence proof can be found in [13, p. 14]. Each single optimization problem in learning an STM requires the estimation of only a few weight

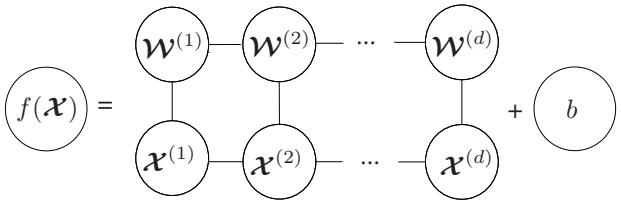

Fig. 5: Tensor graphical representation of an STTM hyperplane function.

parameters, which alleviates the overfitting problem when M is relatively small. The weight tensor obtained from the outer product of the weight vectors

$$\mathcal{W} = \boldsymbol{w}^{(1)} \circ \boldsymbol{w}^{(2)} \circ \cdots \circ \boldsymbol{w}^{(d)} \tag{7}$$

is per definition rank-1 and allows us to rewrite (5) as

$$f(\mathcal{X}) = \langle \mathcal{W}, \mathcal{X} \rangle + b. \tag{8}$$

The constraint that $\mathcal{W}$ is a rank-1 tensor has a significant impact on the expressive power of the STM, resulting in an usually unsatisfactory classification accuracy for many real-world data. In this paper, we address this problem by representing $\mathcal{W}$ as a TT with prescribed TT-ranks.

III. Support Tensor Train Machines

A. Linear Support Tensor Train Machines

We first introduce the proposed STTM for binary classification, and then extend it to the multi-classification case. The graphical representation for tensors shown in Figure 1 will be used to illustrate the different operations. As mentioned in Section II-D, an STM suffers from its weak expressive power due to its rank-1 weight tensor $\mathcal{W}$. To this end, the proposed STTM replaces the rank-1 weight tensor by a TT with prescribed TT-ranks. Moreover, most real-world data contains redundancies and uninformative parts. Based on this knowledge, STTM also utilizes a TT decomposition to approximate the original data tensor as to alleviate the overfitting problem even further. The conversion of the training sample to a TT can be done using the TT-SVD algorithm [18, p. 2301], which allows the user to determine the relative error of the approximation. A graphical representation of the STTM hyperplane equation is shown in Figure 5. Both the data tensor $\mathcal{X}$ and the weight tensor $\mathcal{W}$ are represented by TTs and the summations correspond to computing the inner product $\langle \mathcal{X}, \mathcal{W} \rangle$. The TT-cores $\mathcal{W}^{(1)}$, $\mathcal{W}^{(2)}$, ..., $\mathcal{W}^{(d)}$ are also computed using an alternating projection optimization procedure [11], namely iteratively fixing $d-1$ TT-cores and updating the remaining core until convergence. This updating occurs in a "sweeping" fashion, whereby we first update $\mathcal{W}^{(1)}$ and proceed towards $\mathcal{W}^{(d)}$. Once the core $\mathcal{W}^{(d)}$ is updated, the algorithm sweeps back to $\mathcal{W}^{(1)}$ and repeats this procedure until meets the termination criterion. Suppose we want to update $\mathcal{W}^{(k)}$. First, the TT of the weight tensor $\mathcal{W}$ is brought into site-k-mixed-canonical form. From Section II-B2, the norm of the whole weight tensor is located in the $\mathcal{W}^{(k)}$ TT-core. In order to reformulate the optimization problem (6) in terms of the unknown core $\mathcal{W}^{(k)}$, we first need to re-express the inner product $\langle \mathcal{X}, \mathcal{W} \rangle$ in terms of $\mathcal{W}^{(k)}$ as $\text{vec}(\mathcal{W}^{(k)})^T \hat{\boldsymbol{x}}$. The vector $\hat{\boldsymbol{x}}$ is obtained by summing over the tensor network for $\langle \mathcal{W}, \mathcal{X} \rangle$ depicted in Figure 5 with the TT-core $\mathcal{W}^{(k)}$ removed and vectorizing the resulting 3-way tensor. These two computational steps to compute $\hat{\boldsymbol{x}}$ are graphically depicted in Figure 6. The STTM hyperplane function can then be rewritten as $\text{vec}(\mathcal{W}^{(k)})^T \hat{\boldsymbol{x}} + b$, so that $\mathcal{W}^{(k)}$ can be updated from the following optimization problem

