TENSOR-BASED NONLINEAR CLASSIFIER FOR HIGH-ORDER DATA ANALYSIS

Konstantinos Makantasis[1]*, Anastasios Doulamis*[2]*, Nikolaos Doulamis*[2]*,*
Antonis Nikitakis[3]*, and Athanasios Voulodimos*[2,4]

[1] KIOS Research and Innovation Center of Excellence, University of Cyprus, Nicosia, Cyprus
[2] National Technical University of Athens, Athens, Greece
[3] Althexis Solutions Ltd, Nicosia, Cyprus
[4] Department of Informatics, Technological Educational Institute of Athens, Athens, Greece

ABSTRACT

In this paper we propose a tensor-based nonlinear model for high-order data classification. The advantages of the proposed scheme are that (i) it significantly reduces the number of weight parameters, and hence of required training samples, and (ii) it retains the spatial structure of the input samples. The proposed model, called *Rank*-1 FNN, is based on a modification of a feedforward neural network (FNN), such that its weights satisfy the *rank*-1 canonical decomposition. We also introduce a new learning algorithm to train the model, and we evaluate the *Rank*-1 FNN on third-order hyperspectral data. Experimental results and comparisons indicate that the proposed model outperforms state of the art classification methods, including deep learning based ones, especially in cases with small numbers of available training samples.

Index Terms— Tensor-based classification, hyperspectral data, tensor data analysis, Rank-1 FNN

1. INTRODUCTION

Recent advances in sensing technologies have stimulated the development and deployment of sensors that can generate large amounts of high-order data. Interdependencies between information from different data modalities can improve the performance of data classification techniques [1]. However, exploitation of high-order data raises new research challenges mainly due to the high dimensionality of the acquired information and, depending on the application at hand, the limited number of labeled examples [2].

Tensor subspace learning methods, such as HOSVD, Tucker decomposition and CANDECOMP [3], MPCA [4] and probabilistic decompositions [5, 6, 7] have been proposed to tackle the dimensionality problem. These methods project the raw data to a lower dimensional space, in which the projected data can be considered as highly descriptive features of the raw information. The key problem in applying such methods in classifying high-order data, is that they do not take into consideration the data labels; therefore, the resulting features may not be sufficiently discriminative with respect to the classification task. Tensor-based classifiers capable of mapping high-order data to desired outputs have also been proposed [8, 9, 10, 11, 12]. However, these methods are restricted to producing linear decision boundaries in feature space, and are therefore unable to cope with complex problems, where nonlinear decision boundaries are necessary to obtain classification results of high accuracy. In order to better disentangle the input-output statistical relationships, deep learning approaches [13, 14] have been investigated for high-order data classification [15, 16, 17, 18]. Nevertheless, a typical deep learning architecture contains a huge number of tunable parameters, implying that a large number of labeled samples is also needed for accurate training.

The present work draws its inspiration from [9], which proposes a linear tensor regression model for binary classification. In contrast to [9], the paper at hand investigates a multi-class classification problem using a nonlinear tensor-based classifier. The proposed classifier is able to (i) handle raw high-order data without vectorizing them, and (ii) produce nonlinear decision boundaries, thus capturing complex statistical relationships between the data. The proposed scheme, henceforth called *Rank*-1 FNN, is based on a modification of a feedforward neural network (FNN), such that its weights satisfy the *rank*-1 canonical decomposition property, i.e., the weights are decomposed as a linear combination of a minimal number of possibly non-orthogonal *rank*-1 terms [19]. Thence, the number of model parameters, and thus of training samples required, can be significantly reduced. We also introduce a new learning algorithm to train the network without violating the canonical decomposition property.

2. PROBLEM FORMULATION AND TENSOR ALGEBRA NOTATION

2.1. Problem Formulation

Let us denote as $\boldsymbol{X}_i \in \mathbb{R}^{p_1 \times \cdots \times p_D}$ the i-th D-order tensor example that we aim at classifying into one of C classes.

This paper is supported by the European Union Project TERPSICHORE funded under grant agreement 691218.

