IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 26, 2021, accepted February 12, 2021, date of publication February 16, 2021, date of current version March 8, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3059660

Application of Tensor Train Decomposition in S2VT Model for Sign Language Recognition

BIAO XU, SHILIANG HUANG, AND ZHONGFU YE
National Engineering Laboratory for Speech and Language Information Processing, Institute of Statistical Signal Processing, University of Science and Technology of China, Hefei 230027, China

Corresponding author: Zhongfu Ye (yezf@ustc.edu.cn)

This work was supported by the National Natural Science Foundation of China under Grant 61671418.

ABSTRACT Sign language recognition is a conversion of sign language into text or speech, bridging the communication between the hearing and society. Recently, sequence-to-sequence video to text (S2VT) models has been employed in the field of sign language recognition as an effective method. However, more than 20 million parameters trained in S2VT models will result in a huge consumption in memory and computational resources, making it hard to be applied in mobile devices. In order to overcome this issue, we proposed to employ tensor-train decomposition in S2VT models to reduce the parameters. First, the impact of parameters of tensor-train factorization on the model performance was investigated systematically. After that, we applied tensor-train decomposition in different layers of a S2VT model to establish 6 tensor-train S2VT models for Chinese sign language recognition. The experimental results demonstrated that when the fully-connected layer and the first LSTM layer in S2VT was represented with tensor-train format, the model could obtain the best performance, remaining high accuracy and reducing parameters and memory significantly. The proposed tensor-train S2VT models can also be applied in other sequence-to-sequence problems to improve the performance.

INDEX TERMS LSTM, sign language recognition, S2VT, tensor-train decomposition.

I. INTRODUCTION

China has the largest number of hearing disabilities in the world. According to statistics, there are about 27.8 million people with hearing disabilities in China, accounting for more than 30% of the country's disabled people [1]. Due to the hearing impairment, the hearing disabled people often encounter inconvenience and embarrassment in daily life when communicating with the society. Therefore, it is highly necessary to develop an effective method to help the communication between the hearing disabled and other people.

Sign language recognition [2] is an effective way to help deaf people communicate with other people, and it is regarded as a form of human-computer interaction. In general, sign language recognition can be divided into two methods, namely traditional and neural network (NN) based methods [3]. Nowadays, more and more people use the latter for sign language recognition due to the rapid development of Convolutional Neural Network (CNN) [4] which can be applied for feature extraction, and of Recurrent Neural Network (RNN) [5], which can be employed as recognition model. The inputs of sign language and the recognition output can be regarded as two sequences, so the sequence-to-sequence model, like sequence-to-sequence video to text (S2VT) [6] model, can be used for sign language recognition. Previous researchers [7]–[9] have demonstrated that sequence-to-sequence models are suitable and effective for sign language recognition. However, enormous parameters in S2VT models result in huge consumption in memory and make it hard to be applied in mobile devices. For instance, when the input shape of S2VT is 4096 and the number of neurons is 1000, the number of parameters in the fully-connected layer will be up to 4.1×10^6. There are also two Long-Short-Term- Memory (LSTM) layers in a S2VT model, so the total number of parameters can reach 24 million. Therefore, it is highly necessary to compress the parameters in S2VT model. It has been proved in [10]that tensor-train decomposition could effectively reduce the model parameters without a large loss in accuracy. In this article, we will employ tensor-train decomposition in S2VT model to reduce the model's parameters, and apply the tensor-train S2VT models for Chinese sign language recognition.

The associate editor coordinating the review of this manuscript and approving it for publication was Naveed Akhtar.

In summary, the major contributions of this paper are:

1) the impact of parameters of tensor-train factorization on the model performance, including accuracy, parameters and runtime, are discussed systematically for the first time, showing us how to select the parameters. Proper parameter configuration can reduce the model runtime significantly.
2) Six tensor-train S2VT models are proposed when the tensor-train decomposition is applied in different layers of S2VT. The comparison among the original S2VT model and the proposed tensor-train S2VT models are discussed in aspect of model accuracy, parameters.

The rest of this paper is organized as follows: in Section II, we will give a brief review of the related works. We describe our method in Section III. Then we discuss the experimental results in Section IV. Finally, we make a conclusion in Section V.

II. RELATED WORK

Nowadays, with the rapid development of deep learning networks, more and more researchers have begun to use deep learning methods for sign language recognition, such as CNNs and LSTM networks. Pigou *et al.* [11] used CNN extracting sign language features, replacing traditional feature extraction methods, to recognize Italian sign language. Huang *et al.* [12] designed a multi-channel 3D-CNN network using the multi-channel information obtained by Kinect, and achieved good results in Chinese sign language recognition. Yang and Zhu [13] proposed a video-based Chinese sign language recognition method using CNN to extract upper body images, and used a pre-trained CNN to recognize gestures. Pigou *et al.* [14] proposed to use RNN in sign language samples for modeling, combined with CNN timing pooling and a two-way RNN to achieve sign language recognition. Wu *et al.* [15] proposed to use the Belief Network (Deep Belief Network, DBN) to process trajectory information, use 3D-CNN to process RGB information, and then use Hidden Markov Model (HMM) for modeling through fusion features for sign language recognition. Liao *et al.* [16] presented a multimodal dynamic sign language recognition method based on a deep 3-dimensional residual ConvNet and bi-directional LSTM networks, which was named as BLSTM-3D residual network (B3D ResNet). In [17], a cross-modal learning approach embedding video and text in a Joint-Latent Space was proposed for continuous sign language recognition.

