HYPERSPECTRAL IMAGE CLASSIFICATION BASED ON TENSOR-TRAIN CONVOLUTIONAL LONG SHORT-TERM MEMORY

Wen-Shuai Hu[1]*, Heng-Chao Li*[1]*, Tian-Yu Ma*[1]*, Qian Du*[2]*, Antonio Plaza*[3]*, and William J. Emery*[4]

[1] School of Information Science and Technology,
Southwest Jiaotong University, Chengdu 610031, China
[2] Department of Electrical and Computer Engineering,
Mississippi State University, Mississippi State, MS 39762, USA
[3] Department of Technology of Computers and Communications,
Escuela Politécnica, University of Extremadura, 10003 Cáceres, Spain
[4] Department of Aerospace Engineering Sciences,
University of Colorado, Boulder, CO 80309, USA

IGARSS 2020 - 2020 IEEE International Geoscience and Remote Sensing Symposium | 978-1-7281-6374-1/20/$31.00 ©2020 IEEE | DOI: 10.1109/IGARSS39084.2020.9324095

ABSTRACT

In recent years, deep learning models have shown great advantages for hyperspectral images (HSIs) classification, in which long short-term memory (LSTM) has attracted plenty of attentions for its characteristic of modeling long-range dependencies. However, for the 2-D extended architecture of it (namely 2-D convolutional LSTM, ConvLSTM2D), it is the special gate structures of ConvLSTM2D that leads to a large number of training parameters and high requirements for device storage. To address this shortcoming, in this paper, a lightweight ConvLSTM2D cell is developed by using tensor-train decomposition (TTD) for the compression of training parameters, which is named TT-ConvLSTM2D and further applied to two state-of-the-art ConvLSTM2D-based HSI classification models for verifying its superiority. Experiments on a widely-used Indian Pines HSI data set are conducted, whose results demonstrate that the proposed TT-ConvLSTM2D cell can effectively reduce the number of the parameters and memory requirements of the whole models within a small range of accuracy degradation.

Index Terms— Classification, convolutional long short-term memory, tensor-train decomposition, hyperspectral image.

1. INTRODUCTION

Hyperspectral images (HSIs) carry abundant spectral information, and with the development of imaging systems and remote sensing technology, HSIs can further provide a richer spatial information, which have been utilized in many fields, i.e., geological exploration [1], and precision agriculture [2].

Thanks to the Program for the National Natural Science Foundation of China under Grant 61871335.

Nowadays, the deep learning-based algorithms have presented a great advance for HSI classification [3]. As an effective deep model, recurrent neural network (RNN) has been widely concerned for modeling long-term dependencies and utilized for HSI classification [4]. For solving the gradient vanishing or explosion problems of it, long short-term memory (LSTM) was proposed, and to better model the spatiotemporal relationships, Shi et al. [5] further developed a convolutional LSTM (ConvLSTM) by extending the way of the input-to-state and state-to-state transitions in LSTM to the convolution operation, which is renamed ConvLSTM2D cell in [6]. In addition to the most common way to utilize the ConvLSTM2D cell by combining with CNN [7], there are also some works to build deep feature extraction models by only using it as the basic unit, such as a bidirectional-ConvLSTM (Bi-CLSTM) model [8], a spatial-spectral ConvLSTM 2D neural network (SSCL2DNN) model [6], which have yielded good performance for HSI classification. However, due to the special gate structure in each ConvLSTM2D cell, there will be a large number of parameters and high storage requirements.

Tensor-train decomposition (TTD) [9] is a tensor factorization algorithm, which can be able to scale a tensor to an arbitrary number of dimensions. To reduce the memory requirement of the CNN-based models, in [10], the fully connected layers were compressed by representing the parameters in the TT-format [9]. Inspired by [11], Garipov *et al.* [12] applied TTD into the convolution kernel for the lightweight convolutional layer. In addition, the TT-format representation of simple RNN model was completed in [13]. Yang *et al.* [14] integrated TTD into LSTM for video classification. On the basis of the above works, this paper focuses on the compression of ConvLSTM2D, and a lightweight ConvLSTM2D cell (namely TT-ConvLSTM2D) is constructed by using TTD.