$$\begin{aligned} \min_{\mathcal{W}^{(k)}, b, \xi} \quad & \frac{1}{2} ||\mathcal{W}^{(k)}||_F^2 + C \sum_{i=1}^{M} \xi_i \\ \text{subject to} \quad & y_i (\text{vec}(\mathcal{W}^{(k)})^T \hat{\boldsymbol{x}}_i + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0, \ i = 1, \ldots, M, \end{aligned} \tag{9}$$

using any computational method for standard SVMs. Suppose now that the next TT-core to be updated is $\mathcal{W}^{(k+1)}$. The new TT for $\mathcal{W}$ then needs to be put into site-$(k+1)$-mixed-canonical form, which can be achieved by reshaping the new $\mathcal{W}^{(k)}$ into an $r_k n_k \times r_{k+1}$ matrix $\boldsymbol{W}^{(k)}$ and computing its thin QR decomposition

$$\boldsymbol{W}^{(k)} = \boldsymbol{Q} \, \boldsymbol{R},$$

where $\boldsymbol{Q}$ is a $r_k n_k \times r_{k+1}$ matrix with orthogonal columns and $\boldsymbol{R}$ is an $r_{k+1} \times r_{k+1}$ upper triangular matrix. Updating the tensors $\mathcal{W}^{(k)}, \mathcal{W}^{(k+1)}$ as

$$\begin{aligned} \mathcal{W}^{(k)} &:= \text{reshape}(\boldsymbol{Q}, [r_k, n_k, r_{k+1}]), \\ \mathcal{W}^{(k+1)} &:= \mathcal{W}^{(k+1)} \times_1 \boldsymbol{R}, \end{aligned}$$

results in a site-$(k+1)$-mixed-canonical form for $\mathcal{W}$. An optimization problem similar to (9) can then be derived for $\mathcal{W}^{(k+1)}$.

The training algorithm of the STTM is summarized as pseudo-codes in Algorithm 1. The TT-cores for the weight tensor $\mathcal{W}$ are initialized randomly. Bringing this TT into site-1-mixed-canonical form can then be done by applying the QR decomposition step starting from $\mathcal{W}^{(d)}$ and proceeding towards $\mathcal{W}^{(2)}$. The final $\boldsymbol{R}$ factor is absorbed by $\mathcal{W}^{(1)}$, which brings the TT into site-1-mixed-canonical form. The termination criterion in line 4 can be a maximum number of loops and/or when the training error falls below a user-defined threshold. To extend the binary classification STTM to an L-class classification STTM, we employ the one-versus-one strategy due to accuracy considerations [20]. Specifically, we construct $L(L-1)/2$ binary classification STTMs, where each STTM is trained on data samples from two classes. The label of a test sample is then predicted by a majority voting strategy.

B. Nonlinear Support Tensor Train Machines

The extension from a linear STTM to a nonlinear STTM is straightforward and we succinctly describe it here. The nonlinearity of the SVM is introduced through feature mapping

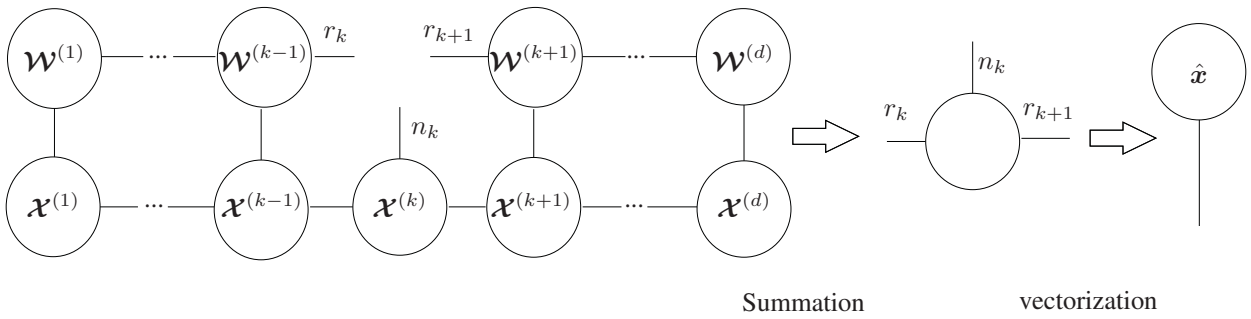

Fig. 6: The computation diagram of $\hat{\boldsymbol{x}}$.

