IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 13, 2018, accepted November 22, 2018, date of publication November 28, 2018,
date of current version December 31, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2883666

Online Real-Time Analysis of Data Streams Based on an Incremental High-Order Deep Learning Model

YULIANG LI, MIN ZHANG, AND WEI WANG (Member, IEEE)
College of Electronics and Communication Engineering, Tianjin Normal University, Tianjin 300387, China

Corresponding author: Wei Wang (weiwang@tjnu.edu.cn)

This work was supported in part by the Natural Youth Science Foundation of China under Grant 61501326 and Grant 61401310, in part by the National Natural Science Foundation of China under Grant 61731006, in part by the Natural Science Foundation of China under Grant 61271411, in part by the Tianjin Research Program of Application Foundation and Advanced Technology under Grant 15JCZDJC31500, in part by the Tianjin Science Foundation under Grant 16JCYBJC16500, and in part by the Tianjin Higher Education Creative Team Funds Program.

ABSTRACT As the core part of the new generation of information technology, the Internet of Things has accumulated a large number of real-time data streams of various types and structures. The data stream is generated at an extremely fast speed, and its content and distribution characteristics are all in high-speed dynamic changes, which must be processed in real time. Therefore, the feature learning algorithm is required to support incremental updates and learn the characteristics of high-speed dynamic change data in real time. Most of the current machine learning models for processing big data belong to the static learning model. The batch learning method makes it impossible to analyze data streams in real time, and the learning ability of dynamic data streams is poor. Therefore, this paper proposes an incremental high-order deep learning model to extend the data from the vector space to the tensor space and update the parameters and structure of the network model in the high-order tensor space. In the process of parameter updating, the first-order approximation concept is introduced to avoid incrementing parameters by the iterative method and to improve the parameter update efficiency, so that the updated model can quickly learn the characteristics of dynamically changing big data and satisfy the real-time requirements of big data feature learning while maintaining the original knowledge of the neural network model as much as possible. To evaluate the performance of the proposed model, experiments were performed on real image data sets-MNIST, and the model was evaluated for stability, plasticity, and run time. The experimental results show that the model not only has the ability to incrementally learn the characteristics of new data online but also retains the ability to learn the original data features, improve the model update efficiency, and maximize the online analysis and real-time processing of dynamic data streams.

INDEX TERMS Incremental high-order deep learning model, online analysis, parameter update, real-time processing, structure update, tensor space.

I. INTRODUCTION

The core content of Internet of Things applications is data, and whether it can be effectively managed is one of the main problems that restricts the development of the Internet of Things. The data information of the Internet of Things mainly includes three categories: object information, perception equipment information and real-time information. The first two categories are basic information that can be resolved by using traditional data sharing processing methods. Real-time information not only has the characteristics of large amount of data and high real-time performance of traditional big data, but also has the unique characteristics of Internet of Things information, including real-time and the massive, high-speed characteristics of big data, resulting in a higher requirement for real-time processing systems for big data in the Internet of Things environment [1], [2]. Moreover, real-time big data analysis technology is widely used in the fields of commercial Internet, intelligent buildings, smart homes, medical services, etc. For example, with medical devices, based on the current behavior of the patient, real-time analysis of the patient's physical condition, and real-time understanding of the occurrence of the disease, early prevention,

and early diagnosis can be achieved [3]. Therefore, it is a very arduous task to study how to process high-speed data streams and simultaneously use it for pattern recognition and prevention detection. It is also a subject worthy of study.