Since the sign language recognition model can be regarded as a sequence model, more and more researchers are beginning to use LSTM [18] networks for sign language recognition. Liu *et al.* [19] only used the three-dimensional trajectory information obtained by Kinect and constructed a classification network with LSTM to recognize Chinese sign language. S2VT [6] is a typical sequence-to-sequence model for video description, which is similar to sign language recognition. Mao *et al.* [7] treated Chinese sign language recognition as a sequence-to-sequence problem, and proposed a processing framework based on encoder and decoder. Li *et al.* [8] proposed a new Hand Shape Descriptor (Specific Hand Shape, SHS), and used the S2VT model to recognize Chinese sign language. Huang *et al.* [9] proposed a sequence-to-sequence Chinese sign language recognition method based on Keyframe Centered Clips. Although the deep learning method has achieved good results in sign language recognition, it still has some drawbacks, such as excessive parameters, slow network training process and serious parameter redundancy, etc.

Some researchers apply tensor-train decomposition on the deep learning model to reduce model parameters. The tensor-train factorization was first introduced by Oseledets [10], showing tensor-train decomposition has the obvious advantage of being capable of scaling to an arbitrary number of dimensions. Novikov *et al.* [20] showed how to reshape a fully-connected layer into a high-dimensional tensor and how to factorize this tensor using tensor-train decomposition. Then in [21], it was shown that even the convolutional layers can also be compressed with tensor-train layers. Yang *et al.* [22] used tensor-train decomposition to decompose the Input-to-hidden matrix to process high-dimensional input signals in RNNs (LSTM, Gated Recurrent Unit), such as video modeling tasks. Utilizing LSTM based on tensor-train format, Samui *et al.* [23] proposed a deep TensorNet model for single-channel speech enhancement tasks, which achieved competitive performances with the state-of-the-art uncompressed RNN model. Xu *et al.* [24] proposed a novel framework based on a tensor-train NN (TensorNet) to extract the essential and discriminative features from the whole-brain fMRI data. In [25], Zou and Yang used the tensor-train decomposition method in the Software Defined Access (SDA) network and proposed a TT-SDA network to improve the performance of sparse signal recovery.

In this article, we will first investigate the impact of parameters of tensor-train factorization on model performances. After that, we intend to apply tensor-train factorization in S2VT models to establish tensor-train S2VT models for sign language recognition.

III. OUR METHOD

A. S2VT: SEQUENCE TO SEQUENCE MODEL

S2VT [6] is a typical sequence-to-sequence model for video description, where the input is a sequence of video frames or features $(x_1, \cdots, x_n)$, and the output is a sequence of words $(y_1, \cdots, y_m)$. The models have been applied widely in the field of sequence-to-sequence problems. Herein, a S2VT model is employed in sign language recognition. Each sign language videos will be transformed into a sequence of video frames or features which can be fed into the S2VT model after feature extraction, while the corresponding captions will be obtained from the outputs' sequence.

The framework of S2VT is shown in Fig.1, The main processing flow of S2VT is as follows. First, the main features of Chinese sign language videos are extracted through the VGG16 [26] network, and a 4096 vector will be obtained

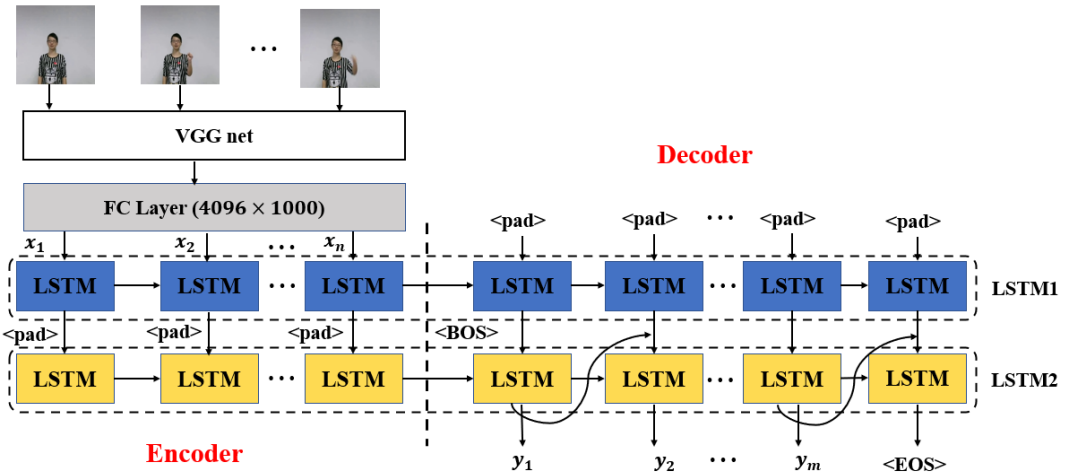

FIGURE 1. The framework of S2VT, which has two LSTMs. The first LSTM layer (colored blue) models visual features inputs of sign language videos. And the second LSTM layer (colored yellow) models the language given the text input and the hidden representation of the video sequence.

to represent the corresponding video. It passes through a fully-connected layer (FC layer) and the shape of the vectors are reduced to 1000. After that, the vectors will be feed into the first LSTM layer (LSTM1) of the S2VT model and the hidden representation from LSTM1 (with 1000 hidden units) will be feed into the second LSTM layer (LSTM2) (with 1000 hidden units). Finally, the corresponding word sequences are generated from the output of the second LSTM layer. In this S2VT model, <BOS> is labelled to indicate the beginning of decoding, while <EOS> is used to indicate the end of decoding. Zeros padding is used when there is no input at the time step.