The remainder of this paper is organized as follows. Section 2 presents the ConvLSTM2D, tensor-train convolutional

layer, and the developed TT-ConvLSTM2D cell. Parameter settings and the related experimental results are introduced in Section 3. Finally, Section 4 summarizes this paper.

2. TT-CONVLSTM2D FOR HSI CLASSIFICATION

2.1. Convolutional Long Short-Term Memory

By extending the way of the input-to-state and state-to-state transitions in LSTM to the convolution calculation, an effective ConvLSTM cell was designed in [5], which is named the ConvLSTM2D cell in [6] for convenience, and the whole calculation formulas in it can be written as:

$$\begin{aligned} i_t &= \sigma(W_{xi} * X_t + W_{hi} * H_{t-1} + W_{ci} \circ C_{t-1} + b_i) \\ f_t &= \sigma(W_{xf} * X_t + W_{hf} * H_{t-1} + W_{cf} \circ C_{t-1} + b_f) \\ \tilde{C}_t &= tanh(W_{xc} * X_t + W_{hc} * H_{t-1} + b_c) \\ C_t &= f_t \circ C_{t-1} + i_t \circ \tilde{C}_t \\ o_t &= \sigma(W_{xo} * X_t + W_{ho} * H_{t-1} + W_{co} \circ C_t + b_o) \\ H_t &= o_t \circ tanh(C_t), \end{aligned} \tag{1}$$

where X_t, H_{t-1}, and C_{t-1} are the input of the current cell, the output and state of the last cell, respectively. i_t, f_t, and H_t are the input, forget, and output gates, respectively, whose convolution kernels are $W_{.i}$, $W_{.f}$, and $W_{.o}$, respectively. $\cdot$ is x, h, and c. $\circ$, σ, and * are the Hadamard product, nonlinear activation function, and convolution operation, respectively.

There are two kinds of convolution kernels in (1). Particularly, $W_{x\cdot}$ is the weight of the input gate with a size of $l \times l \times C \times S$, and $W_{h\cdot}$ is a $l \times l \times S \times S$ convolution filter of the output gate, where l is the kernel size, C and S mean the number of the channels in the inputs and outputs, respectively. This motivates us to design a lightweight ConvLSTM2D cell for improving the efficiency of calculation.

2.2. Tensor-Train Convolutional Layer

TTD of a d-dimensional tensor $\boldsymbol{\mathcal{A}} \in R^{l_1 \times l_2 \ldots \times l_d}$ is a set of tensors $\boldsymbol{\mathcal{G}}_k \in R^{l_k \times r_{k-1} \times r_k}$, and the element of $\boldsymbol{\mathcal{A}}$ is calculated as:

$$\boldsymbol{\mathcal{A}}(t_1, t_2, \ldots, t_d) = \boldsymbol{\mathcal{G}}_1[t_1]\boldsymbol{\mathcal{G}}_2[t_2] \ldots \boldsymbol{\mathcal{G}}_d[t_d], \tag{2}$$

where the set of the elements $\{r_k\}_{k=0}^{d}$ is the $TT-ranks$, and the values of r_0 and r_d are 1. The collection of $\{\boldsymbol{\mathcal{G}}_k\}_{k=1}^{d}$ is called $TT-cores$. Inspired by [10], the factorization of $\boldsymbol{\mathcal{A}} \in R^{(m_1 \cdot n_1) \times (m_2 \cdot n_2) \ldots \times (m_d \cdot n_d)}$ is further expressed as:

$$\begin{aligned} &\boldsymbol{\mathcal{A}}((i_1, j_1), (i_2, j_2), \ldots, (i_d, j_d)) = \\ &\boldsymbol{\mathcal{G}}_1^{'}[i_1, j_1]\boldsymbol{\mathcal{G}}_2^{'}[i_2, j_2] \ldots \boldsymbol{\mathcal{G}}_d^{'}[i_d, j_d], \end{aligned} \tag{3}$$

where $\boldsymbol{\mathcal{G}}_k^{'}[i_k, j_k] \in R^{r_{k-1} \times r_k}$, $i_k = \lfloor \frac{t_k}{n_k} \rfloor$, and $j_k = t_k - n_k \lfloor \frac{t_k}{n_k} \rfloor$. In addition, $l_k = m_k \times n_k$, $t_k = 1, 2, \ldots, l_k$, and $k = 1, 2, \ldots, d$.