Although great progress has been made in the research and application of big data, there is a lack of real-time processing of big data generated at high speed. Currently, a large number of studies have successfully solved the problem of real-time processing of big data for high-speed change characteristics in the Internet of Things environment. The traditional processing methods as follows. Xueyin *et al.* [4] speculated that state monitoring data are a kind of streaming data, which have the characteristics of a large amount of continuous data, and unpredictable scale and order, and proposed a real-time data stream processing framework for complex equipment state monitoring. Sun *et al.* [5] summarized the mainstream technology and development trend of current data stream processing. The data stream computing system should have the characteristics of scalability, system fault tolerance, state inconsistency, load balancing, and high data throughput. In [6], Hadoop technology was used to analyze wide-area measurement data, and data mining algorithms were used to efficiently mine the interaction between sites during cascading failures. However, the traditional big data batch processing method represented by Hadoop requires frequent disk I-O operations, and the calculation efficiency is low, which makes it difficult to meet the requirements of online state detection and the evaluation of data. Guoliang *et al.* [7] studied the classification of state monitoring data by a k-means clustering algorithm and based on the characteristics of state monitoring data, used Storm real-time processing for monitoring data flow, data stream processing topology and message trees to improve the performance of state evaluation and real-time analysis of data streams to reduce the manual analysis requirement. Yuzhou *et al.* [8] proposed a system design and test results for real-time data processing based on the Storm cloud computing method for the big data problem created by passenger traffic in the rail transit industry, and proved that the scalability and fault tolerance of this method are better than other methods. Although these traditional methods can process data streams in real time, the processing effect is not ideal, the memory requirement is large, and the data processing time is relatively long. In recent years, artificial intelligence, machine learning and other technologies have begun to be applied in various fields, and have achieved good results, which has attracted widespread attention from researchers in related fields [9]. However, traditional machine learning uses a batch learning method, before data learning, all data must be prepared. However, in actual applications, all data cannot be obtained and the data change over time, which requires online processing technology [10]. Currently, in dealing with real-time data streams, most researchers mainly optimize the depth learning model by incremental learning so that they can process dynamic data. However, these methods only learn and train new data, forgetting the previous learning results, and need to relearn all the data in the update. When you need to relearn all the data, it consumes considerable time and space resources. With the increase in data, the demand for time and space will gradually increase, and the speed of learning will not keep up with the speed of data updates. For example, Lim and Harrison [11] proposed a neural network model based on self-adaptive learning based on adaptive resonance theory (ART) to construct online pattern classification. Considering both offline and online conditions and conducting experiments on two benchmark data sets, the results showed that online mode classification in nonstationary environments could achieve better results. In [12], an adaptive back-propagation algorithm based on inverse temperature parameterization was studied and compared with the online learning gradient descent (standard backpropagation algorithm) in a two-layer neural network with any number of hidden layer elements. Wan and Banta [13] proposed a new online training algorithm for the neural network-parameter incremental learning algorithm (PIL), which uses first-order approximation to optimize the adjustment parameters to adapt to the newly proposed input-output training model and retain the ability to learn the original data was better than traditional algorithms. Similarly, Yi *et al.* [14] designed an improved incremental support vector machine algorithm. This incremental learning method did not need to load the entire data set into memory, saving time and space resources, and the network model could quickly learn new data features, but this method did not consider the relatively large data stream with dynamic changes and the stability of the original data was poor. Campolucci *et al.* [15] studied the online learning process of local recurrent neural networks (LRNNs), and introduced the multilayer perceptron of infinite impulse response synapses and their changes, and used an online recursive back propagation algorithm to analyze and process online data streams in real time. Polikar *et al.* [16] proposed the Learn++ algorithm, an algorithm for the incremental training of neural network (NN) pattern classifiers. The algorithm did not need to access the previously used data during training of the new data, and achieved a better classification effect. In [17], for nonstationary real scenes, an incremental online learning algorithm based on a two-layer forward neural network was proposed for the gradual or mutated data set and a forgetting function was introduced to mark the importance of new data and achieve the ability to process real-time data streams. In short, there are many on-line processing methods for dynamic big data, but most of them only incrementally update certain features of the dynamic data stream and cannot completely learn all the features of the dynamic data stream [18]–[25].

Incremental learning was first introduced by Coppock and Freund in Science in 1962 [26]. Incremental learning refers to the continuous learning of new knowledge from a new sample from a learning system based on the existing model. It is not necessary to retrain the historical data, and update the parameters and structure of the model according to the

current new data instance. It can learn the characteristics of the change data, and at the same time, it is necessary to try to maintain the original parameters of the model so that the updated model can still learn the historical data features. Aiming at the problems existing in other research methods, this paper proposes an incremental high-order deep learning model (IHODLM), which is composed of multiple incremental high-order automatic coding models. The core part of this paper is to design a high-order automatic coding model that supports incremental update so that for a given batch of new training data, the model can update the parameters based on the existing model parameters. It does not need to relearn historical data and can quickly process new data features in real time. At the same time, the original data features can be completely and stably preserved. Therefore, the high-order automatic coding model supporting incremental updating needs to satisfy the following three properties:

(1) Incremental: Incremental means that the model only needs to learn new data and does not need to repeat the processing of historical data, and thus, save time and space resources. Based on the existing model, the structure and parameters of the existing model can be directly updated according to the characteristics of the newly added data to achieve the ability to process data in real time.

