Accepted Manuscript

A multi-projection deep computation model for smart data in Internet of Things

Fanyu Bu, Xin Wang, Bo Gao

PII: S0167-739X(18)31939-3
DOI: https://doi.org/10.1016/j.future.2018.09.060
Reference: FUTURE 4495

To appear in: *Future Generation Computer Systems*

Received date : 13 August 2018
Revised date : 10 September 2018
Accepted date : 26 September 2018

A Multi-Projection Deep Computation Model for Smart Data in Internet of Things

Fanyu Bu[a,*], Xin Wang[b], Bo Gao[a]

[a]College of Computer and Information Management, Inner Mongolia University of Finance and Economics, Hohhot, China.
[b]Center of Information and Network Technology, Inner Mongolia Agricultural University, Hohhot, China.

Abstract

The double-projection deep computation model (DPDCM) proved to be effective for big data feature learning. However, DPDCM cannot capture the underlying correlations over the different modalities enough since it projects the input data into only two subspaces. To tackle this problem, this paper presents a multi-projection deep computation model (MPDCM) to generalize DPDCM for smart data in Internet of Things. Specially, MPDCM maps the input data into multiple nonlinear subspaces to learn the interacted features of IoT big data by substituting each hidden layer with a multi-projection layer. Furthermore, a learning algorithm based on back-propagation and gradient descent is designed to train the parameters of the presented model. Finally, extensive experiments are conducted on two representative datasets, i.e, Animal-20 and NUS-WIDE-14, to verify the presented model by comparing with DPDCM. Results show that the presented model achieves higher classification accuracy than DPDCM, proving the potential of the presented model to drill smart data for Internet of Things.

Keywords: Big data, Internet of Things, Smart data, Deep Computation Model, Back-propagation

1. Introduction

Recently, Internet of Things (IoT) have achieved great progress by integrating advanced sensing devices such as sensors and RFIDs into communication networks [1]. Specially, big data processing techniques such as data compression, deep learning, correlation analysis and clustering are playing a remarkable role in Internet of Things [2], [3]. For example, deep learning, an recently advanced artificial intelligence technique is used to find the valuable information, i.e., smart data, from IoT big data for smart market analysis in industrial manufacture. A unique property of IoT big data is its high variety, i.e., data comes from various sources such as cameras and sensors, with different formats like text, image and audio [4]. Typically, each heterogeneous data object has more than one modalities, implying that heterogeneous data is typically multi-modal [5]. For instance, a piece of video usually contains two modalities, i.e., image and audio, or three modalities, i.e., image, audio and text.

*Corresponding author: bufanyu@imufe.edu.cn.

Each modality of the multi-modal object shows the distinct information from one another, however, each modality has the close relation with others. The multi-modal property of heterogeneous data imposes a huge challenge on deep learning models for drilling smart data in IoT applications [6].

The first successfully trained deep learning model is the deep belief network which is constructed by several restricted boltzmann machines [7]. Over the last decade, some other deep learning models like stacked autoencoders and recursive neural networks have also been trained successfully [8]. Generally speaking, deep learning has more than one hidden layers and each hidden layer represents a layer of learned features of the input data. So, deep learning can learn multi-level features for the input data. Furthermore, deep learning enjoys its success in various applications like speech recognition and machine translation with a two-stage training policy, i.e., pre-training and fine-tanning [9], [10]. However, the traditional deep learning models are only suitable for single-modal data feature learning [11]. In other words, it is difficult for the traditional deep learning models to learn features for heterogeneous or multi-modal data. To tackle this issue, some multi-modal deep neural networks such as multi-modal deep boltzmann machines and multi-model deep learning were presented [12], [13]. Representative multi-modal deep neural networks first learn a joint representation for the multi-modal object by concatenating the features of each modality learned by a special deep learning model. Furthermore, they learn the features on the learned joint representation. Although the multi-modal deep neural networks have made some progress for heterogeneous data feature learning, they are also hard to capture the inherent correlations over different modalities by the means of linearly linking the learned features of each modality [11].

To address this problem, a deep computation model was presented to mine smart data in IoT applications. A deep computation model can be viewed as a generalization of a deep learning model for big data feature learning. Specially, a deep computation generalizes a stacked auto-encoder from the vector space to the high-order tensor space. In the deep computation model, each multi-modal data object is presented by a tensor while the tensor distance is utilized to define the objective function for capturing the inherent features of multi-modal data. More recently, a double-projection deep computation model (DPDCM) was presented to further generalize the deep computation model for big data feature learning by substituting each hidden layer with a double-projection layer [11]. DPDCM can effectively reveal the underlying correlations over different modalities by mapping the input data into two nonlinear subspaces. However, only two nonlinear subspaces are not enough for the deep computation model to capture the interacted features over different modalities.

Motivated by the neuroscience observation that the interacted inherent features of multi-modal data are generally hidden among different subspaces [15], the paper presents a multi-projection deep computation model (MPDCM) for smart data in IoT systems. Specially, MPDCM aims to generalize the double-projection deep computation model by the means of substituting each double-projection layer with a multi-projection layer. In detail, MPDCM first maps each multi-modal object into several different subspaces to reveal the features hidden in the different subspaces, and then learns the interacted inherent features to capture the underlying correlations by mapping the subspaces into the output via a weight tensor. To train the parameters of MPDCM, an equivalent alternative form of MPDCM is devised and accordingly an updating approach for the parameters based on back-propagation and gradient descent is implemented. Finally, MPDCM is verified on two representative datasets, namely Animal-20 and NUS-WIDE-14, by comparing with DPDCM regarding the classification accuracy in the experiments. Results imply that MPDCM can achieve higher classification accuracy than DPDCM, proving its potential for big data feature learning.

In summary, the contributions are three-fold:

- A multi-projection deep computation model is presented to generalize DPDCM for heterogeneous data feature learning by substituting each hidden layer of the deep computation model with a multi-projection layer. Specially, MPDCM is constructed by stacking several multi-projection tensor auto-encoders to learn hierarchical features for heterogeneous data.

- A multi-projection tensor auto-encoder (MPTAE) is devised to reveal the features hidden in the different subspaces by mapping each layer to several different nonlinear subspaces. Furthermore, MPTAE learns the interacted inherent features to capture the underlying correlations by mapping the subspaces into the output via a weight tensor.

- To train the parameters of each layer in MPDCM, an equivalent alternative form of MPTAE is devised and accordingly an updating approach for the parameters based on back-propagation and gradient descent is implemented in this paper.

The remainder of this paper is organized as follows. Section 2 provides preliminaries and Section 3 describes the details of the presented model. The learning algorithm for training the presented model is illustrated in Section 4 and the results are reported in Section 5. Section 6 reviews the related work and Section 7 concludes the paper.