Taking the basic convolutional layer in [12] as an example. The input tensor $\boldsymbol{\mathcal{X}} \in R^{W \times H \times C}$ is transformed into the output tensor $\boldsymbol{\mathcal{Y}} \in R^{W \times H \times S}$ by a convolution kernel $\boldsymbol{\mathcal{K}} \in R^{l \times l \times C \times S}$. To reduce the computational and memory complexity of it, Garipov *et al.* [12] formulated the convolutional layer as a matrix-by-matrix multiplication and designed a TT-convolution (TTC) layer by decomposing the convolution kernel according to the channel dimension with TTD as shown in Fig. 1, and the outputs of it can be described as:

$$\begin{aligned} \boldsymbol{\mathcal{Y}}(x, y, s_1, s_2, \ldots, s_d) = \sum_{i=1}^{l} \sum_{j=1}^{l} \sum_{c_1, c_2, \ldots, c_d} \\ \boldsymbol{\mathcal{X}}(i + x - 1, j + y - 1, c_1, c_2, \ldots, c_d) \cdot \\ \boldsymbol{\mathcal{G}}_0^{'}[i, j]\boldsymbol{\mathcal{G}}_1^{'}[c_1, s_1]\boldsymbol{\mathcal{G}}_2^{'}[c_2, s_2] \ldots \boldsymbol{\mathcal{G}}_d^{'}[c_d, s_d], \end{aligned} \tag{4}$$

where $C = \prod_{k=1}^{d} C_k$, $S = \prod_{k=1}^{d} S_k$, and $x, y = 1, 2, \ldots, l$. $\boldsymbol{\mathcal{G}}_0^{'}[i, j]$ and $\boldsymbol{\mathcal{G}}_k^{'}[c_k, s_k]$ $(k = 1, 2, \ldots, d)$ are the $TTC-cores$, which are the trainable parameters of this TTC layer. Corresponding, the set of the elements $\{r_k\}_{k=0}^{d+1}$ is the $TTC-ranks$, and the values of r_0 and r_{d+1} are fixed to 1.

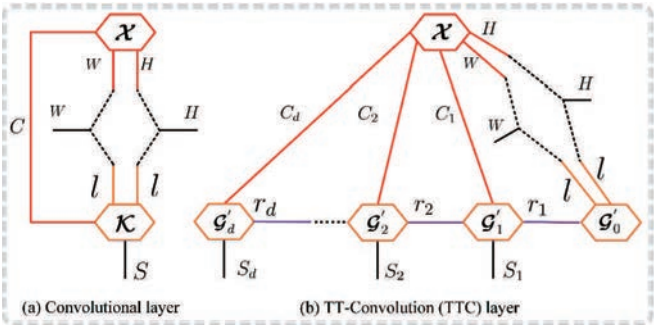

Fig. 1. Illustration of the TTC layer.

For convenience, we define the following notation to describe the above calculation formula of the TTC layer:

$$Y = TTCL(K, X). \tag{5}$$

2.3. HSI classification based on TT-ConvLSTM2D

Inspired by the TTC layer in Section 2.2, an useful TT-ConvLSTM2D cell is developed in this section. As shown in Section 2.1, there are two types of weights, such as $W_{x\cdot}$ and $W_{h\cdot}$ (ignoring $W_{c\cdot}$), whose the kernel sizes are $l \times l \times C \times S$ and $l \times l \times S \times S$, respectively.