(2) Adaption: Adaption means that the updated model can completely preserve the characteristics of the historical data, that is, the updated model can still effectively learn the historical data.

(3) Preservation: Preservation means that the updated model can learn new knowledge and can learn and process new data features effectively, that is, the classification of new data and prediction errors are continuously reduced.

There are many challenges in implementing a high-order autoencoding model that supports incremental updates. This section focuses on the following key issues. Fast parameter updating: In a stable environment, the new data distribution characteristics change little. For this type of data, the high-order automatic coding model can be updated by only parameter update. The traditional incremental machine learning method has high time complexity in updating parameters. Although it is not necessary to retrain the historical data, it fails to make full use of the knowledge of the existing model. Therefore, in determining the parameter update speed is slow and cannot meet the big data real-time learning requirements. Therefore, how to fully combine the knowledge of existing models, according to the characteristics of new data, designing a fast parameter update method is a key issue to implement a high-order automatic coding model that supports incremental updates.

The rest of this paper is organized as follows. Section 2 introduces the representation of tensors and the basic operations used in this article; Section 3 introduces the details of the incremental high-order deep learning model. Performance evaluation and analysis are given in Section 4, and the entire paper concludes in Section 5.

II. TENSOR THEORY

Tensor theory is a branch of mathematics that was first used by Waldmar Vogt in 1899 [27]. Einstein used tensors to study general relativity, leading to tensors receiving extensive attention and recognition. In mathematics, a tensor is a geometric entity, or ''quantity'' in a broad sense. A tensor is a high-dimensional extension of the vector in the spatial dimension. In the traditional machine learning domain, most classical machine learning algorithms are designed based on vector space data. However, using a vector form to represent data, especially heterogeneous data, may result in the loss of a large amount of structural information of the data, making the learning result less than ideal. In the actual process, many real data sets need to be expressed in tensor form to be better represented. Therefore, machine learning methods based on tensor data have received considerable attention from many researchers in recent years. Using tensor type data not only retains its unique spatial structure information, but also the tensor learning method can effectively control the number of variables in the optimization problem to overcome the overfitting phenomenon that often occurs during vector learning;

A. TENSOR REPRESENTATION

Tensor definition: If a certain physical quantity X is described by 3^n ordered components $X_{a\ldots b}$ in the three-dimensional Cartesian coordinate system, and after the transformation α_{ij} from the coordinate system $\sum$ to $\sum'$, the following relationship is satisfied:

$$X'_{i\ldots j} = \alpha_{ia} \cdots \alpha_{jb} X_{a\ldots b} \tag{1}$$

Then the amount X is called the n-order tensor.

FIGURE 1. Gray image sample (2^{nd} order tensor).

With the advent of the information age and the increase of big data, it is especially important to mine valuable data from big data. However, most of the current data is heterogeneous and finding a unified data representation is very important for machine learning. For example, a grayscale image should be represented as a second-order tensor, as shown in Fig. 1,

FIGURE 2. Color image sample (3rd order tensor).

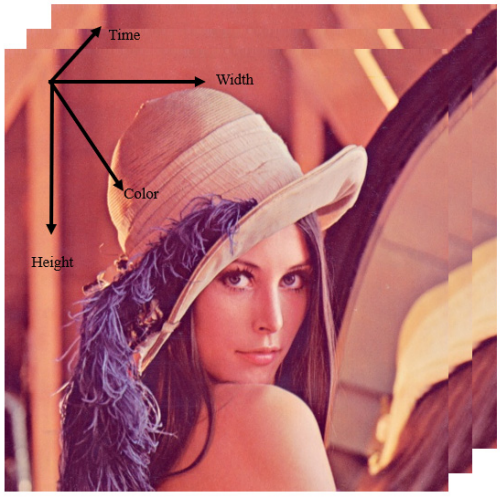


FIGURE 3. Color video sample (4th order tensor).

and a color image is represented by a third-order tensor, as shown in Fig. 2. However, in many applications, color images are usually converted to grayscale images by (2). A video is represented by a fourth-order tensor, as shown in Fig. 3.

$$Gray = 0.299R + 0.587G + 0.114B \tag{2}$$

When a color image is represented as a third-order tensor $R^{I_w \times I_h \times I_c}$, I_w, I_h, I_c always represent the width, height and color channel of the image, respectively. For example, one 512×480 resolution RGB color images can be represented as $R^{512\times480\times3}$. It can be seen in (2) that a color image can be converted into a grayscale image, and after conversion, it is expressed as $R^{512\times480}$. Similarly, a video signal can be represented as a fourth order tensor $R^{I_w \times I_h \times I_c \times I_d}$; then, I_d represents the dimension of the video, and the meaning of the other steps is the same as described above. For example, a video composed of the above pictures for 30 frames per second for a total of 20 seconds can be represented as $R^{512\times480\times3\times600}$.