2. Preliminaries

2.1. Deep Computation Model

The deep computation model is effective to abstract multi-level features for big data, especially for heterogeneous data, typically by stacking a couple of tensor auto-encoders, as presented in Figure 1 [14]. A tensor can be viewed as a multidimensional array in mathematics. For example, $R^{I_1 \times I_2 \times I_3}$ denotes a three-order tensor in which $I_i (i = 1, 2, 3)$ denotes the dimensionality of the i-th order.

In the tensor auto-encoder with the parameters θ, each original multi-modal object X is formatted as a tensor $X \in R^{I_1 \times I_2 \times \cdots \times I_N}$ and it is mapped into the M-order hidden space via an encoding function f_θ:

$$H_{j_1 \ldots j_M} = f_\theta\left(\sum_{i_1 \cdots i_N}^{I_1 \cdots I_N} W^{(1)}_{\alpha i_1 \cdots i_N} X_{i_1 \cdots i_N} + b^{(1)}_{j_1 \cdots j_M}\right), \tag{1}$$

where $\alpha = j_M + \sum_{t=1}^{M-1} (j_t - 1) \prod_{s=t+1}^{M} J_s$.

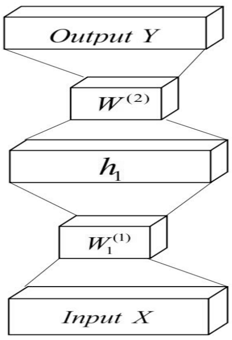
Afterwards, the hidden data H is reconstructed to the N-order output layer Y via the decoding function g_θ:

$$Y_{i_1 \ldots i_N} = g_\theta\left(\sum_{j_1 \cdots j_M}^{J_1 \cdots J_M} W^{(2)}_{\beta j_1 \cdots j_M} H_{j_1 \cdots j_M} + b^{(2)}_{i_1 \ldots i_N}\right), \tag{2}$$

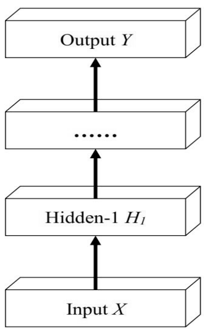
where $\beta = i_N + \sum_{t=1}^{N-1} (i_t - 1) \prod_{s=t+1}^{N} J_s$.

The objective function is defined based on the tensor distance regarding the object X as:

$$J(\theta; X) = \frac{1}{2}(Y - X)^T G (Y - X), \tag{3}$$



(a) Tensor auto-encoder



(b) Deep computation model

Figure 1: Architecture of the deep computation model.

where G denotes the coefficient matrix.

Moreover, the global objective function regarding the training set $X = \{X^{(1)}, X^{(2)}, ..., X^{(m)}\}$ is defined as:

$$J_{TAE}(\theta) = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{1}{2}(Y^{(i)} - X^{(i)})^T G(Y^{(i)} - X^{(i)})\right) + \frac{\lambda}{2}(W^{(1)2} + W^{(2)2}) \tag{4}$$

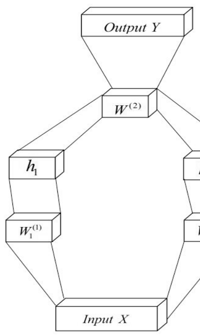
A couple of tensor auto-encoders could be stacked to a deep computation model as presented in Figure 1 to abstract multi-level features for smart data. The parameters of the deep computation model could pre-trained in a layer-wise manner from bottom to top first, followed by a fine-tuning step by a global high-order back-propagation from top to bottom after imposing the supervised labels. To enhance the training efficiency, several improved deep computation models have been devised by utilizing the tensor decomposition schemes like canonical polyadic decomposition and tensor-train to compress the parameters significantly [14], [16].

2.2. Double-Projection Deep Computation Model (DPDCM)

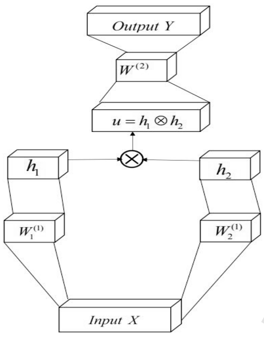
DPDCM attempts to learn the interacted inherent features for multi-modal objects. As the basic module, the double-projection tensor auto-encoder (DPTAE) maps each input object into two subspaces by substituting the hidden layer with a double-projection layer, as presented in Figure 2 [11].

From Figure 2(a), DPTAE maps the input X into two different subspaces, i.e., $h_1 \in R^{P_1 \times P_2 \times \cdots \times P_S}$ and $h_2 \in R^{Q_1 \times Q_2 \times \cdots \times Q_T}$, via f_θ:

$$H = \begin{pmatrix} h_1 \\ h_2 \end{pmatrix} = f_\theta\left(\begin{bmatrix} W_1^{(1)} \\ W_2^{(1)} \end{bmatrix} X + \begin{bmatrix} b_1^{(1)} \\ b_2^{(1)} \end{bmatrix}\right). \tag{5}$$



(a) Double-projection tensor auto-encoder



(b) An alternative form of DPTAE

Figure 2: Architecture of double-projection tensor auto-encoder.

Furthermore, DPTAE reconstructs the hidden representations to the output via g_θ:

$$y_{i_1\cdots i_N} = g_\theta(\sum_{p_1\cdots p_S}^{P_1\cdots P_S}\sum_{q_1\cdots q_T}^{Q_1\cdots Q_T} w^{(2)}_{p_1\cdots p_S,q_1\cdots q_T,i_1\cdots i_N} \cdot h_{1p_1\cdots p_S} \cdot h_{2q_1\cdots q_T} + b^{(2)}_{i_1\cdots i_N}). \tag{6}$$

To use back-propagation to train the parameters, an alternative form of DPTAE was provided via the tensor Kronecker product $\otimes$, as presented in Figure 2(b):

$$u = h_1 \otimes h_2, \tag{7}$$

where u can be viewed as the interactive features of h_1 and h_2.

Given two tensors, i.e., $A \in R^{I_1\times I_2\times\cdots\times I_N}$ and $B \in R^{J_1\times J_2\times\cdots\times J_N}$, their Kronecker product will produce $C = A \otimes B \in R^{I_1J_1\times I_2J_2\times\cdots\times I_NJ_N}$ with each entry $c_{\overline{i_1,j_1},\ldots,\overline{i_N,j_N}} = a_{i_1,\ldots,i_N} b_{j_1,\ldots,j_N}$ where $\overline{i_k,j_k} = j_k + (i_k - 1)J_k$. Eq. (8) shows an example of the Kronecker product of two matrices.