Firstly, we reshape the input X_t and output H_{t-1} in (1) into the $(2 + d)$ tensors $\boldsymbol{\mathcal{X}}_t \in R^{W \times H \times C_1 \times C_2 \times \ldots \times C_d}$ and $\boldsymbol{\mathcal{H}}_{t-1} \in R^{W \times H \times S_1 \times S_2 \times \ldots \times S_d}$, respectively. Then, by applying the TTC layer in (4), the decompositions of $W_{x\cdot}$ and $W_{h\cdot}$ corresponding to X_t and H_{t-1} can be written as:

$$\begin{aligned} \boldsymbol{\mathcal{W}}_{x\cdot} &= \boldsymbol{\mathcal{G}}_0^{'}[i, j]_{x\cdot}\boldsymbol{\mathcal{G}}_1^{'}[c_1, s_1]_{x\cdot}\boldsymbol{\mathcal{G}}_2^{'}[c_2, s_2]_{x\cdot} \ldots \boldsymbol{\mathcal{G}}_d^{'}[c_d, s_d]_{x\cdot} \\ \boldsymbol{\mathcal{W}}_{h\cdot} &= \boldsymbol{\mathcal{G}}_0^{'}[i, j]_{h\cdot}\boldsymbol{\mathcal{G}}_1^{'}[s_1, s_1]_{h\cdot}\boldsymbol{\mathcal{G}}_2^{'}[s_2, s_2]_{h\cdot} \ldots \boldsymbol{\mathcal{G}}_d^{'}[s_d, s_d]_{h\cdot}. \end{aligned} \tag{6}$$

Finally, substituting $\boldsymbol{\mathcal{X}}_t$, $\boldsymbol{\mathcal{H}}_{t-1}$, and (6) into (4), we can obtain the corresponding (2+d)-dimensional outputs $\boldsymbol{\mathcal{Y}}_t^{x\cdot}$ and $\boldsymbol{\mathcal{Y}}_{t-1}^{h\cdot} \in R^{W\times H\times S_1\times S_2\times\ldots\times S_d}$, which then are reshaped into the 3-D tensors to obtain the final outputs $Y_{t-1}^{x\cdot}$ and $Y_{t-1}^{h\cdot} \in R^{W\times H\times S}$ of each gate.

On the basis of the above analysis, the calculation formulas of the designed TT-ConvLSTM2D cell are expressed as:

$$\begin{aligned}
i_t &= \sigma(TTCL(W_{xi}, X_t) + TTCL(W_{hi}, H_{t-1}) \\
&\quad + W_{ci} \circ C_{t-1} + b_i) \\
f_t &= \sigma(TTCL(W_{xf}, X_t) + TTCL(W_{hf}, H_{t-1}) \\
&\quad + W_{cf} \circ C_{t-1} + b_f) \\
\tilde{C}_t &= tanh(TTCL(W_{xc}, X_t) + TTCL(W_{hc}, H_{t-1}) \\
&\quad + b_c) \\
C_t &= f_t \circ C_{t-1} + i_t \circ \tilde{C}_t \\
o_t &= \sigma(TTCL(W_{xo}, X_t) + TTCL(W_{ho}, H_{t-1}) \\
&\quad + W_{co} \circ C_t + b_o) \\
H_t &= o_t \circ tanh(C_t).
\end{aligned} \tag{7}$$

Based on the above TT-ConvLSTM2D cell and by applying it as the fundamental unit, a TT-ConvLSTM2D layer can be constructed. Similar to ordinary ConvLSTM2D layer [6]-[8], the TT-ConvLSTM2D layer can also be used alone or by combining with CNN to build the feature extraction models for various applications. For verifying the effectiveness of the TT-ConvLSTM2D cell, we apply it into two state-of-the-art HSI classification methods built by using the ConvLSTM2D cell alone, such as SSCL2DNN [6] and Bi-CLSTM [8]. The detailed experimental results are introduced in Section 3.

3. EXPERIMENTAL RESULTS AND DISCUSSIONS

In our experiments, Bi-CLSTM and SSCL2DNN are selected as two comparative methods, and as shown in [6], [8], the same parameter settings are utilized. Concretely, Bi-CLSTM has the following structure: two parallel $3\times3\times32$ ConvLSTM2D layers, two maxpooling layers (2×2 with stride 2). For SSCL2DNN, a $4\times4\times32$ ConvLSTM2D layer, a $3\times3\times64$ ConvLSTM2D layer, two 2×2 maxpooling layers with stride 2, a fully connected layer with 128 nodes are formed the backbone. In addition, the common used Indian Pines data set is used for quantitative analysis, and three widely used metrics are applied in the experiments, i.e., overall accuracy (OA), average accuracy (AA), and Kappa coefficient (κ).