Algorithm 1 STTM Algorithm

Input: TT-ranks $r_2, \ldots, r_d$ of $\mathcal{W}^{(1)}, \mathcal{W}^{(2)}, \ldots, \mathcal{W}^{(d)}$; Training dataset $\{\mathcal{X}_i \in \mathbb{R}^{n_1 \times \cdots \times n_d}, y_i \in \{-1, 1\}\}_{i=1}^{M}$; Relative error ϵ of TT approximation of $\mathcal{X}$.

Output: The TT-cores $\mathcal{W}^{(1)}, \mathcal{W}^{(2)}, \ldots, \mathcal{W}^{(d)}$; The bias b.

Initialize $\mathcal{W}^{(k)} \in \mathbb{R}^{r_k \times n_k \times r_{k+1}}$ as a random/prescribed 3-way tensor for $k = 1, 2, \ldots, d$.
Compute the TT approximation of training samples $\{\mathcal{X}_i\}_{i=1}^{M}$ with relative error ϵ using TT-SVD.
Cast $\mathcal{W}$ into the site-1-mixed-canonical TT form.
while termination criterion not satisfied **do**
  for $k = 1, \ldots, d-1$ **do**
    $\mathcal{W}^{(k)}, b \leftarrow$ Solve optimization problem (9).
    $\boldsymbol{W}^{(k)} \leftarrow \text{reshape}(\mathcal{W}^{(k)}, [r_k n_k, r_{k+1}])$.
    Compute thin QR decomposition $\boldsymbol{W}^{(k)} = \boldsymbol{Q}\boldsymbol{R}$.
    $\mathcal{W}^{(k)} \leftarrow \text{reshape}(\boldsymbol{Q}, [r_k, n_k, r_{k+1}])$.
    $\mathcal{W}^{(k+1)} \leftarrow \mathcal{W}^{(k+1)} \times_1 \boldsymbol{R}$.
  end for
  for $k = d, \ldots, 2$ **do**
    $\mathcal{W}^{(k)}, b \leftarrow$ Solve optimization problem (9).
    $\boldsymbol{W}^{(k)} \leftarrow \text{reshape}(\mathcal{W}^{(k)}, [r_k, n_k r_{k+1}])$.
    Compute thin QR decomposition $\boldsymbol{W}^{(k)^T} = \boldsymbol{Q}\boldsymbol{R}$.
    $\mathcal{W}^{(k)} \leftarrow \text{reshape}(\boldsymbol{Q}^T, [r_k, n_k, r_{k+1}])$.
    $\mathcal{W}^{(k-1)} \leftarrow \mathcal{W}^{(k-1)} \times_3 \boldsymbol{R}^T$.
  end for
end while

$\phi(\boldsymbol{x})$. The dual format of the original SVM is

$$\begin{aligned} \min_{\alpha_1, \alpha_2, \cdots, \alpha_M} \quad & \sum_{i=1}^{M} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{M} \alpha_i \alpha_j y_i y_j \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle \\ \text{subject to} \quad & \sum_{i=1}^{M} \alpha_i y_i = 0, \\ & 0 \leq \alpha_i \leq C, i = 1, \ldots, M, \end{aligned} \tag{10}$$

where α_i are the Lagrangian multipliers. When the dual problem (10) is solved, the model hyperplane parameters can be represented as:

$$\boldsymbol{w} = \sum_{i}^{M} \alpha_i y_i \phi(\boldsymbol{x}_i). \tag{11}$$

We can then derive the resulting hyperplane function as

$$f(\boldsymbol{x}) = \sum_{i}^{M} \alpha_i y_i \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}) \rangle + b. \tag{12}$$

It is easily noticed that all input samples appear in the inner product format in equation (12). Therefore, we can introduce the kernel trick so that the model of kernel SVM reads

$$f(\boldsymbol{x}) = \sum_{i}^{M} \alpha_i y_i \boldsymbol{k}(\phi(\boldsymbol{x}_i), \phi(\boldsymbol{x})) + b, \tag{13}$$

where $\boldsymbol{k}()$ denotes the kernel function, which can be Gaussian RBF kernel, polynomial kernel etc. Based on the kernel SVM formulation, the nonlinear STTM is then introduced as follows. We first construct two new vectors, namely,

$$\begin{aligned} \bar{\boldsymbol{w}} &= [\alpha_1, \alpha_2, \ldots, \alpha_M]^T \in \mathbb{R}^M \\ \bar{\boldsymbol{x}} &= [y_1 \boldsymbol{k}(\phi(\boldsymbol{x}_1), \phi(\boldsymbol{x})), \ldots, y_M \boldsymbol{k}(\phi(\boldsymbol{x}_M), \phi(\boldsymbol{x}))]^T \in \mathbb{R}^M. \end{aligned} \tag{14}$$