When a S2VT model is directly employed in sign language recognition, over 20 million of parameters will be generated. For instance, as shown in Fig.1, the fully-connected layer has 4,097,000 parameters, and the first LSTM layer has 8,004,000 parameters. The second LSTM layer has 24,004,000 parameters. In this study, millions of parameters in this model will be reduced through tensor-train factorization.

B. TENSOR-TRAIN FACTORIZATION

Tensor-train factorization [10] is a kind of tensor factorization models that can scale to an arbitrary of dimensions. Assuming a d-dimensional target tensor of the form $\mathcal{A} \in \mathbb{R}^{p_1 \times p_2 \times \cdots \times p_d}$, it can be factorized in form of:

$$\widehat{\mathcal{A}}(l_1, \cdots, l_d) = \mathcal{G}_1(l_1)\, \mathcal{G}_2(l_2) \cdots \mathcal{G}_d(l_d) \tag{1}$$

where $\mathcal{G}_k \in \mathbb{R}^{p_k \times r_{k-1} \times r_k}$, $l_k \in [1, p_k]\, \forall k \in [1, d]$, and $r_0 = r_d = 1$.

As shown in Fig.2, all matrices $\mathcal{G}_k(l_k)$ will be transformed with the size of $r_{k-1} \times r_k$, and $r_0 = r_d = 1$, to retain the final matrix multiplication result as a scalar. The set of matrices $\{\mathcal{G}_k\}_{k=1}^{d}$ are called TT-core, the complexity is determined by the ranks $\{r_0, r_1, \cdots, r_d\}$.

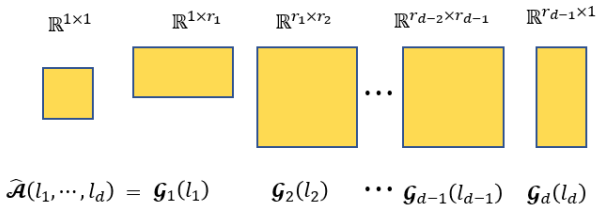

FIGURE 2. Tensor-Train Factorization Model: Calculating an element $\widehat{\mathcal{A}}(j_1, \cdots j_d)$ using a set of TT-cores $\{\mathcal{G}_k\}_{k=1}^{d}$.

If p_k in (1) is factorized as $p_k = m_k \cdot n_k \forall k \in [1, d]$, $\mathcal{G}_k$ can be reshaped as $\mathcal{G}_k^* \in \mathbb{R}^{m_k \times n_k \times r_{k-1} \times r_k}$. Then, target tensor $\mathcal{A}$ will be represented equivalently to (1) as follows [20]:

$$\begin{aligned}&\widehat{\mathcal{A}}((i_1, j_1), (i_2, j_2), \cdots, (i_d, j_d)) \\ &\quad = \mathcal{G}_1^*(i_1, j_1)\, \mathcal{G}_2^*(i_2, j_2) \cdots \mathcal{G}_d^*(i_d, j_d)\end{aligned} \tag{2}$$

where the indices $i_k = \left\lfloor \frac{l_k}{n_k} \right\rfloor$, $j_k = l_k - n_k \left\lfloor \frac{l_k}{n_k} \right\rfloor$ and $\mathcal{G}_k^*(i_k, j_k) \in \mathbb{R}^{r_{k-1} \times r_k}$.

C. TENSOR-TRAIN LAYER

The linear transformation in NNs can be written as:

$$\boldsymbol{y} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b} \tag{3}$$

where $\boldsymbol{x} \in \mathbb{R}^M$, $\boldsymbol{y} \in \mathbb{R}^N$, $\boldsymbol{b} \in \mathbb{R}^N$ and $\boldsymbol{W} \in \mathbb{R}^{N \times M}$. (3) can be rewritten equally in scalar format as:

$$\widehat{\boldsymbol{y}}(j) = \sum\nolimits_{i=1}^{M} \boldsymbol{W}(i, j)\, \boldsymbol{x}(i) + \boldsymbol{b}(j) \quad \forall j \in [1, N] \tag{4}$$

If both M and N can be factorized into two integer arrays of the same length, such as $M = \prod_{k=1}^{d} m_k$, $N = \prod_{k=1}^{d} n_k$. Then, the input $\boldsymbol{x}$ and the output vector $\boldsymbol{y}$ should be reshaped into two tensors with $\mathcal{X} \in \mathbb{R}^{m_1 \times \cdots \times m_d}$ and $\mathcal{Y} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$. Therefore, the linear mapping function can be written as:

Algorithm 1 Tensor-Train Factorization

1: calculate $i_k = \left\lfloor \frac{l_k}{n_k} \right\rfloor$ and $j_k = l_k - n_k \left\lfloor \frac{l_k}{n_k} \right\rfloor$.
2: establish $\mathcal{G}_k^* (i_k, j_k)$ to construct $\widehat{\mathcal{A}}$.
3. update $\mathcal{G}_k^* (i_k, j_k)$ to maximally reduce the error ensuring $||\mathcal{A}\text{-}\widehat{\mathcal{A}}|| < \varepsilon$.

$$\widehat{\mathcal{Y}}(j_1, \cdots, j_d) = \sum_{i_1, i_2, \cdots, i_d} \mathcal{W}((i_1, j_1), (i_2, j_2), \cdots, (i_d, j_d)) \cdot \boldsymbol{X}(i_1, \cdots, i_d) + \mathcal{B}(j_1, \cdots, j_d) \quad (5)$$

The d-dimensional double-indexed tensor of weights $\mathcal{W}$ in (5) can be replaced by its tensor-train representation:

$$\mathcal{W}((i_1, j_1), (i_2, j_2), \cdots, (i_d, j_d)) = \mathcal{G}_1^*(i_1, j_1)\, \mathcal{G}_2^*(i_2, j_2) \cdots \mathcal{G}_d^*(i_d, j_d) \quad (6)$$

Now instead of explicitly storing the full tensor $\mathcal{W}$ of size $\prod_{k=1}^{d} m_k \cdot n_k = MN$, we only store its TT format, i.e., the set of low-rank core tensors $\{\mathcal{G}_k\}_{k=1}^{d}$ of size $\sum_{k=1}^{d} m_k n_k r_{k-1} r_k$, which can approximately reconstruct $\mathcal{W}$ [22].

Thus, the computational complexity of an $M \times N$ fully-connected layer can be reduced from $O(MN)$ to $O\left(dr^2 m \cdot max\{M, N\}\right)$, where $m = max\{m_k\}$, $r = max\{r_k\}$.

When the weight matrix of a fully-connected layer is transformed into the TT format, it produces a tensor-train Layer (*TTL*). For the rest of the paper, $\boldsymbol{y} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$, whose weight matrix W is factorized with tensor-train factorization, will be written as

$$\boldsymbol{y} = TTL(\boldsymbol{W}, \boldsymbol{x},) + \boldsymbol{b} \quad (7)$$

D. TENSOR-TRAIN LSTM

LSTM [18] is a variant of RNN and is widely employed in sequence-to-sequence problems. As shown in Fig.3, an LSTM unit contains a memory cell and three gates, called forget gate (f_t), input gate (i_t) and output gate (o_t). The forget gate determines the information that the cell state needs to discard, the input gate determines how much information the cell should input, and the output gate needs to determine what value to output.

Using TTL as in (7), the tensor-train LSTM model can be represented as follows [22]:

$$\begin{aligned} i_t &= \sigma(TTL(\boldsymbol{W}_i, [x_i, h_{t-1}] + \boldsymbol{b}_i)) \\ f_t &= \sigma\left(TTL\left(\boldsymbol{W}_f, [x_i, h_{t-1}] + \boldsymbol{b}_f\right)\right) \\ o_t &= \sigma(TTL(\boldsymbol{W}_o, [x_i, h_{t-1}], +\boldsymbol{b}_o)) \\ g_t &= tanh\left(TTL\left(\boldsymbol{W}_g, [x_i, h_{t-1}] + \boldsymbol{b}_g\right)\right) \\ c_t &= f_t{}^{\circ} c_{t-1} + i_t{}^{\circ} g_t \\ h_t &= o_t{}^{\circ} tanh(c_t) \end{aligned} \quad (8)$$

where $\boldsymbol{W_i}$, $\boldsymbol{W_f}$, $\boldsymbol{W_o}$ and $\boldsymbol{W_g} \in \mathbb{R}^{H \times (I+H)}$, represent the concatenated matrices which is required to compute each gate output. As shown as (8), each LSTM block needs 4 TTL to represent its functionality in tensor-train format.

Algorithm 2 Tensor-Train LSTM

1: calculate the linear transformation of three gates, namely input gate i_t, forget gate f_t, output gate (o_t), and memory cell g_t based on TTL.
2: calculate the corresponding activations through sigmoid function σ, and tangent functions *tanh*.
3. update memory cell state, $c_t = f_t{}^{\circ} c_{t-1} + i_t{}^{\circ} g_t$ and hidden state, $h_t = o_t{}^{\circ} tanh(c_t)$.

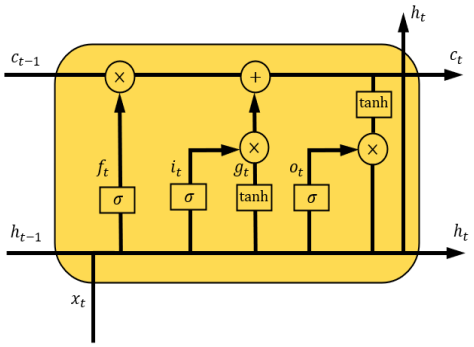

FIGURE 3. Block of a basic LSTM unit.

Based on the above description, we can replace the fully-connected layer of the S2VT model with tensor-train layer, and replace its first LSTM layer and second LSTM layer with tensor-train LSTM to improve the performance of the model in sign language recognition.