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \oplus \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} a_{11}b_{11} & a_{11}b_{12} & a_{12}b_{11} & a_{12}b_{12} \\ a_{11}b_{21} & a_{11}b_{22} & a_{12}b_{21} & a_{21}b_{22} \\ a_{21}b_{11} & a_{21}b_{12} & a_{22}b_{11} & a_{22}b_{12} \\ a_{21}b_{21} & a_{21}b_{22} & a_{22}b_{21} & a_{22}b_{22} \end{bmatrix}. \tag{8}$$

Based on Eq. (7), Eq. (6) can be rewritten as:

$$y_{i_1\cdots i_N} = g_\theta(\sum_{p_1\cdots p_S q_1\cdots q_T}^{P_1\cdots P_S Q_1\cdots Q_T} w^{(2)}_{p_1\cdots p_S q_1\cdots q_T,i_1\cdots i_N} \cdot u_{p_1\cdots p_S q_1\cdots q_T} + b^{(2)}_{i_1\cdots i_N}). \tag{9}$$

To simplify the training for the parameters, DPTAE defines the objective function regarding X as:

$$J_{DPTAE}(\theta; X) = \frac{1}{2}(Y - X)^T(Y - X)$$
$$= \frac{1}{2}\sum_{i_1}^{I_1}\cdots\sum_{i_N}^{I_N}(Y_{i_1\cdots i_N} - X_{i_1\cdots i_N})^2. \tag{10}$$

Moreover, the global objective function regarding the training set $X = \{X^{(1)}, X^{(2)}, ..., X^{(m)}\}$ is defined as:

$$J_{DPTAE}(\theta) = [\frac{1}{m}\sum_{i=1}^{m}(\frac{1}{2}(Y - X)^T(Y - X))] +$$
$$\frac{\lambda}{2}(\sum_{i_1\cdots i_N}^{I_1\cdots I_N}\sum_{p_1\cdots p_S}^{P_1\cdots P_S}(w^{(1)}_{1(i_1\cdots i_N, p_1\cdots p_S)})^2$$
$$+\sum_{i_1\cdots i_N}^{I_1\cdots I_N}\sum_{q_1\cdots q_T}^{Q_1\cdots Q_T}(w^{(1)}_{2(i_1\cdots i_N, q_1\cdots q_T)})^2$$
$$+\sum_{p_1\cdots p_S}^{P_1\cdots P_S}\sum_{q_1\cdots q_T}^{Q_1\cdots Q_T}\sum_{i_1\cdots i_N}^{I_1\cdots I_N}(w^{(2)}_{p_1\cdots p_S, q_1\cdots q_T, i_1\cdots i_N})^2). \tag{11}$$

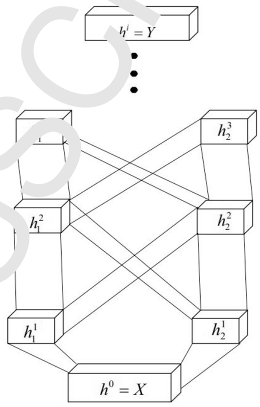
A couple of DPTAEs are stacked to a double-projection deep computation model described in Figure 3 [11].

3. Multi-Projection Deep Computation Model (MPDCM)

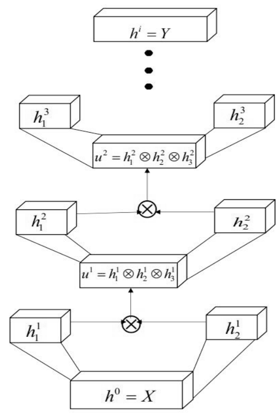
To learn the interactive inherent features for multi-modal objects, MPDCM generalizes DPDCM by substituting each hidden layer of DPDCM with a multi-projection layer. As the basic module of MPDCM, the multi-projection tensor auto-encoder is illustrated first, followed by the MPDCM model. Specially, this paper takes the triple-projection model for example to describe the model architecture and the learning algorithm.

Figure 4 shows the architecture of the multi-projection tensor auto-encoder.

Regarding the triple-projection tensor auto-encoder presented in Fig. 4(a), the input X is firstly mapped into three different nonlinear subspaces, i.e, $h_1 \in R^{O_1\times\cdots\times O_R}$, $h_2 \in$

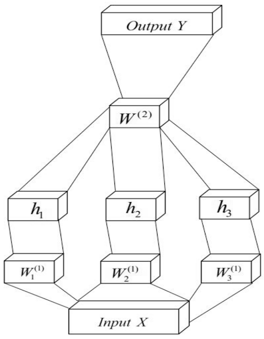


(a) Double-projection tensor auto-encoder (DPDCM)

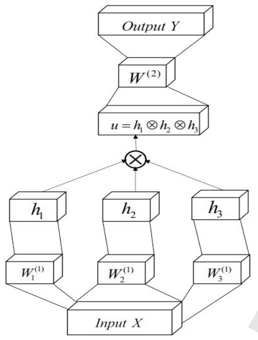


(b) An alternative form of D-PDCM

Figure 3: Architecture of double-projection deep computation model.



(a) Triple-projection tensor auto-encoder (TPTAE)



(b) An alternative form of TPTAE

Figure 4: Architecture of multi-projection tensor auto-encoder.

$R^{P_1\times\cdots\times P_S}$ and $h_3 \in R^{Q_1\times\cdots\times Q_T}$, in the hidden layer via the encoding function $f(x) = 1/(1+e^{-x})$:

$$H = \begin{pmatrix} h_1 \\ h_2 \\ h_3 \end{pmatrix} = f\left(\begin{bmatrix} W_1^{(1)} \\ W_2^{(1)} \\ W_3^{(1)} \end{bmatrix} X + \begin{bmatrix} b_1^{(1)} \\ b_2^{(1)} \\ b_3^{(1)} \end{bmatrix}\right) \tag{12}$$

where $W_1^{(1)} \in R^{I_1\times\cdots\times I_N\times O_1\times\cdots\times O_R}$, $W_2^{(1)} \in R^{I_1\times\cdots\times I_N\times P_1\times\cdots\times P_S}$ and $W_3^{(1)} \in R^{I_1\times\cdots\times I_N\times Q_1\times\cdots\times Q_T}$ represent the weigh tensors projecting X into three subspaces, respectively and three corresponding biases are $b_1^{(1)} \in R^{O_1\times\cdots\times O_R}$, $b_2^{(1)} \in R^{P_1\times\cdots\times P_S}$ and $b_3^{(1)} \in R^{Q_1\times\cdots\times Q_T}$, respectively.