B. BASIC OPERATIONS OF TENSOR

1) TENSOR *n* MODE EXPANSION

The tensor $X \in R^{I_1 \times I_2 \times \cdots \times I_N}$ is a N-order tensor. The tensor X is expanded into an $I_n \times (I_1 I_2 \ldots I_{n-1} I_{n+1} \ldots I_N)$ matrix as an array of columns in the sense of mode n, denoted as $X_{(n)}$. This matrix expansion uses the correspondence as shown in i(3) to map the element $x_{i_1 i_2 \ldots i_N}$ of the tensor X to the element $x_{(n)_{ij}}$ of the matrix $X_{(n)}$ [28].

$$\begin{aligned} j &= 1 + \sum_{l=1, l\neq n}^{N} (i_l - 1) J_l \\ J_l &= \prod_{m=1, m\neq n}^{l-1} I_m \end{aligned} \tag{3}$$

2) TENSOR *n* PATTERN PRODUCT

For the n-mode product of the N-order tensor $X \in R^{I_1 \times I_2 \times \cdots \times I_N}$ and the matrix $A \in R^{I_n \times J_n}$, it is denoted as $X \times_n A$, and finally, [29] an N-order tensor $(X \times_n A) \in R^{I_1 \times \cdots \times I_{n-1} \times J_n \times I_{n+1} \times \cdots \times I_N}$ with the following form will be obtained:

$$(X \times_n A)_{i_1 i_2 \ldots i_{n-1} j_n i_{n+1} \ldots i_N} = \sum_{i_n=1}^{I_N} \left(X_{i_1 i_2 \cdots i_{n-1} i_n i_{n+1} \cdots i_N} A_{j_n i_n} \right) \tag{4}$$

Similar to N-order tensor $X \in R^{I_1 \times I_2 \times \cdots \times I_N}$ with vector $B \in R^{I_n}$ of n, the pattern product is noted as $(X \times_n B)$ and finally, obtain an $N-1$ order tensor $(X \times_n B) \in R^{I_1 \times \cdots \times I_{n-1} \times I_{n+1} \cdots \times I_N}$.

$$(X \times_n B)_{i_1 i_2 \cdots i_{n-1} i_{n+1} \cdots i_N} = \sum_{i_n=1}^{I_N} X_{i_1 i_2 \cdots i_N} B_{i_n} \tag{5}$$

C. TENSOR PRODUCT

Given one N-order tensor $X \in R^{I_1 \times I_2 \times \cdots \times I_N}$ and one M-order tensor $C \in R^{J_1 \times J_2 \times \cdots \times J_M}$. Their outer product will produce an $(N+M)$-order tensor $D \in R^{I_1 \times I_2 \times \cdots I_N \times J_1 \times J_2 \times \cdots J_M}$ [30], tensor D expressed as:

$$\begin{aligned} D &= (X \otimes C)_{I_1 \times I_2 \times \cdots \times I_N \times J_1 \times J_2 \times \cdots \times J_M} \\ &= X_{I_1 \times I_2 \times \cdots \times I_N} C_{J_1 \times J_2 \times \cdots \times J_M} \end{aligned} \tag{6}$$

The tensor decomposition includes Canonical Polyadic decomposition and tucker decomposition, and the relationship between them and the decomposition process is shown in Fig. 4.

III. INCREMENTAL HIGH-ORDER DEEP LEARNING MODEL

A. INCREMENTAL HIGH-ORDER DEEP LEARNING MODEL BASED ON PARAMETER UPDATE

The incremental parameter update means that the parameters of the model are updated from θ to $\theta + \Delta\theta$ according to the characteristics of the newly added data without changing the network connection structure, so that the updated parameters can learn the characteristics of the newly added data. The incremental parameter update is mainly for data structures