IV. EXPERIMENTS

In this section, we first explore the influence of its parameters' setting on tensor-train factorization, and then established six tensor-train S2VT models. These models are carried out based on a sign language dataset from USTC-SLR [12]. At last, we evaluate the effectiveness of our models on this dataset.

A. IMPACT OF TENSOR-TRAIN PARAMETERS

We assume $\boldsymbol{W} \in \mathbb{R}^{M \times N}$, $M = \prod_{i=1}^{4} m_i$ and $N = \prod_{i=1}^{4} n_i$. It is obvious that m_i and n_i can have different values and different orders. However, the impact of tensor-train parameters on the model performance is still unknown, especially in model runtime. Thus, in this part, we will compare different strategies for setting tensor-train parameters: the ranks and different decomposition methods of the tensors representing the input/output.

Since the size of $\boldsymbol{W}$ in the fully-connected layer of S2VT model is 4096×1000, we build a two-layer NN, and the first layer with the shape of 4096×1000 will be represented with TTL. We run the experiments on the MNIST dataset [27], resizing the origin images size from 28×28 to 64×64.

One of the important parameters is the rank of tensor-train factorization. We use rank = 0 to indicate that tensor-train decomposition is not applied on the model, and the rank is varied from 0 to 6, to compare the experimental performance.

As shown in Fig.4, the parameters of model after tensor-train factorization are reduced drastically, and the accuracy rate is increased at the same time. When ranks raise, the number of parameters increases but the accuracy rate changes slightly. We can obtain the best result when rank is equal to 3. The accuracy is the highest, increasing by 25.8%, compared with the undecomposed model, and the number of parameters was also relatively small. Therefore, rank = 3 is chosen for the following experiments.

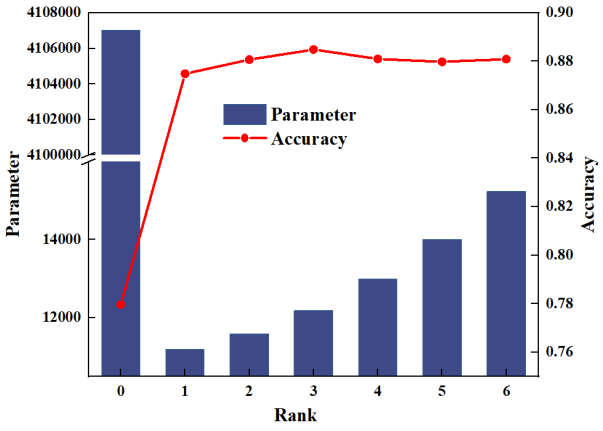

FIGURE 4. Comparative results with the change of rank.

Subsequently, we study the impact of different decomposition methods of the tensor-train factorization. At first, we study the decomposition in the output tensor. Herein, M = 4096 is decomposed into (8,8,8,8), but N = 1000 is decomposed differently. We assume $\{n_i|2, 5, 10, 10\}$ to explore the effect of different arrangement orders on the results.

As shown in Fig.5, we can find that when the output tensor 1000 is decomposed into (10, 10, 5, 2), the runtime is the shortest, while the runtime is the longest when $n_1 = 2$, $n_2 = 5$,$n_3 = 10$,$n_4 = 10$, increasing by 376%. It can also be found that the accuracy varies slightly. Therefore, when the order of the output shape is from large to small, the runtime is the shortest, and the accuracy still keep high.

Then, we research different decomposition methods of the input tensors shape. We assume $\{m_i|4, 4, 16, 16\}$ to explore the effect of different arrangement orders on the model performance.

As shown in Fig.6, the runtime in the first case is the shortest, where the output is decomposed into (4, 4, 16, 16), and the runtime is 85s. However, the runtime is the longest when $m_1 = 16$,$m_2 = 16$,$m_3 = 4$,$m_4 = 4$, increasing by 261% compared to the first case. We can conclude that the order of input tensor shapes should be arranged from small to large, which will result in a shorter runtime and still remain high accuracy.

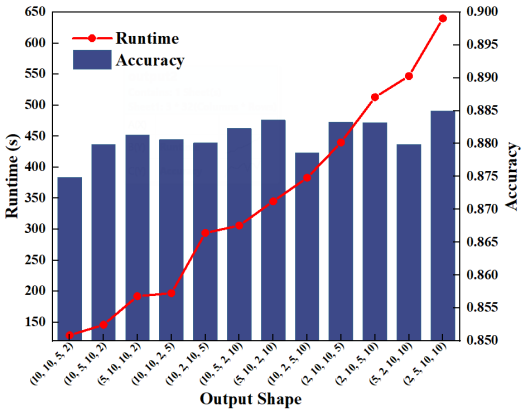

FIGURE 5. Different tensors representing the output.

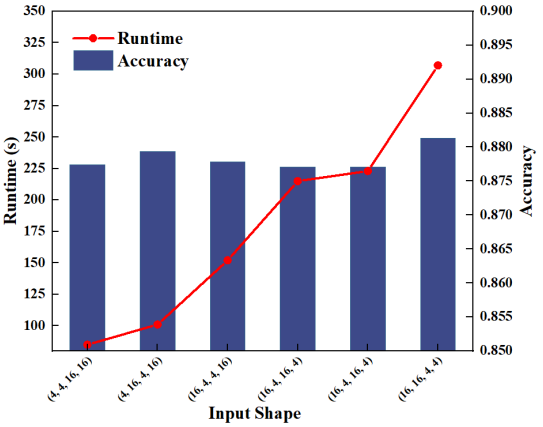

FIGURE 6. Different tensors representing the input.