Furthermore, TPTAE reconstructs the hidden features H into the output Y via an $(R+S+T+N)$-order weight $W^{(2)} \in R^{O_1\times\cdots\times O_R\times P_1\times\cdots\times P_S\times Q_1\times\cdots\times Q_T\times I_1\times\cdots\times I_N}$ and an N-order bias $b^{(2)} \in R^{I_1\times\cdots\times I_N}$:

$$y_{i_1\cdots i_N} = f\Big(\sum_{o_1\cdots o_S}^{O_1\cdots O_S}\sum_{p_1\cdots p_S}^{P_1\cdots P_S}\sum_{q_1\cdots q_T}^{Q_1\cdots Q_T} w^{(2)}_{o_1\cdots o_R p_1\cdots p_S q_1\cdots q_T i_1\cdots i_N} \\ h_{1(o_1\cdots o_R)}\cdot h_{2(p_1\cdots p_S)}\cdot h_{3(q_1\cdots q_T)} + b^{(2)}_{i_1\cdots i_N}\Big). \tag{13}$$

To train the parameters represented by θ, TPTAE defines the objective function regarding the training set as:

$$J_{TPTAE}(\theta) = \Big[\frac{1}{m}\sum_{i=1}^{m}\big(\tfrac{1}{2}(Y-X)^T(Y-X)\big)\Big] + \\ \frac{\lambda}{2}\Big(\sum_{i_1\cdots i_N}^{I_1\cdots I_N}\sum_{o_1\cdots o_R}^{O_1\cdots O_R}\big(w^{(1)}_{1(i_1\cdots i_N,p_1\cdots p_S)}\big)^2 \\ + \sum_{i_1\cdots i_N}^{I_1\cdots I_N}\sum_{p_1\cdots p_S}^{P_1\cdots P_S}\big(w^{(1)}_{2(i_1\cdots i_N,p_1\cdots p_S)}\big)^2 \\ + \sum_{i_1\cdots i_N}^{I_1\cdots I_N}\sum_{q_1\cdots q_T}^{Q_1\cdots Q_T}\big(w^{(1)}_{3(i_1\cdots i_N,q_1\cdots q_T)}\big)^2 \\ + \sum_{p_1\cdots p_S}^{P_1\cdots P_S}\sum_{q_1\cdots q_T}^{Q_1\cdots Q_T}\sum_{i_1\cdots i_N}^{I_1\cdots I_N}\big(w^{(2)}_{p_1\cdots p_S,q_1\cdots q_T,i_1\cdots i_N}\big)^2\Big). \tag{14}$$

Figure 4(b) gives an equivalent form of TPTAE via the tensor Kronecker product $\otimes$:

$$u = h_1 \otimes h_2 \otimes h_3, \tag{15}$$

where u can be viewed as interactive representation of three nonlinear subspaces. Therefore, Eq. (11) can be rewritten as:

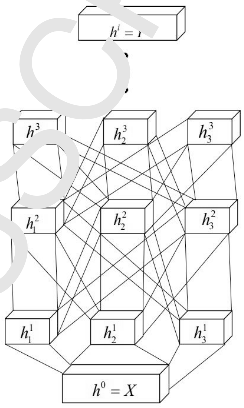
$$y_{i_1 \cdots i_N} = f\Big(\sum_{o_1 \cdots o_R p_1 \cdots p_S q_1 \cdots q_T}^{O_1 \cdots O_R P_1 \cdots P_S Q_1 \cdots Q_T} w^{(2)}_{o_1 \cdots o_R p_1 \cdots p_S q_1 \cdots q_T, i_1 \cdots i_N} \cdot u_{o_1 \cdots o_R p_1 \cdots p_S q_1 \cdots q_T} + b^{(2)}_{i_1 \cdots i_N}\Big). \tag{16}$$

With Eq. (16), back-propagation can be utilized to train the parameters, described in the following section.

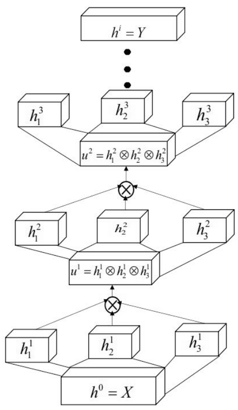
Figure 5 presents the architecture of a triple-projection deep computation model which is stacked by a couple of TPTAEs for big data multi–level feature learning.

Assume that $h^0 = X$, $h^i (0 < i < l)$ and $h^l = Y$ denote the input data, the i-th hidden layer and the output, respectively, presented in Fig. 5(a). TPDCM first maps the input data X to the first hidden layer h^1 which has three different subspaces, h_1^1, h_2^1 and h_3^1, via the encoding function, and then maps every subspace $h_j^1 (j = 1, 2, 3)$ to the second hidden layer h^2 which also has three different subspaces, h_1^2, h_2^2 and h_3^2. TPDCM repeats this link from bottom to top until h^{l-1} which represents the learned last level features of X. Finally, the three subspaces, h_1^{l-1}, h_2^{l-1} and h_3^{l-1}, are mapped to the output Y for the tasks of classification and recognition. Fig. 5(b) gives an alternative form of TPDCM in which the interactive inherent representation of each layer can be obtained via the tensor Kronecker product. For example, the i-layer interactive representation u_i of X can be obtained via:

$$u_i = h_1^i \otimes h_2^i \otimes h_3^i. \tag{17}$$



(a) Triple-projection deep computation model (TPDCM)



(b) An alternative form of TPDCM

Figure 5: Architecture of multi-projection deep computation model.

4. Learning Algorithm

In this work, a updating approach based on gradient descent is implemented to learn the parameters of TPTAE. Let ΔW and Δb denote the derivatives of $J_{TPTAE}(\theta; X)$ regarding the parameters $\theta = \{W_1^{(1)}, b_1^{(1)}, W_2^{(1)}, b_2^{(1)}, W_3^{(1)}, b_3^{(1)}; W^{(2)}, b^{(2)}\}$ for X. The average derivatives $\Delta\overline{W}$ and $\Delta\bar{b}$ of the global objective function $J_{TPTAE}(\theta)$ regarding all the training objects $\{X^{(1)}, X^{(2)}, \ldots, X^{(m)}\}$ can be obtained via:

$$\Delta\overline{W} = \frac{1}{m}\sum_{i=1}^{m}\Delta W_i + \lambda W. \tag{18}$$

$$\Delta\bar{b} = \frac{1}{m}\sum_{i=1}^{m}\Delta b_i. \tag{19}$$

Depending on gradient descent, the parameters θ can be updated via:

$$W \leftarrow W - \eta\Delta\overline{W}. \tag{20}$$

$$b \leftarrow b - \eta\Delta\bar{b}. \tag{21}$$

Therefore, the key for updating the parameters is to compute the partial derivatives ΔW and Δb of the objective function regarding each training object. In this paper, back-propagation is extended to the tensor space for computing ΔW and Δb. To this end, the variables $z_1^{(2)}$, $z_2^{(2)}$, $z_3^{(2)}$ and $z^{(3)}$ are introduced to describe the forward-propagation of TPTAE as:

$$z^{(2)}_{1(o_1\ldots o_R)} = \sum_{i_1\cdots i_N}^{I_1\cdots I_N} W^{(1)}_{o_1\ldots o_R i_1\cdots i_N} X_{i_1\cdots i_N} + b^{(1)}_{i_1\cdots i_N}, \tag{22}$$

$$h_{1(o_1\ldots o_R)} = f(z^{(2)}_{1(o_1\ldots o_R)}), \tag{23}$$

$$z^{(2)}_{2(p_1\ldots p_S)} = \sum_{i_1\cdots i_N}^{I_1\cdots I_N} W^{(1)}_{p_1\ldots p_S i_1\cdots i_N} X_{i_1\cdots i_N} + b^{(1)}_{i_1\cdots i_N}, \tag{24}$$