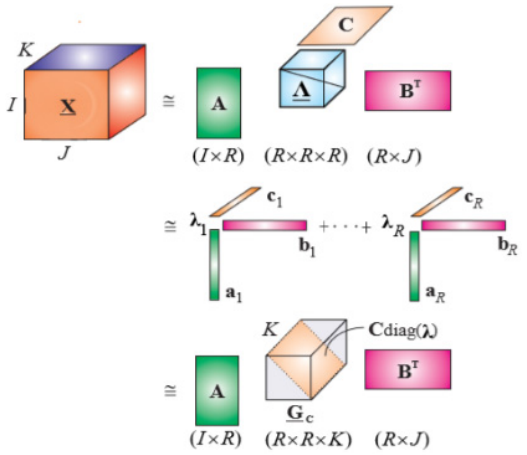


FIGURE 4. Tensor decomposition.

with little change in dynamic data or similar characteristics between the data, and the incremental update of parameters is relatively simple. In this paper, using the first-order approximation of the Taylor expansion, the incremental parameter $\Delta\theta$ of the model can be quickly obtained to maximize the update speed of the model parameters, and meet the requirements of real-time processing of big data and real-time learning of dynamic features.

The tensor of the high-order deep learning model (HODLM) parameters is expressed as $\theta = \{W^{(l)}, b^{(l)}; W^{(l)}, b^{(l)}\}$. For example, when the input data are the N-order tensor, $b^{(l)}(l = 1, 2)$ and $W^{(l)}(l = 1, 2)$ are represented by the N-order tensor and the $(N + 1)$-order tensor, respectively. To use the first-order approximation of the function to quickly find the incremental parameter $\Delta\theta$ of the model, this paper maps the parameters of the HODLM from tensor space to the vector space in the form of the parameter vector in the form of tensor n-mode expansion and the parameter quantity is calculated in the form of a parameter vector. According to the nature of incremental learning, the HODLM based on the parameter update proposed in this paper requires the updated model to learn the characteristics of the new data, that is, the plasticity of the model. At the same time, it is necessary to try to maintain the original knowledge of the model, so that the updated model can still effectively learn the characteristics of the historical data, that is, the stability of the model. Therefore, for a given new training data A, to obtain the plasticity of the model, the definition based on the weight adaption error function $J_{adaption}$ is:

$$J_{adaption} = \frac{1}{2}\Delta a_{new}^{T} W \Delta a_{new} = \frac{1}{2} W \Delta a_{new}^{2} \tag{7}$$

Where W is a weight matrix, a typical weight matrix takes the value $W = \left(I - u\left(\frac{\partial f(a,\theta)}{\partial\theta}\right)^{T}\left(\frac{\partial f(a,\theta)}{\partial\theta}\right)\right)^{T}$; $\Delta a_{new} = f(a, \theta + \Delta\theta) - a$ represents the reconstruction error after the model parameter is updated from θ to $\theta + \Delta\theta$. θ represents the original parameters of the model, $\Delta\theta$ represents the increment of the parameter θ and u represents the learning efficiency. f is the activation function, and generally, f uses sigmoid or tanh.

To ensure that the updated parameters satisfy the stability of the model, the definition based on the parameter update stability error function $J_{adaption}$ is:

$$J_{preservation} = \frac{1}{2u}\Delta\theta^{T}\Delta\theta = \frac{1}{2u}\Delta\theta^{2} \tag{8}$$

To balance the plasticity and stability of the parameter update, the reconstruction error function $J(a, \theta + \Delta\theta)$ is defined as:

$$J(a, \theta + \Delta\theta) = J_{adaption} + J_{preservation} \tag{9}$$

The incremental $\Delta\theta$ of the model parameters is solved by minimizing the error function $J(a, \theta + \Delta\theta)$. Because the minimum error function $J(a, \theta + \Delta\theta)$ is a typical nonlinear optimization problem when calculating the parameter increment $\Delta\theta$, we first use Taylor's theorem to expand $J(a, \theta + \Delta\theta)$:

$$\begin{aligned} f(a, \theta + \Delta\theta) = f(a, \theta) &+ \left(\frac{\partial f(a, \theta)}{\partial\theta}\right)\Delta\theta \\ &+ \left(\frac{\partial'' f(a, \theta)}{\partial\theta^{2}}\right)\Delta\theta^{2} + \cdots \\ &+ \left(\frac{\partial^{(n)} f(a, \theta)}{\partial\theta^{n}}\right)\Delta\theta^{n} + R_n(\theta + \Delta\theta) \end{aligned} \tag{10}$$