Last, we consider the impact of different values of tensor-train factorization on the model performance. Herein, input tensor is selected for research.

As shown in Fig.7, when the distribution of the value in the decomposition is more scattered, the runtime is shorter, but the accuracy rate reduces obviously. When the distribution is tighter, the accuracy will be improved, but the runtime is prolonged. Therefore, we should balance the accuracy and runtime when we determine the decomposition methods of tensor-train factorization.

In summary, when tensor-train factorization is performed on the weight matrix $\boldsymbol{W}$, the performance is the best when the input tensor is arranged from small to large, and the output tensor is arranged from large to small, and the value of m_i and n_i cannot be too dispersed. Therefore, in the following experiments, we will follow the rules to apply tensor-train factorization in S2VT model.

B. DATASET AND SETTINGS

The dataset is from USTC-SLR [12], which is a public dataset. We choose 50 short sentences from it, each containing 5-7 isolated words, that are widely used in our daily

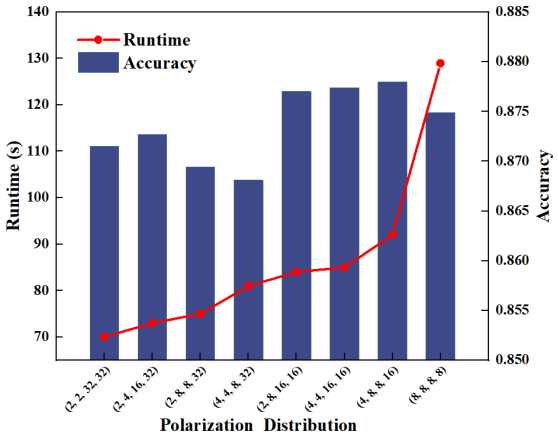

FIGURE 7. Influence of polarization distribution on results.

life. 20 signers play each word for 5 times, so each sentence has 100 samples, and the dataset consists of 5,000 samples. We divide it into 2 subsets, one for training and another for testing. In the training subsets, we randomly choose 16 signers from all the 20 signers in the dataset. The rest of the dataset is used as the testing subsets. Details of our datasets are shown in Table 1. The data is recorded by Kinect 2.0, and we can capture the color image, depth map, and skeleton joint location in realty. In this work, we ignore the trajectory of four skeleton joints and depth information, and only focus on exploring the color information. The features of the videos in the dataset are extracted through VGG16 [26]. It should be noted that, before feature extraction through VGG16 network, the original size of videos, 1280×720 is resized into 480×400 centering on signer, and then compressed to 224×224, which can reduce the information loss partly.

TABLE 1. The details of our dataset.

Dataset	Signs	Signers	Repetitions	Samples
Training	50	16	5	4000
Testing	50	4	5	1000

C. RESULTS AND ANALYSIS

In this paper, we proposed several tensor-train S2VT models for sign language recognition. As introduced in Section III, the S2VT model mainly contains three layers, namely the fully-connected layer (FC layer), the first LSTM layer (LSTM1) and the second LSTM layer (LSTM2), so we can represent the three layers with tensor-train format separately or together. Therefore, 6 scenarios are established to compare the performance of the 6 tensor-train S2VT models with the original S2VT model. It should be noted that the parameters of the tensor-train S2VT models are set according to the rules found in previous section.

6 scenarios are shown in Table 2. Scenario 1 replaces the FC layer in S2VT model with tensor-train layer. Scenario 2 and scenario 3 replace the LSTM1 layer and the LSTM2 layer with tensor-train LSTM respectively. Scenario 4 is to replace the two LSTM layers with tensor-train LSTM and scenario 5 is to replace the FC layer and the LSTM1 layer with tensor-train layer together. For scenario 6, the FC layer and two LSTM layers in S2VT model are formulated by tensor-train factorization.

TABLE 2. Correspondence between model and scenario.

Scenario	Model
1	TTFC+LSTM1+LSTM2
2	FC+TTLSTM1+LSTM2
3	FC+LSTM1+TTLSTM2
4	FC+TTLSTM1+TTLSTM2
5	TTFC+TTLSTM1+LSTM2
6	TTFC+TTLSTM1+TTLSTM2

As shown in Fig.8, the models of baseline and scenario2,3 and 5 converge faster and to a small value after 50 epochs. Compared to baseline, scenario 5 converges a little faster. However, the models of scenario 3,4 and 6 converge slowly, and the loss train of scenario 3 can converge to a smaller value after 90 epochs. Accordingly, when the FC layer and the LSTM1 are represented with tensor-train format, the convergence is better than the original S2VT model, while these models with the LSTM2 layer represented by tensor-train LSTM, perform worse, due to the slow convergence.