Firstly, for Indian Pines data set, it was acquired by the Airborne Visible/Infrared Imaging Spectrometer (AVIRIS) sensor over the Indian Pines test site in northwest Indiana in 1992, whose ground-truth map, false-color map, and training samples are given in Fig. 2 and Table I, respectively. There are 145×145 pixels with a spatial resolution of 20 meters per pixel (mpp), 16 ground-truth classes, and 224 bands that cover the spectral range from 0.4 to 2.5 μm. After removing 24 unusable bands, 200 bands are maintained. In our experiments, 10% samples of the available labeled samples are randomly selected for training, while the rest samples are utilized for testing, and the size of local spatial window of each pixel is fixed as 27×27.

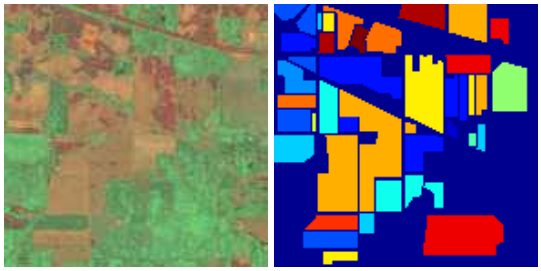

Fig. 2. (Left) False-color map and (Right) ground-truth map of the Indian Pines (bands 20, 40, and 60).

Then, based on the TT-ConvLSTM2D cell in Section 2.3, the corresponding TT-ConvLSTM2D layer can be further designed, with which all ConvLSTM2D layers in Bi-CLSTM and SSCL2DNN are modified, and to describe convenience, the improved models are named Bi-TTCLSTM and SSTTCL2DNN, respectively. Concretely, for Bi-TTCLSTM, two parallel ConvLSTM2D layers have the same dimensions, in which the input and output feature dimensions remain the same, and the $TTC-ranks$ are set to $[1, 8, 1]$. As for SSTTCL2DNN, the decompositions of the input and output feature dimensions in the $4\times4\times32$ ConvLSTM2D layer are the same as that in Bi-TTCLSTM, while the input and output feature dimensions in the $3\times3\times64$ ConvLSTM2D layer are factorized as $4\times2\times4$ and $4\times4\times4$, respectively, and the $TTC-ranks$ are $[1, 8, 8, 8, 1]$.

Table I. The Number of Training Samples for The Indian Pines Data Set

NO.	Color	Class	Training	Total
1		Alfalfa	5	46
2		Corn-notill	143	1428
3		Corn-mintill	83	830
4		Corn	24	237
5		Grass-pasture	48	483
6		Grass-trees	73	730
7		Grass-pasture-mowed	3	28
8		Hay-windrowed	48	478
9		Oats	2	20
10		Soybean-notill	97	972
11		Soybean-mintill	246	2455
12		Soybean-clean	59	593
13		Wheat	21	205
14		Woods	127	1265
15		Buildings-Grass-Trees-Drives	39	386
16		Stone-Steel-Towers	9	93
Total			1027	10249

We compare Bi-TTCLSTM and SSTTCL2DNN with the naive methods on the Indian Pines data set, and for comparing and analyzing fairly, all models are trained from

Table II. Classification Results of Different Methods for The Indian Pines Data Set Using 10% Training Samples with $TTC - rank = 8$

Class	Bi-CLSTM	Bi-TTCLSTM	SSCL2DNN	SSTTCL2DNN
1	91.06	94.31	**100.00**	98.37
2	94.29	93.90	**98.11**	97.02
3	93.13	91.03	96.56	**96.74**
4	88.89	90.61	**96.56**	95.62
5	94.25	93.95	96.09	**96.32**
6	99.49	98.78	98.02	**98.83**
7	93.33	66.67	84.00	**92.00**
8	99.46	**99.69**	**99.69**	98.84
9	38.89	40.74	55.56	**81.48**
10	95.73	96.19	**97.07**	95.62
11	96.60	96.50	**99.34**	98.43
12	89.37	85.83	96.75	**96.82**
13	95.65	96.56	**98.19**	91.67
14	**99.80**	98.95	99.65	98.18
15	97.12	95.77	**98.85**	97.98
16	89.29	85.32	85.32	**92.46**
OA	95.62	95.08	**98.03**	97.33
AA	90.90	89.05	93.37	**95.40**
κ	94.99	94.37	**97.40**	96.96