Because $\Delta\theta$ is small, the first order approximation of $J(a, \theta + \Delta\theta)$ can be expressed as:

$$f(a, \theta + \Delta\theta) \cong f(a, \theta) + \left(\frac{\partial f(a, \theta)}{\partial\theta}\right)\Delta\theta \tag{11}$$

Further available:

$$\begin{aligned} \Delta a_{new} &= f(a, \theta + \Delta\theta) - a \\ &= f(a, \theta + \Delta\theta) - f(a, \theta) + f(a, \theta) - a \\ &\cong \left(\frac{\partial f(a, \theta)}{\partial\theta}\right)\Delta\theta + \Delta a \end{aligned} \tag{12}$$

Therefore, the reconstruction error function $J(a, \theta + \Delta\theta)$ can be approximated as:

$$J(a, \theta + \Delta\theta) \cong \frac{W}{2}\left(\left(\frac{\partial f(a, \theta)}{\partial\theta}\right)\Delta\theta + \Delta a\right)^{2} + \frac{1}{2u}\Delta\theta^{2} \tag{13}$$

The reconstruction error function formula (13) is derived for $\Delta\theta$, and the derivative is equal to 0. The approximate calculation formula for obtaining $\Delta\theta$ is as follows:

$$\frac{\partial J(a, \theta+\Delta\theta)}{\partial\Delta\theta} \cong W\left(\left(\frac{\partial f(a, \theta)}{\partial\theta}\right)\Delta\theta + \Delta a\right) + \frac{1}{u}\Delta\theta \triangleq 0 \tag{14}$$

Therefore, the approximate solution is:

$$\Delta\theta = -u\left(\frac{\partial f(a, \theta)}{\partial\theta}\right)^{-1}\Delta a \tag{15}$$

For parameter updates, given a batch of new data, based on the guaranteed stability and plasticity, the original parameter θ is updated by identifying a parameter increment $\Delta\theta$ that minimizes the error function $J(a, \theta + \Delta\theta)$. Therefore, the parameter update algorithm based on the first-order approximation is summarized as follows:

(1) Calculate the output value $f(a, \theta)$ of the HODLM by forward propagation;

(2) Calculate the difference Δa between the output value of the HODLM and the ideal output value a;

(3) Calculate the partial derivative $\partial f(a, \theta)/\partial\theta$ of the original parameter θ by the high-order back propagation algorithm to calculate the output value $f(a, \theta)$ of the HODLM;

(4) Calculate the inverse matrix $\left(\partial f(a, \theta)/\partial\theta\right)^{-1}$ of the partial derivative, and finally calculate the parameter increment $\Delta\theta$ of the HODLM according to (15), and update the parameter of the model to $\theta + \Delta\theta$.

It can be seen from the above summary that the update algorithm based on the first-order approximation parameter does not need to iteratively update the parameters, only the current new data instance needs to be loaded into the memory, and the inverse matrix of the partial derivative of $f(a, \theta)$ to θ is mainly calculated, so the time complexity of the algorithm approximates $O(m)$, where m represents the number of parameters of the network model, so the parameter update speed can meet the real-time requirements of big data feature learning to the greatest extent.

B. HIGH-ORDER DEEP LEARNING MODEL SUPPORTING INCREMENTAL UPDATES

In this paper, multiple incremental high-order automatic encoders are stacked to construct an incremental high-order deep learning model. The incremental deep learning model consists of two phases: pretraining and finetuning. In the pre-training phase, each basic module of the model is trained layer by layer and incremental methods are used by means of parameter updating or structural updating to obtain the initialization parameters. In the fine-tuning phase, we use the high-order backpropagation algorithm and the gradient descent algorithm to fine-tune the initialization parameters from top to bottom, so that the parameters are optimal.

IV. EXPERIMENT

To evaluate the effectiveness of the proposed algorithm, we perform experiments on two typical image data sets-MNIST and CUAVE. In the experiment, we verify the reliability and practicability of the algorithm from the aspects of stability, plasticity and time spent on parameter training. In this experiment, it is running MATLABR2017 and it is based on the deep learning framework of PYTORCH.

A. BASED ON PARAMETER UPDATE VERIFICATION–MNIST HANDWRITTEN DATA SET

MNIST is a handwritten database created by Corinna Cortes of Google Labs and Yann LeCun of the New York University's Kelang Institute. The database consists of

FIGURE 5. Part of the sample of MNIST.

60,000 training images and 10,000 test images, which are divided into 10 categories. Each picture can be represented as a second-order tensor $R^{28\times28}$, where 28×28 represents the resolution of the picture. Fig. 5 shows a partial sample of the MNIST dataset.