$$h_{2(p_1\ldots p_S)} = f(z^{(2)}_{2(p_1\ldots p_S)}), \tag{25}$$

$$z^{(2)}_{3(q_1\ldots q_T)} = \sum_{i_1\cdots i_N}^{I_1\cdots I_N} W^{(1)}_{q_1\ldots q_T i_1\cdots i_N} X_{i_1\cdots i_N} + b^{(1)}_{i_1\cdots i_N}, \tag{26}$$

$$h_{3(q_1\ldots q_T)} = f(z^{(2)}_{3(q_1\ldots q_T)}), \tag{27}$$

$$z^{(3)}_{i_1\cdots i_N} = \sum_{o_1\cdots o_S}^{O_1\cdots O_S}\sum_{p_1\cdots p_S}^{P_1\cdots P_S}\sum_{q_1\cdots q_T}^{Q_1\cdots Q_T} w^{(2)}_{o_1\cdots o_R p_1\cdots p_S q_1\cdots q_T i_1\cdots i_N}, \\ h_{1(o_1\cdots o_R)} \cdot h_{2(p_1\cdots p_S)} \cdot h_{3(q_1\cdots q_T)} + b^{(2)}_{i_1\cdots i_N} \tag{28}$$

$$y_{i_1\cdots i_N} = f(z^{(3)}_{i_1\cdots i_N}). \tag{29}$$

The following four steps describe the computation of the partial derivatives ΔW and Δb.

Step 1. Compute "error term" $\sigma^{(3)}$ via Eq.(30).

$$\begin{aligned}\sigma^{(3)}_{i_1\cdots i_N} &= \tfrac{\partial J_{TPTAE}(\theta;X)}{\partial z^{(3)}_{i_1\cdots i_N}} \\ &= \tfrac{\partial}{\partial z^{(3)}_{i_1\cdots i_N}}[\tfrac{1}{2}\sum_{i_1}^{I_1}\cdots\sum_{i_N}^{I_N}(y_{i_1\cdots i_N} - x_{i_1\cdots i_N})^2] \\ &= f'(z^{(3)}_{i_1\cdots i_N})(y_{i_1\cdots i_N} - x_{i_1\cdots i_N}) \\ &= y_{i_1\cdots i_N}(1 - y_{i_1\cdots i_N})(y_{i_1\cdots i_N} - x_{i_1\cdots i_N}).\end{aligned} \tag{30}$$

Step 2. Compute "error term" $\sigma^{(2)}$ via:

$$\sigma^{(2)}_{1(o_1\cdots o_R)} = \frac{\partial J_{TPTAE}(\theta;X)}{\partial z^{(2)}_{1(o_1\cdots o_R)}} = \sum_{i_1\cdots i_N}^{I_1\cdots I_N} \frac{\partial J_{TPTAE}(\theta;X)}{\partial z^{(3)}_{i_1\cdots i_N}} \cdot \frac{\partial z^{(3)}_{i_1\cdots i_N}}{\partial z^{(2)}_{1(o_1\cdots o_R)}} = \sum_{i_1\cdots i_N}^{I_1\cdots I_N} (\sigma^{(3)}_{i_1\cdots i_N} \sum_{q_1\cdots q_T}^{Q_1\cdots Q_T} \sum_{p_1\cdots p_S}^{P_1\cdots P_S} w^{(2)}_{o_1\cdots o_R p_1\cdots p_S q_1\cdots q_T i_1\cdots i_N} \cdot h_{3(q_1\cdots q_T)} \cdot h_{2(p_1\cdots p_S)} h_{1(o_1\cdots o_R)}(1 - h_{1(o_1\cdots o_R)})), \tag{31}$$

$$\sigma^{(2)}_{2(p_1\cdots p_S)} = \frac{\partial J_{TPTAE}(\theta;X)}{\partial z^{(2)}_{2(p_1\cdots p_S)}} = \sum_{i_1\cdots i_N}^{I_1\cdots I_N} \frac{\partial J_{TPTAE}(\theta;X)}{\partial z^{(3)}_{i_1\cdots i_N}} \cdot \frac{\partial z^{(3)}_{i_1\cdots i_N}}{\partial z^{(2)}_{2(p_1\cdots p_S)}} = \sum_{i_1\cdots i_N}^{I_1\cdots I_N} (\sigma^{(3)}_{i_1\cdots i_N} \sum_{q_1\cdots q_T}^{Q_1\cdots Q_T} \sum_{o_1\cdots o_S}^{O_1\cdots O_S} w^{(2)}_{o_1\cdots o_R p_1\cdots p_S q_1\cdots q_T i_1\cdots i_N} \cdot h_{3(q_1\cdots q_T)} \cdot h_{2(p_1\cdots p_S)} h_{1(o_1\cdots o_R)}(1 - h_{2(p_1\cdots p_S)})), \tag{32}$$

$$\sigma^{(2)}_{3(q_1\cdots q_T)} = \frac{\partial J_{TPTAE}(\theta;X)}{\partial z^{(2)}_{3(q_1\cdots q_T)}} = \sum_{i_1\cdots i_N}^{I_1\cdots I_N} \frac{\partial J_{TPTAE}(\theta;X)}{\partial z^{(3)}_{i_1\cdots i_N}} \cdot \frac{\partial z^{(3)}_{i_1\cdots i_N}}{\partial z^{(2)}_{3(q_1\cdots q_T)}} = \sum_{i_1\cdots i_N}^{I_1\cdots I_N} (\sigma^{(3)}_{i_1\cdots i_N} \sum_{p_1\cdots p_T}^{P_1\cdots P_T} \sum_{o_1\cdots o_S}^{O_1\cdots O_S} w^{(2)}_{o_1\cdots o_R p_1\cdots p_S q_1\cdots q_T i_1\cdots i_N} \cdot h_{3(q_1\cdots q_T)} \cdot h_{2(p_1\cdots p_S)} h_{1(o_1\cdots o_R)}(1 - h_{3(q_1\cdots q_T)})). \tag{33}$$