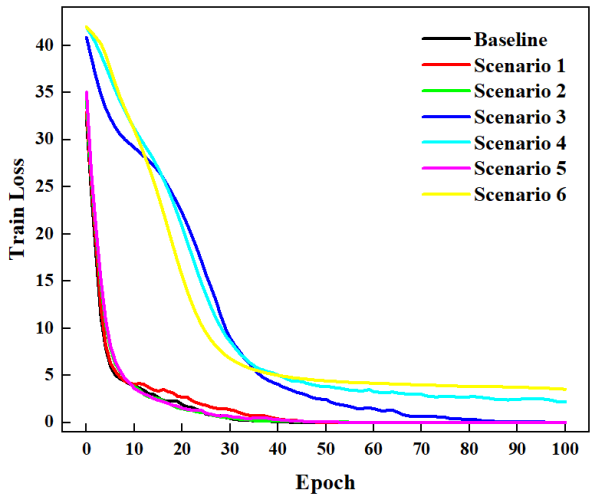

FIGURE 8. Training loss of various scenarios.

As shown in Table 3, we compare the parameters, memory and accuracy of 6 scenarios with that of the baseline systematically, and can claim that:

1) the parameters of tensor-train S2VT models can be reduced significantly. When the FC layer or the two-layer LSTM are represented by tensor-train

TABLE 3. Experimental Results on USTC-SLR Dataset. We report i) the number of parameters, ii) the memory of models, iii) the decrease of parameters iv) the accuracy.

	Parameters	Memory	Decrease	Accuracy
Baseline[6]	24,393,144	97.6MB	-	98.4%
Scenario-1	20,298.440	81.2MB	16.8%	98.0%
Scenario-2	16,395,224	65.6MB	32.8%	98.2%
Scenario-3	12,395,464	49.6MB	49.2%	91.9%
Scenario-4	4,397,544	17.6MB	81.9%	83.9%
Scenario-5	**12,300,520**	**49.2MB**	**49.5%**	**95.6%**
Scenario-6	302,840	1.2MB	98.8%	29.3%

format separately, the corresponding parameters of the model can be reduced by 16.8%, 32.8% and 49.2% respectively. When the three layers are represented with tensor-train factorization together, the number of parameters can be reduced to 302840, which is reduced by 98.8% compared with the original S2VT model. Correspondingly, the memory of tensor-train S2VT can also decrease significantly.

2) The accuracy of the original S2VT model was 98.4%. When the FC layer and the first LSTM layer in the S2VT model are represented with tensor-train format separately or together, the accuracies are similar to that of the original model basically. When the second LSTM layer is represented with tensor-train format, the accuracy decreased 6.5%. But, when two LSTM layers in S2VT are represented by tensor-train format, the accuracy is reduced obviously by 14.5%. The worse performance is in scenario 6, where all layers are represented by tensor-train format, which only has the accuracy of 29.3%.
3) To sum up, when the FC layer and the first LSTM layer in the S2VT model are transformed into tensor-train format, the accuracy of the model will remain high, and the parameters in the model will decrease significantly. However, the second LSTM layer in S2VT is not recommended to be transformed into tensor-train format, because it will result in a worse model performance in accuracy.

V. CONCLUSION

In this paper, we proposed to employ tensor-train factorization in S2VT to reduce the parameters. First, we explored the influence of parameters settings of tensor-train factorization on model performance, finding that the model performs best when the rank is 3, the input tensors are arranged from small to large, and the output tensors are from large to small, and the distribution cannot be too dispersed. We then apply tensor-train factorization in different layers of S2VT model for Chinese sign language recognition. The experimental result of 6 scenarios demonstrated that the parameters of the tensor-train S2VT models can be reduced significantly. Among them, the fully-connected layer and the first LSTM in S2VT model, represented with tensor-train format, has the similar accuracy compared to the original S2VT model, but the number of parameters is decreased by 49.5%. However, when the second LSTM layer of S2VT is expressed with tensor-train format, the convergence rate slows down significantly and the accuracy decreases greatly. Therefore, the tensor-train S2VT with the fully-connected layer and the first LSTM layer expressed with tensor-train format is the best model, which can remain similar accuracy and reduce parameters and memory significantly. This makes it possible to apply sign language recognition models to mobile devices without strict requirements in hardware devices, and making it easier for the hearing impaired to communicate with society and others. The proposed tensor-train S2VT models is also significant to other sequence-to-sequence problem and improve the performance.

In the future, we will try to apply tensor-train decomposition in other models for sign language. Also, we will also try to investigate the possibility of developing a novel end-to-end S2VT model, which do not need any CNN model to extract the features of videos. Instead, the images of videos will be directly feed into the S2VT through TTL.