Because the training characteristics of the training set and the test set of the MNIST data set are not obvious, the parameter update number is small, and only the parameter update is required to cause the updated network model to learn the characteristics of the new data. Therefore, this paper uses the MNIST data set to verify the validity of the first-order approximation parameter incremental update algorithm. It is evaluated in terms of plasticity, stability and running time.

To verify the validity of the incremental deep learning model, the following three subsets are designed based on the MNIST data set:

X_0: Contains 60,000 training pictures as a pretraining data set.

X_1: Randomly select 5000 images from the test data set as an incremental training data set.

X_2: The remaining 5000 images are selected from the test data set as test data sets.

The following four parameters are trained using the above three data subsets:

θ_0: Perform a basic deep learning model on data set X_0 to obtain the model parameters θ_0.

θ_1: Using θ_0 as the original parameter, perform a first-order approximation of the deep learning model supporting incremental update in data set X_1, and obtain the update parameter θ_1 of the incremental model.

θ_2: Perform a basic deep learning model on data set X_0+X_1 to obtain parameter θ_2.

θ_3: Perform a basic deep learning model on data set X_2 to obtain parameter θ_3.

First, we use X_2 as the test data set to verify the plasticity of the incremental deep learning model for new data. We use the two functions of loss function and accuracy to evaluate the degree of adaptation of the updated model learning big data in real-time with dynamic changes. The smaller the loss function (Loss), the better the plasticity of the updated model

for new data; the higher the accuracy (ACC), the stronger the plasticity of the updated model for new data. All experiments are repeated five times and all experiments use the same model structure. The experimental results are shown in the figures, and we also present the average of the five experiments, as shown in Table I.

TABLE 1. Average results of five experiments (adaption).

Parameter	Accuracy (%)	Loss
θ_0	98.072	0.218
θ_1	98.846	0.128
θ_2	98.778	0.138
θ_3	99.720	0.03

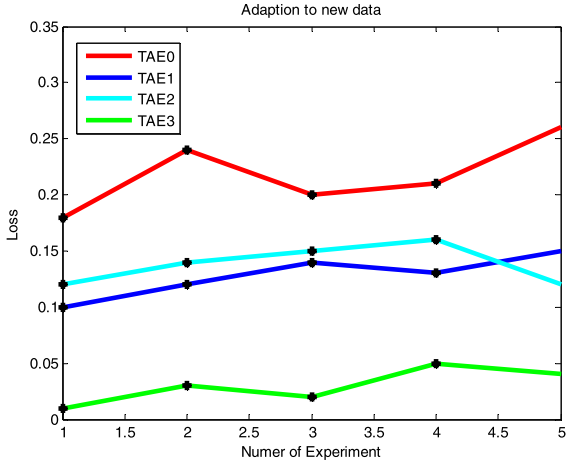


FIGURE 6. Loss function experimental results (adaption).

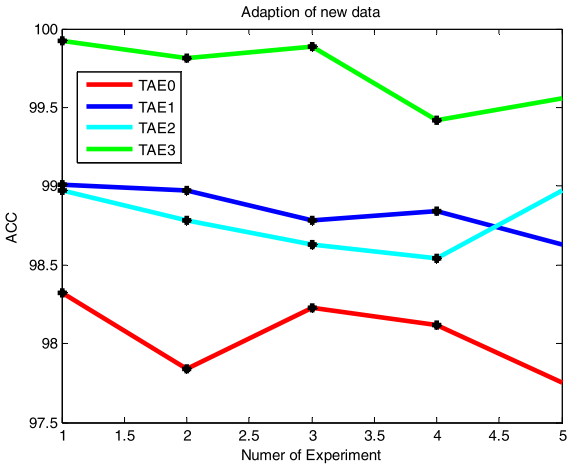


FIGURE 7. Accuracy test results (adaption).

It can be seen in Fig.6, Fig.7 and Table I that the training of X_2 with θ_3 as the parameter yields the highest accuracy and the lowest loss function. This is because you can obtain the best result by training yourself taking θ_0 as the parameter. Training X_2, the accuracy is the smallest, and the loss function is the largest, because the basic deep learning model is a static learning model. Once the parameters are determined, it can't be updated, and it is difficult to learn the characteristics of the new data. The incremental high-order deep learning model proposed in this paper is different from the basic deep learning model. It can incrementally update the parameters, so that the model can learn the characteristics of the newly added data. Therefore, the accuracy is obtained by using θ_1 as a parameter. It is obviously higher than θ_0, and the loss function is significantly lower than θ_0, which indicates that the incremental deep learning model can adapt to the new data characteristics and effectively learn new data after updating the parameters. At the same time, it can be seen that the accuracy obtained by classifying θ_2 is slightly lower than θ_1, and the loss function is slightly higher than θ_1, but it also achieves good results, because θ_2 is the whole in the new data instance and the original data. A basic deep learning model is implemented such that the parameters include information about the new data features.