In that case, equation (13) can be reformulated as

$$f(\bar{\boldsymbol{x}}) = \bar{\boldsymbol{w}}^T \bar{\boldsymbol{x}} + b, \tag{15}$$

which can be regarded as a new linear SVM problem and we can then utilize the linear STTM method for finding the solution with similar procedures as in solving equation (3).

C. Complexity Analysis

We consider the linear STTM here only since the analysis for the nonlinear case is similar. Assume that the tensorial training data D=$\{\mathcal{X}_i\ y_i\}_{i=1}^{M}$ are given, where tensors $\mathcal{X}_i \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ are in TT format and their ranks are $r_1, \ldots, r_d$. With $n := \max\{n_1, \ldots, n_d\}$ and $r := \max\{r_1, \ldots, r_d\}$, the computation complexity of forming the small-size SVM optimization problem (9) from the overall STTM optimization problem is $\mathcal{O}(Mdnr^3)$. The complexity is linear to the tensorial data order d due to the TT structure. Moreover, real-world tensorial data often exhibit the low-rank property, namely

r is often small, which indicates the overall complexity is also small. For data storage, the traditional SVM calls for $\mathcal{O}(Mn^d)$ space, while that of the STTM is $\mathcal{O}(Mnr^2)$. This again shows a great reduction especially when the data order d is large.

IV. Experiments

Since STM and STuM are both linear classifiers and no kernel trick is introduced, here we only compare linear STTM with them for fairness. Specifically, we present three experiments that show the superiority of the proposed linear STTM over standard SVM, STM and STuM in terms of classification accuracy. All experiments are implemented in MATLAB on an Intel i5 3.2GHz desktop with 16GB RAM. Note that STM, STuM and STTM can separate their overall optimization problem into d standard SVM problems. We employ the MATLAB built-in SVM solver `fitcsvm` to get the solution for standard SVM, STM and STTM, while use the public code* released by the authors of [12] for STuM. When calling `fitcsvm`, we select a linear kernel with default parameters and set the outlier fraction as 2% for all experiments.

A. CIFAR-10 Binary Classification

Here we demonstrate three different aspects of the proposed STTM: a comparison of its test accuracy versus SVMs, STMs and STuMs, the influence of the TT-rank on the test accuracy, and the necessity of using the site-k-mixed-canonical form in Algorithm 1.

1) Classification: The CIFAR-10 database [21] is used in this binary classification experiment, which consists of 60k 32×32 color images from 10 classes, with 6000 images each. The `airplane` and `automobile` classes were arbitrarily chosen to compare the test accuracy of the proposed STTM with SVM, STM and STuM. The first 3000 samples of both classes were used for training while the test sets were used to check the model classification performance. Vectorizing the data samples results in a feature dimension of 3072, which may lead to overfitting when the training sample size is much smaller. To verify the effectiveness of STTM with different number of training samples, we divided the 3000 training samples into 30 experiments of varying sample batch sizes, namely 100, 200,..., 2900, 3000. For each batch size we trained a standard SVM, STM, STuM and STTM. The dimensions of the weight Tucker core in STuM are set as all 3 due to the its code constraint. Prior to training the STTM, each data sample was converted into a TT of 3 TT-cores with dimensions $n_1 = n_2 = 32, n_3 = 3$ and $\epsilon = 10^{-2}$. The TT-ranks of the weight TT were fixed to $r_1 = r_4 = 1, r_3 = 3$ and different experiment runs were performed where r_2 varied from 2 to 32. The best r_2 are chosen on a validation set. The resultant test accuracy of STTM are compared with the test accuracy of the SVM, STM and STuM subject to different training sample sizes in Figure 7. It is easily noticed that STTM almost always achieves the best test accuracy in all sample sizes, while STM sometimes performs worse than a standard SVM, especially when the batch size is below 400. The limitation on the performance of the STM is probably due to the poor expressive power of the rank-1 weight tensor. A batch size of 500 samples suffices for the STTM to achieve the best test accuracy of the standard SVM over all sample sizes, which demonstrates the superiority of STTM at fewer training samples.