Step 3. Compute $\frac{\partial z^{(l+1)}}{\partial W^{(l)}}(l = 1, 2)$ via:

$$\frac{\partial z^{(3)}_{i_1\cdots i_N}}{\partial w^{(2)}_{o_1\cdots o_R p_1\cdots p_S q_1\cdots q_T i_1\cdots i_N}} = \frac{\partial}{\partial w^{(2)}_{o_1\cdots o_R p_1\cdots p_S q_1\cdots q_T i_1\cdots i_N}} (\sum_{o_1\cdots o_S}^{O_1\cdots O_S} \sum_{p_1\cdots p_S}^{P_1\cdots P_S} \sum_{q_1\cdots q_T}^{Q_1\cdots Q_T} w^{(2)}_{o_1\cdots o_R p_1\cdots p_S q_1\cdots q_T i_1\cdots i_N} h_{1(o_1\cdots o_R)} \cdot h_{2(p_1\cdots p_S)} \cdot h_{3(q_1\cdots q_T)} + b^{(2)}_{i_1\cdots i_N}) = h_{1(o_1\cdots o_R)} \cdot h_{2(p_1\cdots p_S)} \cdot h_{3(q_1\cdots q_T)}, \tag{34}$$

$$\frac{\partial z^{(2)}_{1(o_1\cdots o_R)}}{\partial w^{(1)}_{1(i_1\cdots i_N o_1\cdots o_R)}} = X_{i_1\cdots i_N} \tag{35}$$

$$\frac{\partial z^{(2)}_{2(p_1\cdots p_S)}}{\partial w^{(1)}_{2(i_1\cdots i_N p_1\cdots p_S)}} = X_{i_1\cdots i_N}, \tag{36}$$

$$\frac{\partial z^{(2)}_{3(q_1\cdots q_T)}}{\partial w^{(1)}_{3(i_1\cdots i_N q_1\cdots q_T)}} = X_{i_1\cdots i_N}. \tag{37}$$

Step 4. Compute ΔW and Δb according to the chain rule:

$$\Delta W^{(1)}_{1(i_1\cdots i_N o_1\cdots o_R)} = \sigma^{(2)}_{1(o_1\cdots o_R)} \cdot X_{i_1\cdots i_N}, \tag{38}$$

$$\Delta W^{(1)}_{2(i_1\cdots i_N p_1\cdots p_S)} = \sigma^{(2)}_{2(p_1\cdots p_S)} \cdot X_{i_1\cdots i_N}, \tag{39}$$

$$\Delta W^{(1)}_{3(i_1\cdots i_N q_1\cdots q_T)} = \sigma^{(2)}_{1(q_1\cdots q_T)} \cdot X_{i_1\cdots i_N}, \tag{40}$$

$$\Delta b^{(1)}_{1(o_1\cdots o_R)} = \sigma^{(2)}_{1(o_1\cdots o_R)}, \tag{41}$$

$$\Delta b^{(1)}_{2(p_1\cdots p_S)} = \sigma^{(2)}_{2(p_1\cdots p_S)}, \tag{42}$$

$$\Delta b^{(1)}_{3(q_1\cdots q_T)} = \sigma^{(2)}_{3(q_1\cdots q_T)}, \tag{43}$$

$$\Delta w^{(2)}_{o_1\cdots o_R p_1\cdots p_S q_1\cdots q_T i_1\cdots i_N} = \sigma^{(3)}_{i_1\cdots i_N} \cdot h_{1(o_1\cdots o_R)} \cdot h_{2(p_1\cdots p_S)} \cdot h_{3(q_1\cdots q_T)}, \tag{44}$$

$$\Delta b^{(2)}_{i_1\cdots i_N} = \sigma^{(3)}_{i_1\cdots i_N}. \tag{45}$$

In summary, the updating approach for training TPTAE is described in Algorithm 1.

From Algorithm 1, the computational cost of the updating approach is dominated by forward-propagation and back-propagation. Let $I = max\{I_1, \ldots, I_N\}$, $U = max\{R, S, T\}$, and $V = max\{O_1, \ldots, O_R, P_1, \ldots, P_S, Q_1 \ldots, Q_T\}$. During each iteration, the forward-propagation has a computational cost of

Algorithm 1: Learning Algorithm for Training TPTAE.

Input: $\{X^{(i)}\}_{i=1}^{m}, \eta, \lambda, threshold$
Output: $\theta = \{W_1^{(1)}, b_1^{(1)}, W_2^{(1)}, b_2^{(1)}, W_3^{(1)}, b_3^{(1)}; W^{(2)}, b^{(2)}\}$

for $example = 1, 2, ..., m$ **do**
- **for** $o_l = 1, \ldots, o_r (s = 1, \ldots, R)$ **do**
  - Compute $z^{(2)}_{1(o_1 \ldots o_R)}$;
  - Compute $h_{1(o_1 \ldots o_R)}$;
- **for** $p_l = 1, \ldots, p_s (s = 1, \ldots, S)$ **do**
  - Compute $z^{(2)}_{2(p_1 \ldots p_S)}$;
  - Compute $h_{2(p_1 \ldots p_S)}$;
- **for** $q_l = 1, \ldots, q_t (t = 1, \ldots, T)$ **do**
  - Compute $z^{(2)}_{3(q_1 \ldots q_T)}$;
  - Compute $h_{3(q_1 \ldots q_T)}$;
- **for** $i_k = 1, \ldots, I_k (k = 1, \ldots, N)$ **do**
  - Compute $z^{(3)}_{i_1 \cdots i_N}$;
  - Compute $Y_{i_1 \ldots i_N}$;
- **if** $J_{DPTAE}(\theta) > threshold$ **then**
  - **for** $i_k = 1, \ldots, I_k (k = 1, \ldots, N)$ **do**
    - Compute $\sigma^{(3)}_{i_1 \cdots i_N}$;
  - **for** $o_l = 1, \ldots, o_r (s = 1, \ldots, R)$ **do**
    - Compute $\sigma^{(2)}_{1(o_1 \cdots o_R)}$;
  - **for** $p_l = 1, \ldots, p_s (s = 1, \ldots, S)$ **do**
    - Compute $\sigma^{(2)}_{2(p_1 \cdots p_S)}$;
  - **for** $q_l = 1, \ldots, q_t (t = 1, \ldots, T)$ **do**
    - Compute $\sigma^{(2)}_{3(q_1 \cdots q_T)}$;
  - **for** $i_k = 1, \ldots, I_k (k = 1, \ldots, N)$ **do**
    - Compute $\Delta b^{(2)}_{i_1 \ldots i_N}$;
    - **for** $o_l = 1, \ldots, o_r (s = 1, \ldots, R)$ **do**
      - **for** $p_l = 1, \ldots, p_s (s = 1, \ldots, S)$ **do**
        - **for** $q_l = 1, \ldots, q_t (t = 1, \ldots, T)$ **do**
          - Compute $\Delta w^{(2)}_{o_1 \cdots o_R p_1 \cdots p_S q_1 \cdots q_T i_1 \cdots i_N}$;

$O(I^N V^{3U})$ while the back-propagation has the same computational cost as the forward-propagation. So, the overall computational cost of Algorithm 1 is approximately $O(kI^N V^{3U})$ with k denoting the number of iterations. Note that N and U are typically very small constant positive integers. Once the architecture of TPTAE is fixed, the computational cost of the updating approach is polynomial regarding I and V. Moreover, Algorithm 3 is easily extended to update the parameters of multi-projection tensor auto-encoder for obtaining a multi-projection deep computation model.

5. Experiments

In this work, the presented model (MPDCM) is verified on two highly heterogeneous datasets, Animal-20 and NUS-WIDE-14, by comparing with DPDCM in the experiments.