Let us also denote as $p_w^k(\boldsymbol{X}_i)$ the probability of $\boldsymbol{X}_i$ belonging to the k-th class. Aggregating the values $p_w^k(\cdot)$ over all classes, we form a classification vector, $\boldsymbol{y}_i$, the elements of which $y_{i,k} \equiv p_w^k(\cdot)$. Then, the maximum $p_w^k(\cdot)$ value over all classes indicates the class to which the $\boldsymbol{X}_i$ belongs. The values of $y_{i,k}$ are estimated by minimizing a loss function over a dataset $\mathcal{S} = \{(\boldsymbol{X}_i, \boldsymbol{t}_i)\}_{i=1}^N$ during the training phase of a machine learning model. Vector $\boldsymbol{t}_i \in \{0,1\}^C$ and its elements $t_{i,j}$ are all zero except for one which equals unity indicating the class to which $\boldsymbol{X}_i$ belongs. In the following, we omit subscript i for simplicity purposes if we refer to an input sample.

2.2. Tensor Algebra Notations and Definitions

In this paper, tensors, vectors and scalars are denoted in bold uppercase, bold lowercase and lowercase letters, respectively. We hereby present some definitions that will be used through out this work.

Tensor vectorization. The $vec(\boldsymbol{B})$ operator stacks the entries of a D-order tensor $\boldsymbol{B} \in \mathbb{R}^{p_1 \times \cdots \times p_D}$ into a column vector.

Tensor matricization.The mode-d matricization, $\boldsymbol{B}_{(d)}$, maps a tensor $\boldsymbol{B}$ into a $p_d \times \prod_{d' \neq d} p_{d'}$ matrix by arranging the mode-d fibers to be the columns of the resulting matrix.

***Rank*-R decomposition**. A tensor $\boldsymbol{B} \in \mathbb{R}^{p_1 \times \cdots \times p_D}$ admits a *rank*-R decomposition if $\boldsymbol{B} = \sum_{r=1}^R \boldsymbol{b}_1^{(r)} \circ \cdots \circ \boldsymbol{b}_D^{(r)}$, where $\boldsymbol{b}_d^{(r)} \in \mathbb{R}^{p_d}$. The decomposition can be represented by $\boldsymbol{B} = [\![\boldsymbol{B}_1, ..., \boldsymbol{B}_D]\!]$, where $\boldsymbol{B}_d = [\boldsymbol{b}_d^{(1)}, ..., \boldsymbol{b}_d^{(R)}] \in \mathbb{R}^{p_d \times R}$. When a tensor $\boldsymbol{B}$ admits a *rank*-R decomposition, it holds that:

$$\boldsymbol{B}_{(d)} = \boldsymbol{B}_d(\boldsymbol{B}_D \odot \cdots \odot \boldsymbol{B}_{d+1} \odot \boldsymbol{B}_{d-1} \odot \cdots \odot \boldsymbol{B}_1)^T \quad (1)$$

where $\odot$ stands for the Khatri-Rao product. For more information on tensor algebra see [3].

3. HIGH-ORDER NONLINEAR MODELING

The proposed *Rank*-1 FNN is based on the concepts of [9]; however, in our case, the probability $p_w^k(\cdot)$ of an input example $\boldsymbol{X}$ belonging to the k-th class is nonlinearly interwoven with respect to the input tensor data and the weight parameters through a function $f_w(\cdot)$, i.e., $p_w^k(\boldsymbol{X}) = f_w(\boldsymbol{X})$. The main difficulty in implementing $p_w^k(\boldsymbol{X})$ is that $f_w(\cdot)$ is actually unknown. One way to parameterize $f_w(\cdot)$ is to exploit the principles of the universal approximation theorem, stating that a function can be approximated by a FNN with a finite number of neurons within any degree of accuracy.

However, applying a FNN for high-order data classification involves two drawbacks. First, a large number of weights has to be learned; $Q\prod_{l=1}^D p_l + QC$, where Q refers to the number of hidden neurons. This, in the sequel, implies that a large number of labeled samples are needed to successfully train the network. Second, the weights of the network are not directly related to the physical properties of the information belonging to different modes of the data, since the inputs are vectorized and thus they do not preserve their structure.

To overcome these problems, we propose a modification of FNN so that network weights from the input to the hidden layer satisfy the *rank*-1 canonical decomposition. Before presenting the *Rank*-1 FNN, we briefly describe how $p_w^k(\cdot)$ is modeled through a FNN.