REFERENCES

[1] [Online]. Available: https://www.thepaper.cn/newsDetail_forward_9383778
[2] U. Zeshan, "Sign language of the world," in *Encyclopedia of Language and Linguistics*, vol. 11. Amsterdam, The Netherlands: Elsevier, 2006, pp. 358–365.
[3] M. J. Cheok, Z. Omar, and M. H. Jaward, "A review of hand gesture and sign language recognition techniques," *Int. J. Mach. Learn. Cybern.*, vol. 10, no. 1, pp. 131–153, 2019.
[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, Jun. 2017.
[5] W. Zaremba, I. Sutskever, and O. Vinyals, "Recurrent neural network regularization," 2014, *arXiv:1409.2329*. [Online]. Available: http://arxiv.org/abs/1409.2329
[6] S. Venugopalan, M. Rohrbach, J. Donahue, R. Mooney, T. Darrell, and K. Saenko, "Sequence to sequence-video to text," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2015, pp. 4534–4542.
[7] C. Mao, S. Huang, X. Li, and Z. Ye, "Chinese sign language recognition with sequence to sequence learning," in *Proc. Chin. Conf. Comput. Vis. (CCF)*. Singapore: Springer, Oct. 2017, pp. 180–191.
[8] X. Li, C. Mao, S. Huang, and Z. Ye, "Chinese sign language recognition based on SHS descriptor and encoder-decoder LSTM model," in *Proc. Chin. Conf. Biometric Recognit.* Cham, Switzerland: Springer, Oct. 2017, pp. 719–728.
[9] S. Huang, C. Mao, J. Tao, and Z. Ye, "A novel Chinese sign language recognition method based on keyframe-centered clips," *IEEE Signal Process. Lett.*, vol. 25, no. 3, pp. 442–446, Mar. 2018.
[10] I. V. Oseledets, "Tensor-train decomposition," *SIAM J. Sci. Comput.*, vol. 33, no. 5, pp. 2295–2317, Jan. 2011.
[11] L. Pigou, S. Dieleman, P.-J. Kindermans, and B. Schrauwen, "Sign language recognition using convolutional neural networks," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, Sep. 2014, pp. 572–578.
[12] J. Huang, W. Zhou, H. Li, and W. Li, "Sign language recognition using 3D convolutional neural networks," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, Jun. 2015, pp. 1–6.
[13] S. Yang and Q. Zhu, "Video-based Chinese sign language recognition using convolutional neural network," in *Proc. IEEE 9th Int. Conf. Commun. Softw. Netw. (ICCSN)*, May 2017, pp. 929–934.
[14] L. Pigou, A. van den Oord, S. Dieleman, M. Van Herreweghe, and J. Dambre, "Beyond temporal pooling: Recurrence and temporal convolutions for gesture recognition in video," *Int. J. Comput. Vis.*, vol. 126, nos. 2–4, pp. 430–439, 2018.

[15] D. Wu, L. Pigou, P.-J. Kindermans, N. D.-H. Le, L. Shao, J. Dambre, and J.-M. Odobez, "Deep dynamic neural networks for multimodal gesture segmentation and recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 8, pp. 1583–1597, Aug. 2016.
[16] Y. Liao, P. Xiong, W. Min, W. Min, and J. Lu, "Dynamic sign language recognition based on video sequence with BLSTM-3D residual networks," *IEEE Access*, vol. 7, pp. 38044–38054, 2019.
[17] I. Papastratis, K. Dimitropoulos, D. Konstantinidis, and P. Daras, "Continuous sign language recognition through cross-modal alignment of video and text embeddings in a joint-latent space," *IEEE Access*, vol. 8, pp. 91170–91180, 2020.
[18] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[19] T. Liu, W. Zhou, and H. Li, "Sign language recognition with long short-term memory," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2016, pp. 2871–2875.
[20] A. Novikov, D. Podoprikhin, A. Osokin, and D. Vetrov, "Tensorizing neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 442–450.
[21] T. Garipov, D. Podoprikhin, A. Novikov, and D. Vetrov, "Ultimate tensorization: Compressing convolutional and FC layers alike," 2016, *arXiv:1611.03214*. [Online]. Available: http://arxiv.org/abs/1611.03214
[22] Y. Yang, D. Krompass, and V. Tresp, "Tensor-train recurrent neural networks for video classification," 2017, *arXiv:1707.01786*. [Online]. Available: http://arxiv.org/abs/1707.01786
[23] S. Samui, I. Chakrabarti, and S. K. Ghosh, "Tensor-train long short-term memory for monaural speech enhancement," 2018, *arXiv:1812.10095*. [Online]. Available: http://arxiv.org/abs/1812.10095
[24] X. Xu, Q. Wu, S. Wang, J. Liu, J. Sun, and A. Cichocki, "Whole brain fMRI pattern analysis based on tensor neural network," *IEEE Access*, vol. 6, pp. 29297–29305, 2018.
[25] C. Zou and F. Yang, "Deep learning approach based on tensor-train for sparse signal recovery," *IEEE Access*, vol. 7, pp. 34753–34761, 2019.
[26] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*. [Online]. Available: http://arxiv.org/abs/1409.1556
[27] Y. LeCun. (1998). *The MNIST Database of Handwritten Digits*. [Online]. Available: http://yann.lecun.com/exdb/mnist/

BIAO XU received the B.E. degree from the College of Mechanical and Electronic Engineering, North West Agriculture and Forestry University, Xianyang, China, in 2017. He is currently pursuing the master's degree with the School of Information Science and Technology, University of Science and Technology of China, Hefei, China. His research interests include model compression and deep learning for computer vision, especially the sign language recognition.

SHILIANG HUANG received the B.E. degree in electronic information science and technology from North China Electric Power University, Baoding, China, in 2011. He is currently pursuing the Ph.D. degree with the University of Science and Technology of China, Hefei, China. His research interests include image processing and video understanding, especially the sign language recognition, video caption, and action recognition.

ZHONGFU YE received the B.Eng. and M.S. degrees in electronic and information engineering from the Hefei University of Technology, Hefei, China, in 1982 and 1986, respectively, and the Ph.D. degree from the University of Science and Technology of China, Hefei, in 1995. He is currently a Professor with the University of Science and Technology of China. His current research interests include statistical and array signal processing, speech processing, sign language recognition, and hand pose estimation.