5.1. Results on Animal-20

Animal-20 is a subset of Animal [11]. Specially, it has 20 groups and totally 12 000 objects. In this work, 9000 objects are randomly chose as the training set and the rest are used as the testing set. In the experiments, each object in Animal-20 is formatted as a 3-order tensor $R^{64 \times 64 \times 20}$, so the input of each model is a $R^{64 \times 64 \times 20}$ tensor.

First, the MPTAE is compared with DPTAE regarding classification accuracy on Animal-20. Each model is performed for 5 times to verify the robustness, each with random initialization. The results are listed in Table 1.

According to the results listed in Table 1, the classification accuracy is gradually improved when the number of the subspaces increases. For instance, when the number of subspaces increases from 2 to 3, the average classification accuracy is improved from 42.8% to 46.2%. Such results clearly imply that increasing the

Table 1: Classification results of various tensor auto-encoders on Animal-20.

Model	1	2	3	4	5
DPTAE	0.42	0.41	0.49	0.45	0.37
TPTAE	0.45	0.49	0.51	0.45	0.41
QPTAE	0.51	0.50	0.55	0.47	0.48
FPTAE	0.51	0.51	0.55	0.46	0.48

number of subspaces can improve the learning performance of the tensor auto-encoder for heterogeneous data. Furthermore, when the number of subspaces is more than 4, the classification accuracy keeps unchangeable on Animal-20. Specially, QPTAE produces almost the same classification results as FPTAE.

Next, MPDCM is compared with DPDCM on Animal-20. Table 2 shows the classification results.

Table 2: Classification results of various tensor auto-encoders on Animal-20.

Model	1	2	3	4	5
DPDCM	0.66	0.64	0.67	0.69	0.67
TPDCM	0.69	0.71	0.67	0.72	0.68
QPDCM	0.73	0.72	0.69	0.75	0.71
FPDCM	0.74	0.72	0.71	0.75	0.71

Table indicates three important observations. First, with the growing number of subspaces, the classification accuracy of various deep computation models increases. For instance, QPDCM achieves significantly higher classification accuracy than TPDCM on Animal-20, implied by the fact that they yield the average classification accuracy of 72% and 69.4%, respectively. Second, the deep computation model performs significantly better than the corresponding tensor auto-encoder when they have the same number of subspaces. Specially, QPDCM and QPTAE produce the average classification results of 72% and 50.2%, respectively. Finally, as the number of subspaces increases to 4, the multi-projection deep computation models produce almost the same classification accuracy. Such observations prove the effectiveness of the multi-projection deep computation model for heterogenous data feature learning.

Table 3 and Table 4 show the average training time of each model.

Table 3: Average training time (Minutes) of various tensor auto-encoders on Animal-20.

DPTAE	TPTAE	QPTAE	FPTAE
13.37	16.48	15.29	15.81

Table 4: Average training time (Minutes) of various deep computation models on Animal-20.

DPDCM	TPDCM	QPDCM	FPDCM
152.26	148.54	167.85	166.93

Table 3 shows that various tensor auto-encoders take almost the same time to train the parameters while different multi-projection deep computation models spend almost the same time in training the parameters despite the different numbers of subspaces. This is because various deep computation models have the same number of hidden units and they all use the extended back-propagation to train parameters, resulting in almost the same computational cost.

5.2. Results on NUS-WIDE-14

NUS-WIDE-14 is a subset of NUS-WIDE and it has 20 000 objects, fallen into 14 groups [11]. Specially, 15 000 objects are chosen to train the parameters and the rest are utilized to test the performance regarding the classification accuracy. In this work, each object in

NUS-WIDE-14 is formatted as a 3-order tensor $R^{192\times192\times24}$, so the input of various models is a $R^{192\times192\times24}$ tensor.Table 5 and Table 6 show the classification results of various models on NUS-WIDE-14.

Table 5: Classification results of various tensor auto-encoders on NUS-WIDE-14.

Model	1	2	3	4	5
DPTAE	0.66	0.62	0.61	0.65	0.58
TPTAE	0.69	0.65	0.66	0.65	0.62
QPTAE	0.71	0.74	0.69	0.71	0.73
FPTAE	0.72	0.75	0.69	0.71	0.72

Table 6: Classification results of various deep computation models on NUS-WIDE-14.

Model	1	2	3	4	5
DPDCM	0.79	0.76	0.81	0.79	0.73
TPDCM	0.81	0.79	0.83	0.79	0.76
QPDCM	0.82	0.84	0.86	0.81	0.79
FPDCM	0.83	0.84	0.86	0.80	0.81

From Table 5 and Table 6, each multi-projection deep computation model achieves significantly higher accuracy than the corresponding multi-projection tensor auto-encoders with the same number of subspaces. For example, QPDCM and QPTAE produce the average classification accuracy of 82.4% and 71.6%, respectively. Such results argue that the multi-projection deep computation model is more effective than the multi-projection tensor auto-encoder for heterogeneous data feature learning. Moreover, the MPDCM models outperform DPDCM in terms of classification accuracy on NUS-WIDE-14. For example, FPDCM produces the average accuracy of 82.8% while DPDCM yields the average accuracy of 77.6%. Such experimental results demonstrate the effectiveness of the presented generalized multi-projection deep computation models.

The average training time of each model is shown in Table 7 and Table 8.

Table 7: Average training time (Minutes) of various tensor auto-encoders on NUS-WIDE-14.

DPTAE	TPTAE	QPTAE	FPTAE
37.28	[illegible]	51.06	45.59

Table 8: Average training time (Minutes) of various deep computation models on NUS-WIDE-14.

DPDCM	TPDCM	QPDCM	FPDCM
259.46	287.32	261.53	271.48

Although the multi-projection deep computation models have more subspaces than DPDCM in each hidden layer, the MPDCM models are not significantly more time-consuming than DPDCM for training the parameters since they have almost the same computational cost due to the same number of the hidden units.

6. Related Work

In the past few years, a couple of multi-model deep neural networks have been investigated to learn multi-level features for big data, especially for heterogeneous data. The fist multi-model deep neural network was investigated by Ngiam et al. to learn features over two different modalities, i.e., audio and video [12].

Two sparse boltzmann machines are firstly constructed to learn features for audio and video separately, and then the learned features are linked linearly as the shared representation of two modalities. Furthermore, a belief neural network is constructed to learn features on the shared representation.

Another typical multi-modal deep neural network is the multimodal learning model with deep boltzmann machines for text-image bimodal learning [13]. Specially, an image-specific deep belief network and a text-specific deep belief network are constructed to learn features for image and text separately. Furthermore, the shared representation is obtained by means of linking the learned features to model the joint distribution over the image modality and the text modality.