3.1. FNN Modeling

A FNN, with Q hidden neurons, nonlinearly approximates the probability $p_w^k(\cdot)$ by associating a nonlinear activation function $g(\cdot)$ with each one of its hidden neurons. In this paper, the sigmoid function $g(x) = 1/(1+\exp(-ax))$ is selected. The activation function of the i-th neuron receives as input the inner product of $vec(\boldsymbol{X})$ and a weight vector $\boldsymbol{w}^{(i)}$ and produces as output a scalar u_i given by

$$u_i = g(\boldsymbol{w}^{(i)T} vec(X)) \equiv g(\langle \boldsymbol{w}^{(i)}, \boldsymbol{v}ec(X)\rangle). \quad (2)$$

Gathering the responses of all hidden neurons in one vector $\boldsymbol{u} = [u_1, u_2, \cdots, u_Q]^T$, we have that

$$\boldsymbol{u} = g(\langle \boldsymbol{W}, \boldsymbol{X}\rangle), \quad (3)$$

where $\boldsymbol{W} = [\boldsymbol{w}^{(1)}, \cdots, \boldsymbol{w}^{(Q)}]^T$ is a matrix containing the weights $\boldsymbol{w}^{(i)}$. Thus, the output of the network is given as

$$\boldsymbol{p}_w^k = \sigma(\langle \boldsymbol{v}^{(k)}, \boldsymbol{u}\rangle) \equiv \sigma(\boldsymbol{v}^{(k)T}\boldsymbol{u}), \quad (4)$$

where $\sigma(\cdot)$ stands for the softmax function, $\boldsymbol{v}^{(k)}$ the weights between the hidden and the output layer and the superscript for the k-th class.

3.2. *Rank*-1 FNN Modeling

To reduce the number of parameters of the network and to relate the classification results to the information belonging to different modes of the input data, we *rank*-1 canonically decompose the weight parameters $\boldsymbol{w}^{(i)}$ as:

$$\boldsymbol{w}^{(k)} = \boldsymbol{w}_D^{(k)} \otimes \cdots \otimes \boldsymbol{w}_1^{(k)} = \boldsymbol{w}_D^{(k)} \odot \cdots \odot \boldsymbol{w}_1^{(k)}. \quad (5)$$

Eq. (5) can be seen as an expression of the Khatri-Rao product, which is the column-wise Kronecker product, denoted as $\otimes$, of the *rank*-1 canonical decomposition weight parameters $\boldsymbol{w}_l^{(k)}$. Thus, $\boldsymbol{w}_l^{(k)} \in \mathbb{R}^{p_l}$ and the total number of *Rank*-1 FNN is $Q\sum_{l=1}^D p_l + QC$. Based on the statements of Section 2.2, it holds that

$$\begin{aligned}\langle \boldsymbol{w}_D^{(k)} \odot \cdots \odot \boldsymbol{w}_1^{(k)}, \boldsymbol{X}\rangle = \langle \boldsymbol{w}_l^{(k)}, \boldsymbol{X}_{(l)}(\boldsymbol{w}_D^{(k)} \odot \cdots \\ \odot \boldsymbol{w}_{l+1}^{(k)} \odot \boldsymbol{w}_{l-1}^{(k)} \odot \cdots \odot \boldsymbol{w}_1^{(k)})\rangle.\end{aligned} \quad (6)$$

In Eq. (6), $\boldsymbol{X}_{(l)}$ denotes the mode-l matricization of tensor $\boldsymbol{X}$.Then, taking into account the properties of Eq. (6), the output of the i-th hidden neuron u_i can be written as

$$\begin{aligned}u_i &= g(\langle \boldsymbol{w}^{(i)}, \boldsymbol{X}\rangle) = g(\langle \boldsymbol{w}_D^{(i)} \otimes \cdots \otimes \boldsymbol{w}_1^{(i)}, \boldsymbol{X}\rangle) \\ &= g(\langle \boldsymbol{w}_D^{(i)} \odot \cdots \odot \boldsymbol{w}_1^{(i)}, \boldsymbol{X}\rangle) = g(\langle \boldsymbol{w}_l^{(i)}, \boldsymbol{\tau}_{\neq l}^{(i)}\rangle).\end{aligned} \quad (7)$$