scratch. Based on the above parameter settings, the quantitative assessments by using the Indian Pines data set are reported in Table II, from which it is obvious that when $TTC - rank$ of each TT-ConvLSTM2D layer is set to 8, although there are just two ConvLSTM2D layers, we can still obtain 1.106 $\times$ compression with 0.52% classification accuracy drop for Bi-TTCLSTM, while 5.690 $\times$ compression with 0.70% for SSTTCL2DNN, which verifies that the proposed TT-ConvLSTM2D cell can generate a good compression performance while obtaining a small performance loss. This is very significant for the ConvLSTM2D-based models to reduce the computational complexity and storage requirements, which also proves the superiority and effectiveness of the proposed TT-ConvLSTM2D cell.

4. CONCLUSIONS

In this paper, an effective and lightweight TT-ConvLSTM2D cell is developed from the ConvLSTM2D cell by using TTD. The experiments conducted on a common HSI data set and two state-of-the-art classification models show that compared with the original ConvLSTM2D cell, the total number of parameters and memory requirements of the improved TT-ConvLSTM2D cell will be effectively reduced within a small range of performance degradation.

5. REFERENCES

[1] F. van der Meer, "Analysis of spectral absorption features in hyperspectral imagery," *Int. J. Appl. Earth Observ. Geoinf.*, vol. 5, no. 1, pp. 55-68, Feb. 2004.

[2] X. Zhang, Y. Sun, K. Shang, L. Zhang, and S. Wang, "Crop classification based on feature band set construction and objectoriented approach using hyperspectral images," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 9, no. 9, pp. 4117-4128, Sep. 2016.

[3] W. Hu, Y. Huang, W. Li, F. Zhang, and H. Li, "Deep convolutional neural networks for hyperspectral image classification," *Journal of Sensors*, vol. 2015, no. 2, pp. 1-12, Jul. 2015.

[4] L. Mou, P. Ghamisi, and X. X. Zhu, "Deep recurrent neural networks for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 7, pp. 3639-3655, Jul. 2017.

[5] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-k. Wong, and W.-c. Woo, "Convolutional LSTM network: A machine learning approach for precipitation nowcasting," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015.

[6] W. Hu, H Li, L. Pan, W. Li, R. Tao, and Q. Du, "Spatial-spectral feature extraction via deep ConvLSTM neural networks for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 58, no. 6, pp. 4237-4250, Jun. 2020.

[7] M. Rußwurm and M. Körner, "Multi-temporal land cover classification with sequential recurrent encoders," *ISPRS Int. J. Geo-Inf.*, vol. 7, no. 4, pp. 129, Mar. 2018.

[8] Q. Liu, F. Zhou, R. Hang, and X. Yuan, "Bidirectional-convolutional LSTM based spectral-spatial feature learning for hyperspectral image classification," *Remote Sens.*, vol. 9, no. 12, pp. 17-28, Dec. 2017.

[9] I. V. Oseledets, "Tensor-Train decomposition," *SIAM J. Scientific Comput.*, vol. 33, no. 5, pp. 2295-2317, 2011.

[10] A. Novikov, D. Podoprikhin, A. Osokin, and D. Vetrov, "Tensorizing neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, pp. 442-450, 2015.

[11] A. Vedaldi and K. Lenc, "Matconvnet: Convolutional neural networks for matlab," in *Proc. 23rd ACM Int. Conf. Multimedia*, 2015.

[12] T. Garipov, D. Podoprikhin, A. Novikov, and D. Vetrov. (2016). "Ultimate tensorization: Compressing convolutional and FC layers alike." [Online]. Available: https://arxiv.org/abs/1611.03214

[13] A. Tjandra, S. Sakti, and S. Nakamura. (2017). "Compressing recurrent neural network with tensor train." [Online]. Available: https://arxiv.org/abs/1705.08052

[14] Y. Yang, D. Krompass, and V. Tresp. (2017). "Tensor-train recurrent neural networks for video classification." [Online]. Available: https://arxiv.org/abs/1707.01786