More recently, a multi-model convolutional neural network model was investigated to learn the answer to the visual question [17]. In this model, two separate convolutional neural networks are built to encoder the image content and to produe the question representation, respectively. Moreover, another convolutional neural network is built to yield the shared representation by linking the image features and the question features. Furthermore, the shared representation is input to the softmax layer to generate the answer.

In addition, a stochastic long short-term memory is presented to perform the uncertainty backward propagation. Other representative multi-modal deep neural networks include the multi-modal residual learning [18] and the hierarchical multi-modal long short term memory [19] and so on [20].

7. Conclusion

In this study, a multi-projection deep computation model is investigated for Industrial heterogeneous big data feature learning. One key property of the investigated model is to project each input object into multiple nonlinear subspaces to capture the underlying features hidden in the different spaces. Furthermore, the interactive inherent features of the heterogeneous object can be learned by using the Kronecker product to fuse the features in the different subspaces. Experiments are carried out to compare the multi-projection deep computation models with the different number of subspaces. The results clearly imply that the quadruple-projection model and the fivefold-projection model perform significantly better than the double-projection model on Animal-20 and NUS-WIDE-14, proving the effectiveness of the investigated model to generalize the double-projection deep computation model. Actually, the double-projection deep computation model can be seen as a specific example of the investigated model. Therefore, the presented model is potential for industrial big data feature learning.

Although two representative heterogeneous datasets, namely Animal-20 and NUS-WIDE-14, have preliminarily proved the effectiveness of the presented model for heterogenous data learning in the experiments, they are not large enough. In the future work, the presented model will be further verified on industrial larger heterogeneous datasets. Moreover, the classification results are slightly fluctuant because of the effect of initial parameters. Therefore, the advanced initialization methods will be investigated to improve the performance of the presented model in the future work.

8. Acknowledge

This study was supported by four foundations including No. 201605020, No. 61762068, No. 2017MS0610 and No. NJZY16145.

References

[1] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, M. Ayyash, Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications, IEEE Communications Surveys and Tutorials 17(4)(2015) 2347-2376.

[2] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, W. Zhao, A Survey on Internet of Things: Architecture, Enabling Technologies, Security and Privacy, and Applications, IEEE Internet of Things Journal 4(5)(2017) 1125-1142.

[3] Q. Zhang, L. T. Yang, Z. Chen, P. Li, F. Bu, An Adaptive Dropout Deep Computation Model for Industrial IoT Big Data Learning with Crowdsourcing to Cloud Computing, IEEE Transactions on Industrial Informatics 2018, 10.1109/TII.2018.2791424.

[4] M. Z. A. Bhuiyan, G. Wang, J. Wu, J. Cao, Dependable Structural Health Monitoring Using Wireless Sensor Networks, IEEE Transactions on Dependable and Secure Systems 14(4)(2017) 363-376.

[5] A. Fahad, N. Alshartri, Z. Tari, A. Alamri, I. Khalil, A. Y. Zomaya, S. Foufou, A. Bouras, A Survey of Clustering Algorithms for Big Data: Taxonomy and Empirical Analysis, IEEE Transactions on Emerging Topics in Computing 2(3)(2014) 267-279.

[6] Q. Zhang, L. T. Yang, Z. Chen, P. Li, Secure Weighted Possibilistic c-Means Algorithm on Cloud for Clustering Big Data, Information Sciences, 2018, 10.1016/j.ins.2018.02.013.

[7] G. E. Hinton, R. R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science 313(5786)(2006) 504-507.

[8] Y. LeCun, Y. Bengio, G. Hinton, Deep Learning, Nature 521(7553)(2015) 436-444.

[9] Q. Zhang, L. T. Yang, Z. Yan, Z. Chen, P. Li, An Efficient Deep Learning Model to Predict Cloud Workload for Industry Informatics, IEEE Transactions on Industrial Informatics 2018, 10.1109/TII.2018.2808910.

[10] J. Schmidhuber, Deep Learning in Neural Networks: An Overview, Neural networks 61(2015) 85-117.

[11] Q. Zhang, L. T. Yang, Z. Chen, P. Li, M. J. Deen, Privacy-preserving Double-projection Deep Computation Model with Crowdsourcing on Cloud for Big Data Feature Learning, IEEE Internet of Things Journal 5(4)(2018) 2896-2903.

[12] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, A. Y. Ng, Multimodal Deep Learning, in: Proceedings of the 28th International Conference on Machine Learning, 2011, 689-696.

[13] N. Srivastava, R. Salakhutdinov, Multimodal Learning with Deep Boltzmann Machines, in: Advances in Neural Information Processing Systems, 2012, 2222-2230.

[14] Q. Zhang, L. T. Yang, Z. Chen, P. Li, A Tensor-train Deep Computation Model for Industry Informatics Big Data Feature Learning, IEEE Transactions on Industrial Informatics 2018, DOI: 10.1109/TII.2018.2791423.

[15] D. Yu, L. Deng, F. Seide, The Deep Tensor Neural Network with Applications to Large Vocabulary Speech Recognition, IEEE Transactions on Audio, Speech, and Language Processing 21(2)(2013) 388-396.

[16] Q. Zhang, L. T. Yang, Z. Chen, P. Li, An Improved Deep Computation Model Based on Canonical Polyadic Decomposition, IEEE Transactions on Systems, Man, and Cybernetics: Systems 2017, DOI: 10.1109/TSMC.2017.2701797.

[17] L. Ma, Z. Lu, H. Li, Learning to Answer Questions from Image Using Convolutional Neural Network, in: Proceedings of Association for the Advancement of Artificial Intelligence, 2016, 3567-3573.

[18] J. Kim, S. Lee, D, Kwak, M. Heo, Multimodal Residual Learning for Visual QA, in: Proceedings of Neural Information Processing Systems, 2016, pp. 361-369.

[19] Z. Niu, M. Zhou, L. Wang, X. Gao, G. Hua, Hierarchical Multimodal LSTM for Dense Visual-Semantic Embedding, in: Proceedings of IEEE International Conference on Computer Vision, 2017, 1899-1907.

[20] L. Ma, Z. Chen, L. Xu, Y. Yan, Multimodal Deep Learning for Solar Radio Burst Classification, Pattern Recognition 61(2017) 573-582.

Fanyu Bu received the Bachelor's degree in computer science from Inner Mongolia Agricultural University, Hohhot, China, in 2003, and the Master's degree in computer application from Inner Mongolia University, Hohhot, China, in 2009. He got the PhD degree in computer application technology from Dalian University of Technology, Dalian, China, in 2018. He is currently an assistant professor at Department of Biomedical Informatics in Inner Mongolia University of Finance and Economics, China. His research interests include Big Data and Internet of Things.

ACCEPTED MANUSCRIPT

A multi-projection deep computation model is presented to generalize DPDCM for smart data in Internet of Things.
A learning algorithm based on back-propagation and gradient descent is designed to train the parameters of the presented model
The extensive experiments are conducted to evaluate the performance of the proposed scheme by comparing with DPDCM.