*http://www.eecs.qmul.ac.uk/~ioannisp/source.htm

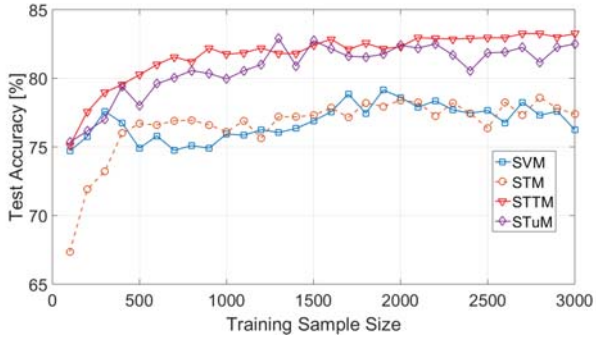

Fig. 7: Test accuracy of SVM, STM, STuM and STTM trained with different sample sizes.

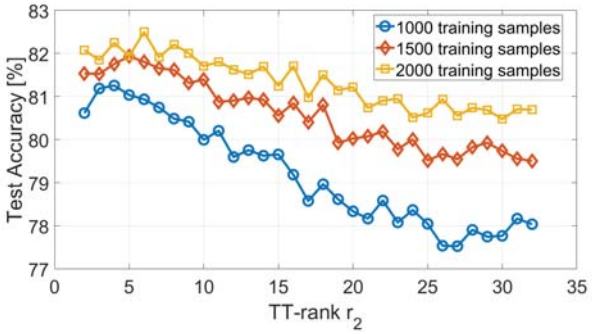

Fig. 8: Test accuracy of STTM on different TT-rank r_2.

2) Effect of TT-Rank on Test Accuracy: Figure 8 shows the STTM test accuracy for all tested 31 TT-ranks when the training batch size is equal to 1k, 1.5k and 2k, respectively. To accommodate for the effect of random initialization, the average test accuracy is presented over five different runs. The maximal test accuracy for these three sizes are achieved when r_2 is 4, 5 and 6, respectively. A downward trend of all three curves can be observed for TT-ranks larger than the optimal value, indicating that higher TT-ranks may lead to overfitting. On the other hand, decreasing the TT-rank from its optimal value also decreases the test accuracy down to the STM case. An extra validation step to determine the optimal TT-ranks is therefore highly recommended. It can also be observed that the overall test accuracy improves with an increasing sample size.

3) Updating in Site-k-Mixed-Canonical Form: The effect of keeping the TT of $\mathcal{W}$ in a site-k-mixed-canonical form when updating $\mathcal{W}^{(k)}$ is also investigated. Figure 9 shows the training accuracy for each TT-core update iteration in Algorithm 1, with and without the site-k-mixed-canonical form. Updating without the site-k-mixed-canonical form implies that lines 3, 8-10 and 15-17 of Algorithm 1 are not executed, which results in an

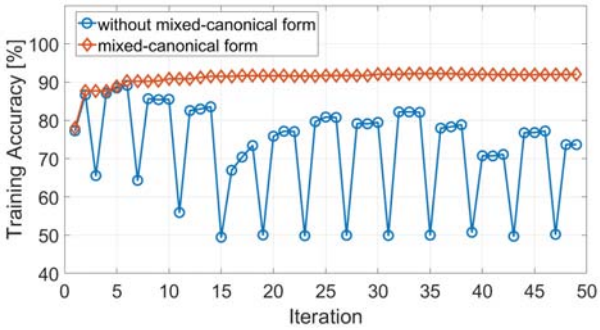

Fig. 9: Comparison of training accuracy of STTMs trained with and without site-k-mixed-canonical form.

oscillatory training accuracy ranging between 50% and 89% without any overall convergence. Updating the TT-cores $\mathcal{W}^{(k)}$ in a site-k-mixed-canonical form, however, displays a very fast convergence of the training accuracy to around 92%.