Vector $\boldsymbol{\tau}_{\neq l}^{(i)}$ is a transformed version of input $\boldsymbol{X}$, that is,

$$\boldsymbol{\tau}_{\neq l}^{(i)} = \boldsymbol{X}_{(l)}(\boldsymbol{w}_D^{(k)} \odot \cdots \odot \boldsymbol{w}_{l+1}^{(k)} \odot \boldsymbol{w}_{l-1}^{(k)} \odot \cdots \odot \boldsymbol{w}_1^{(k)}) \quad (8)$$

and is independent from $\boldsymbol{w}_l^{(i)}$. Eq. (7) actually resembles the operation of a single perceptron having as inputs the weights $\boldsymbol{w}_l^{(i)}$ and the transformed version $\tau_{\neq l}$ of the input data. In other words, if the *rank*-1 canonically decomposed weights $\boldsymbol{w}_r^{(i)}$ with $r \neq l$ are known, then $\tau_{\neq l}^{(i)}$ will be also known. The main modification of this structure compared to a typical FNN lies in the hidden layer, where the weights of a hidden neuron are first decomposed into D canonical factors.

3.3. The Learning Algorithm

Let us aggregate the total *Rank*-1 FNN weight parameters as

$$\boldsymbol{W}_l = [\boldsymbol{w}_l^{(1)}\boldsymbol{w}_l^{(2)} \cdots \boldsymbol{w}_l^{(Q)}], \boldsymbol{V} = [\boldsymbol{v}^{(1)}\boldsymbol{v}^{(2)} \cdots \boldsymbol{v}^{(C)}] \quad (9)$$

with $l = 1, 2, \cdots, D$. In order to train the proposed model a set $\mathcal{S} = \{(\boldsymbol{X}_i, \boldsymbol{t}_i)\}_{i=1}^N$ is used. The learning algorithm minimizes the negative log-likelihood

$$L(\boldsymbol{W}_1, ..., \boldsymbol{W}_D, \boldsymbol{V}; \mathcal{S}) = -\sum_{i=1}^{N}\sum_{k=1}^{C} t_{i,k} \log p_w^k(\boldsymbol{X}_i), \quad (10)$$

with respect to network responses $\boldsymbol{y}_i = [\cdots y_{i,k} \cdots]^T$, with $y_{i,k} \equiv p_w^k(\boldsymbol{X}_i)$, and targets $\boldsymbol{t}_i$ over all training samples.

The weights of the *Rank*-1 FNN must satisfy the *rank*-1 canonical decomposition expressed by Eq. (5). Assuming that all weights $\boldsymbol{V}$ and $\boldsymbol{w}_r^{(i)}$ with $r \neq l$ are fixed, vector $\boldsymbol{\tau}_{\neq l}^{(i)}$ can be estimated; therefore, vector $\boldsymbol{w}_l^{(i)}$ is the only unknown parameter of the network. This vector can be derived through a gradient based optimization algorithm, assuming that the derivative $\partial L/\partial \boldsymbol{w}_l^{(i)}$ is known. This derivative can be computed using the backpropagation algorithm. Therefore, an estimation of the parameters of the *Rank*-1 FNN is obtained by iteratively solving with respect to one of the D canonical decomposed weight vectors, assuming the remaining fixed. Algorithm 1 presents the steps of the proposed algorithm.

Algorithm 1: Estimation of of the *Rank*-1 FNN Weights

```
Initialization:
1. Set Iteration Index $n \to 0$
2. Randomize all the weight $\boldsymbol{w}_l^{(i)}(n)$ and $\boldsymbol{v}^{(k)}(n)$
for $l = 1, ..., D$, $i = 1, 2, \cdots, Q$, $k = 1, ..., C$
3. repeat
    for $l = 1, ..., D$ do
        for $i = 1, ...Q$ do
            3.1 Estimate the transformed input vector
              $\tau_{\neq l}^{(i)} = \boldsymbol{X}_{(l)}(\boldsymbol{w}_D^{(i)}(n) \odot \cdots \odot \boldsymbol{w}_{l+1}^{(i)}(n) \odot$
              $\boldsymbol{w}_{l-1}^{(i)}(n+1) \odot \cdots \odot \boldsymbol{w}_1^{(i)}(n+1))$,
            3.2 Update the weights $\boldsymbol{w}_l^{(i)}(n)$ towards the
              negative direction of $\partial L/\partial \boldsymbol{w}_l^{(i)}$
        end
    end
    for $k = 1, ..., C$ do
        3.3 Update the weights $\boldsymbol{v}^{(k)}(n)$ towards the
          negative direction of $\partial E/\partial \boldsymbol{v}^{(k)}$
    end
    Set $n \to n + 1$
until termination criteria are met;
```