B. MNIST Multi-Classification

Next, the classification accuracy of a standard SVM, STM, STuM and STTM are compared on the MNIST dataset [22], which has a training set of 60k samples, and a testing set of 10k samples. Each sample is a 28×28 grayscale picture of a handwritten digit $\{0, \ldots, 9\}$. Even though the sample structure is a 2-way tensor, we opt to reshape each sample into a $7 \times 4 \times 7 \times 4$ tensor, as this provides us with more flexibility to choose TT-ranks when applying Algorithm 1. Since $10(10-1)/2 = 45$ binary classifiers need to be trained for this multi-classification problem, the weight vector obtained from the standard SVM is used to initialize the STM, STuM and STTM methods. For the STM initialization, the SVM weight vector is reshaped into a 28×28 matrix from which the best rank-1 approximation is used. For the STuM and STTM initialization, the SVM weight vector is reshaped into a $7 \times 4 \times 7 \times 4$ tensor and then converted into its Tucker and TT, respectively, with prescribed tensor ranks. Table I shows the experiment setting for those four methods. All classifiers were trained for training sample batch sizes of 10k, 20k, 30k and 60k in four different experiments. The test accuracy of the different methods for different batch sizes are listed in Table II. STTM achieves the best classification performance for all sizes. The STM again performs worse than the standard SVM due to the restrictive expressive power of the rank-1 weight matrix. Though STuM is one of the generalization formats of STM, it suffers from the curse of dimensionality due to its Tucker tensor model structure. The training procedure of STuM costs more than 7.5hrs and 59hrs when the training sample sizes are 10k and 20k, respectively, while only seconds or minutes for SVM, STM and STTM. We do not post the test accuracy of STuM when the training sample sizes are 30k and 60k since they cost much more time than other three methods. This observation indicates that an STuM may not work well when the training sample size is large due to its exponentially large model size.

TABLE I: Experiment settings for the four methods.

Method	Input Structure	Tensor ranks
SVM	784×1 vector	NA
STM	28×28 matrix	1
STuM	$7 \times 4 \times 7 \times 4$ tensor	$4, 4, 4, 4$
STTM	$7 \times 4 \times 7 \times 4$ tensor	$1, 5, 5, 4, 1$

TABLE II: Test accuracy (%) under different training sample sizes.

Method	Training Sample Size			
	10k	20k	30k	60k
SVM	91.64	92.84	93.28	93.99
STM	88.36	89.96	89.82	90.54
STuM	90.45	92.28	–	–
STTM	92.27	93.71	93.86	94.12

C. ORL Multi-Classification

In this experiment, the classification accuracy of a standard SVM, STM, STuM and STTM are compared on the ORL database[†]. ORL database contains 400 grayscale face images, and the detailed information about ORL datasets is listed in Table III. We randomly choose 320 face pictures as training data and the left 80 as testing data. To solve this multi-classification problem, 40(40-1)/2= 780 binary classifiers are needed to be trained by using each method. The parameter tensors of STM, STuM and STTM are initialized randomly according to their preset tensor ranks. The detailed experiment settings and classification accuracy (average value of five repeated tests) for ORL32x32 and ORL64x64 when employing different methods are listed in Table IV and Table V. STTM achieves a similar classification performance compared with that of SVM and they both perform better than STM and STuM.

TABLE III: Detailed information of experimental datasets.

Datasets	Number of samples	Number of classes	Size
ORL 32x32	400	40	32x32
ORL 64x64	400	40	64x64

TABLE IV: Experimental settings and classification accuracy (%) of four methods for ORL32x32.

Method	Input structure	Tensor ranks	Test accuracy
SVM	1024x1 vector	NA	96.25
STM	32x32 matrix	1	93.75
STuM	8x4x8x4 tensor	$4, 4, 4, 4$	93.50
STTM	8x4x8x4 tensor	$1, 4, 4, 4, 1$	96.25

TABLE V: Experimental settings and classification accuracy (%) of four methods for ORL64x64.

Method	Input structure	Tensor ranks	Test accuracy
SVM	4096x1 vector	NA	96.25
STM	64x64 matrix	1	92.71
STuM	8x8x8x8 tensor	$4, 4, 4, 4$	94.40
STTM	8x8x8x8 tensor	$1, 4, 4, 4, 1$	96.25

[†]http://www.zjucadcg.cn/dengcai/Data/FaceData.html

V. Conclusions

We have proposed, for the first time, a support tensor train machine (STTM) for classifier design. On the one hand, STTM employs a more general tensor train structure to largely escalate the model expressive power, which leads to a better classification accuracy than STM. On the other hand, the tensor model in STTM is more scalable than in STuM, which achieves a faster training when the training sample size is large. Experiments have demonstrated the superiority of STTM over standard SVM, STM and STuM in terms of classification accuracy, particularly when trained with small sample sizes.

References

[1] V. Vapnik, *The nature of statistical learning theory*. Springer science & business media, 2013.

[2] J. Li, N. Allinson, D. Tao, and X. Li, "Multitraining support vector machine for image retrieval," *IEEE Transactions on Image Processing*, vol. 15, no. 11, pp. 3597–3601, 2006.

[3] D. Tao, X. Tang, X. Li, and X. Wu, "Asymmetric bagging and random subspace for support vector machines-based relevance feedback in image retrieval," *IEEE transactions on pattern analysis and machine intelligence*, vol. 28, no. 7, pp. 1088–1099, 2006.

[4] S. Yan, D. Xu, Q. Yang, L. Zhang, X. Tang, and H.-J. Zhang, "Multilinear discriminant analysis for face recognition," *IEEE Transactions on Image Processing*, vol. 16, no. 1, pp. 212–220, 2007.

[5] Z. Chen, K. Batselier, J. A. Suykens, and N. Wong, "Parallelized tensor train learning of polynomial classifiers," *IEEE Transactions on Neural Networks and Learning Systems*, 2017.

[6] A. Novikov, D. Podoprikhin, A. Osokin, and D. P. Vetrov, "Tensorizing neural networks," in *Advances in Neural Information Processing Systems*, 2015, pp. 442–450.

[7] A. Novikov, M. Trofimov, and I. Oseledets, "Exponential machines," *arXiv preprint arXiv:1605.03795*, 2016.

[8] E. Stoudenmire and D. J. Schwab, "Supervised learning with tensor networks," in *Advances in Neural Information Processing Systems 29*. Curran Associates, Inc., 2016, pp. 4799–4807.

[9] V. Lebedev, Y. Ganin, M. Rakhuba, I. Oseledets, and V. Lempitsky, "Speeding-up convolutional neural networks using fine-tuned cp-decomposition," *arXiv preprint arXiv:1412.6553*, 2014.

[10] M. Signoretto, Q. T. Dinh, L. De Lathauwer, and J. A. Suykens, "Learning with tensors: a framework based on convex optimization and spectral regularization," *Machine Learning*, vol. 94, no. 3, pp. 303–351, 2014.

[11] D. Tao, X. Li, W. Hu, S. Maybank, and X. Wu, "Supervised tensor learning," in *Data Mining, Fifth IEEE International Conference on*. IEEE, 2005, pp. 8–pp.

[12] I. Kotsia and I. Patras, "Support Tucker Machines," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE, 2011, pp. 633–640.

[13] D. Tao, X. Li, X. Wu, W. Hu, and S. J. Maybank, "Supervised tensor learning," *Knowledge and Information Systems*, vol. 13, no. 1, pp. 1–42, Sep 2007.

[14] I. Kotsia, W. Guo, and I. Patras, "Higher rank support tensor machines for visual recognition," *Pattern Recognition*, vol. 45, no. 12, pp. 4192–4203, 2012.

[15] J. Håstad, "Tensor rank is NP-complete," *Journal of Algorithms*, vol. 11, no. 4, pp. 644–654, 1990.

[16] Y. Wang, W. Zhang, Z. Yu, Z. Gu, H. Liu, Z. Cai, C. Wang, and S. Gao, "Support vector machine based on low-rank tensor train decomposition for big data applications," in *2017 12th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, June 2017, pp. 850–853.

[17] R. Orús, "A practical introduction to tensor networks: Matrix product states and projected entangled pair states," *Annals of Physics*, vol. 349, pp. 117–158, 2014.

[18] I. V. Oseledets, "Tensor-train decomposition," *SIAM Journal on Scientific Computing*, vol. 33, no. 5, pp. 2295–2317, 2011.

[19] U. Schollwöck, "The density-matrix renormalization group in the age of matrix product states," *Annals of Physics*, vol. 326, no. 1, pp. 96 – 192, 2011, january 2011 Special Issue.

[20] C.-W. Hsu and C.-J. Lin, "A comparison of methods for multiclass support vector machines," *IEEE transactions on Neural Networks*, vol. 13, no. 2, pp. 415–425, 2002.

[21] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," 2009.

[22] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.