4. EVALUATION ON HYPERSPECTRAL DATA

To investigate whether the reduced number of parameters would limit the descriptive power of the *Rank*-1 FNN, we conduct experiments and present quantitative results regarding its classification accuracy on 3-order hyperspectral data. In our study, we used (i) the Indian Pines dataset [20], which consists of 224 spectral bands and $10,086$ labeled pixels and (ii) the Pavia University dataset [21], consisting of 103 spectral bands and $42,776$ labeled pixels.

A hyperspectral image is represented as a 3-order tensor of dimensions $p_1 \times p_1 \times p_3$, where p_1 and p_2 correspond to the height and width of the image and p_3 to the spectral bands. In order to classify a pixel $I_{x,y}$ at location (x, y) on image plane and fuse spectral and spatial information, we use a square patch of size $s \times s$ centered at (x, y). Let us denote as $\boldsymbol{t}_{x,y}$ the label of $I_{x,y}$ and as $\boldsymbol{X}_{x,y}$ the tensor patch centered at (x, y). Then, we can form a dataset $S = \{(\boldsymbol{X}_{x,y}, \boldsymbol{t}_{x,y})\}$, which is used to train the classifier.

To evaluate the performance of the *Rank*-1 FNN, we conducted different experiments using training datasets of 50, 100, 150 and 200 samples from each class respectively. Initially, we evaluate the performance of the *Rank*-1 FNN with respect to its complexity, i.e., the value of Q, indicating the number of hidden neurons. Particularly, we set Q to be equal to 50, 75, 100 and 125. Greater values of Q imply a more complex model. The results of this evaluation are presented in Fig.1. Regarding the Indian Pines dataset, we observe that the model with $Q = 75$ outperforms all other models. When the training set size is very small, i.e. 50 samples per class, the model with $Q = 50$ underfits the data. On the other hand, the models with $Q = 100$ and $Q = 125$ slightly overfit the data due to their high complexity. As far as the Pavia University dataset is concerned, we observe that the model with $Q = 100$ outperforms all other models, when the dataset size is larger than 50 samples per class. When the training dataset size is 50 samples per class the model with $Q = 75$ outperforms all other models. The model with $Q = 125$ overfits the data, while the model with $Q = 50$ underfits them. In both datasets, as training set increases, the misclassification error decreases.

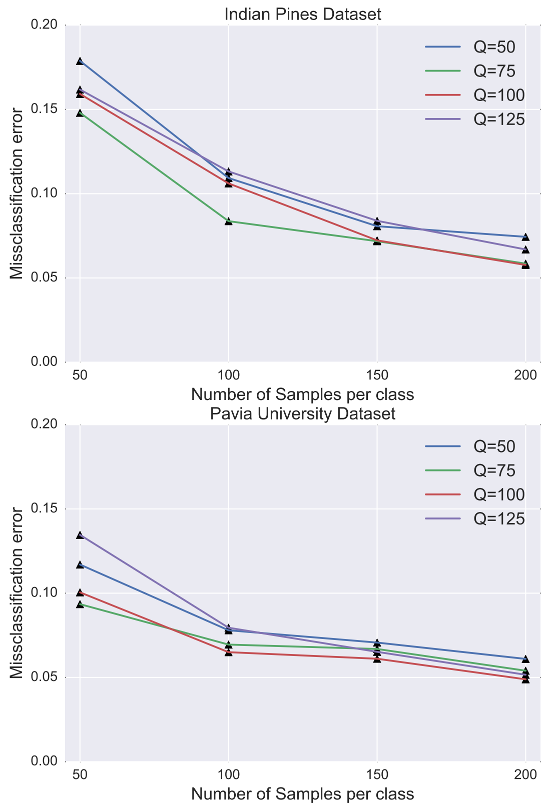

Fig. 1. Misclassification error on test set versus the complexity, determined by Q, of the *Rank*-1 FNN.

In the following, we compare the performance of *Rank*-1 FNN against FNN, RBF-SVM, and two deep learning approaches that have been proposed for classifying hyperspectral data; the first one is based on Stacked-Autoencoders (SAE) [15], while the second on the exploitation of Convolutional Neural Networks (CNN) [16]. The FNN consists of one hidden layer with 75 hidden neurons when trained on Indian Pines dataset and 100 hidden neurons for the Pavia University dataset (as derived from Fig.1). The architecture of the network based on SAE consists of three hidden layers, while each hidden layer contains 10% less neurons than its input. The number of hidden neurons from one hidden layer to the next is gradually reduced, so as not to allow the network to learn the identity function during pre-training. Regarding CNN, we utilize exactly the same architecture as the one presented in [16]. The performance of all these models is evaluated on varying size training sets.

Table 1 presents the outcome of this comparison. When the training set size is small, our approach outperforms all other models. This stems from the fact that the proposed *Rank*-1 FNN exploits tensor algebra operations to reduce the number of coefficients that need to be estimated during training, while at the same time it is able to retain the spatial structure of the input. Although the FNN utilizes the same number of hidden neurons as our proposed model, it seems to overfit training sets when a small size dataset is used, due to the fact that it employs a larger number of coefficients. RBF-SVM performs better than the FNN on the Pavia University dataset, but slightly worse on the Indian Pines dataset. The full connectivity property of SAE implies very high complexity, which is responsible for its poor performance, due to overfitting in the Indian Pines dataset. Finally, the CNN-based approach performs better than FNN, RBF-SVM and SAE mainly because of its sparse connectivity (low complexity) and the fact that it can exploit the spatial information of the input. When the training set consists of 150 and 200 samples per class, for the Pavia University dataset, and 200 samples for the Indian Pines dataset, the CNN-based approach seems to even outperform the *Rank*-1 FNN. This happens because the CNN-based model has higher capacity than the proposed model, which implies that it is capable of better capturing the statistical relationships between the input and the output, when the training set contains sufficient information. However, when the size of the training set is small, which is often the case, *Rank*-1 FNN, due to its lower complexity, consistently outperforms the CNN-based model.

Table 1. Classification accuracy results (%) of *Rank*-1 FNN

Pavia University				
Samples per class	50	100	150	200
Rank-1 FNN (Q=100)	**89.95**	**93.50**	93.89	95.11
FCFFNN	67.79	76.53	78.48	82.59
RBF-SVM	86.98	88.99	89.86	91.82
SAE	86.54	91.90	92.38	93.29
CNN	88.89	92.74	**94.68**	**95.89**
Indian Pines				
Samples per class	50	100	150	200
Rank-1 FNN (Q=75)	**85.20**	**91.63**	**92.82**	94.15
FCFFNN	73.88	81.10	84.14	85.86
RBF-SVM	73.18	77.86	82.11	84.99
SAE	65.51	70.66	74.03	76.49
CNN	82.43	85.48	92.28	**94.81**

5. CONCLUSIONS

In this work, we present a nonlinear tensor-based scheme for high-order data classification. The proposed model is characterized by (i) the small number of weight parameters and (ii) its ability to retain the spatial structure of the high-order input samples. We have evaluated the performance of the model on 3-order hyperspectral data in terms of classification accuracy by comparing it against other nonlinear classifiers, including state-of-the-art deep learning models. The results indicate that in cases where the size of the training set is small, the proposed *Rank*-1 FNN presents superior performance against the compared methods, including deep learning based ones.

6. REFERENCES

[1] Guoxu Zhou, Qibin Zhao, Yu Zhang, Tülay Adalı, Shengli Xie, and Andrzej Cichocki, "Linked component analysis from matrices to high-order tensors: Applications to biomedical data," *Proceedings of the IEEE*, vol. 104, no. 2, pp. 310–331, 2016.

[2] Gustavo Camps-Valls and Lorenzo Bruzzone, "Kernel-based methods for hyperspectral image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 43, no. 6, pp. 1351–1362, 2005.

[3] Tamara G Kolda and Brett W Bader, "Tensor decompositions and applications," *SIAM review*, vol. 51, no. 3, pp. 455–500, 2009.

[4] Haiping Lu, Konstantinos N Plataniotis, and Anastasios N Venetsanopoulos, "Mpca: Multilinear principal component analysis of tensor objects," *IEEE Transactions on Neural Networks*, vol. 19, no. 1, pp. 18–39, 2008.

[5] Piyush Rai, Yingjian Wang, Shengbo Guo, Gary Chen, David Dunson, and Lawrence Carin, "Scalable bayesian low-rank decomposition of incomplete multiway tensors," in *International Conference on Machine Learning*, 2014, pp. 1800–1808.

[6] Wei Chu and Zoubin Ghahramani, "Probabilistic models for incomplete multi-dimensional arrays," in *Artificial Intelligence and Statistics*, 2009, pp. 89–96.

[7] Zenglin Xu, Feng Yan, and Yuan Qi, "Bayesian nonparametric models for multiway data analysis," *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 2, pp. 475–487, 2015.

[8] Xu Tan, Yin Zhang, Siliang Tang, Jian Shao, Fei Wu, and Yueting Zhuang, "Logistic tensor regression for classification," in *International Conference on Intelligent Science and Intelligent Data Engineering*. Springer, 2012, pp. 573–581.

[9] Hua Zhou, Lexin Li, and Hongtu Zhu, "Tensor regression with applications in neuroimaging data analysis," *Journal of the American Statistical Association*, vol. 108, no. 502, pp. 540–552, 2013.

[10] Qun Li and Dan Schonfeld, "Multilinear discriminant analysis for higher-order tensor data classification," *IEEE transactions on pattern analysis and machine intelligence*, vol. 36, no. 12, pp. 2524–2537, 2014.

[11] Dacheng Tao, Xuelong Li, Xindong Wu, and Stephen J Maybank, "General tensor discriminant analysis and gabor features for gait recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 10, 2007.

[12] Peter D Hoff, "Multilinear tensor regression for longitudinal relational data," *The annals of applied statistics*, vol. 9, no. 3, pp. 1169, 2015.

[13] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[14] Yoshua Bengio, Pascal Lamblin, Dan Popovici, Hugo Larochelle, et al., "Greedy layer-wise training of deep networks," *Advances in neural information processing systems*, vol. 19, pp. 153, 2007.

[15] Yushi Chen, Zhouhan Lin, Xing Zhao, Gang Wang, and Yanfeng Gu, "Deep learning-based classification of hyperspectral data," *IEEE Journal of Selected topics in applied earth observations and remote sensing*, vol. 7, no. 6, pp. 2094–2107, 2014.

[16] K. Makantasis, K. Karantzalos, A. Doulamis, and N. Doulamis, "Deep Supervised Learning for Hyperspectral Data Classification through Convolutional Neural Networks," in *IEEE International Geoscience and Remote Sensing Symposium (IGARSS 2015)*, July 2015.

[17] M. Vakalopoulou, K. Karantzalos, N. Komodakis, and N. Paragios, "Building detection in very high resolution multispectral data with deep learning features," in *IEEE International Geoscience and Remote Sensing Symposium (IGARSS 2015)*, July 2015.

[18] K. Makantasis, K. Karantzalos, A. Doulamis, and K. Loupos, "Deep learning-based man-made object detection from hyperspectral data," in *International Symposium on Visual Computing*. Springer, 2015, pp. 717–727.

[19] Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle, "Computation of the canonical decomposition by means of a simultaneous generalized schur decomposition," *SIAM journal on Matrix Analysis and Applications*, vol. 26, no. 2, pp. 295–327, 2004.

[20] Marion F Baumgardner, Larry L Biehl, and David A Landgrebe, "220 band aviris hyperspectral image data set: June 12, 1992 indian pine test site 3," *Purdue University Research Repository*, 2015.

[21] Antonio Plaza, Jon Atli Benediktsson, Joseph W Boardman, Jason Brazile, Lorenzo Bruzzone, Gustavo Camps-Valls, Jocelyn Chanussot, Mathieu Fauvel, Paolo Gamba, Anthony Gualtieri, et al., "Recent advances in techniques for hyperspectral image processing," *Remote sensing of environment*, vol. 113, pp. S110–S122, 2009.