Next, we use X_0 as the test data set to verify the stability of the four parameters to the original data. It is also verified by the two indicators of accuracy and loss function. The smaller the loss function (Loss), the better the stability of the updated model to the original data; the higher the accuracy rate (ACC), the stronger the stability of the updated model to the original data. All experiments are repeated five times and all experiments use the same model structure. The experimental results are shown in the figures, and we also present the average of the five experiments, as shown in Table II.

TABLE 2. Average results of five experiments (preservation).

Parameter	Accuracy (%)	Loss
θ_0	99.376	0.07
θ_1	98.540	0.162
θ_2	98.154	0.208
θ_3	92.690	0.918

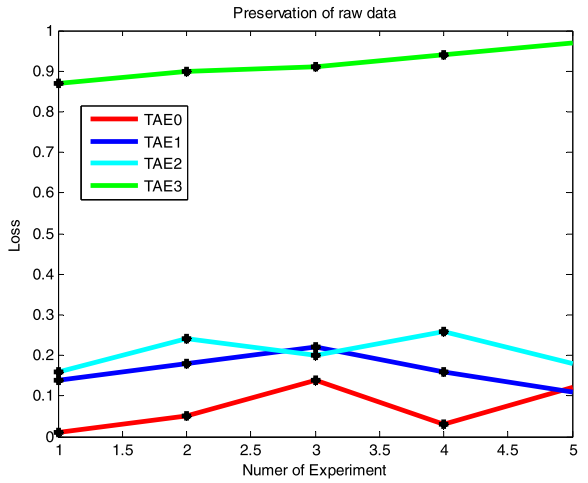


FIGURE 8. Loss function experimental results (preservation).

As shown in Fig. 8, Fig. 9 and Table II, training X_0 with θ_0 as the parameter yields the highest accuracy and the lowest loss function. This is because you can obtain the best results by training yourself taking θ_3 as the parameter. Training X_0,

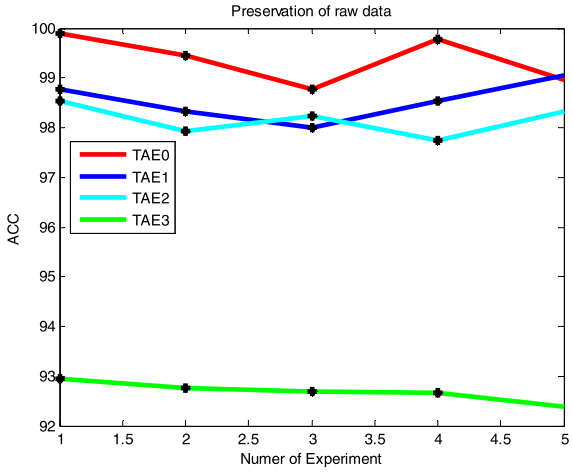


FIGURE 9. Accuracy experiment results (preservation).

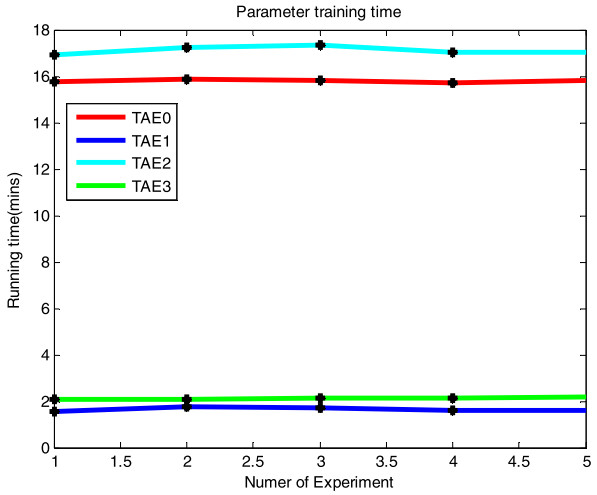


FIGURE 10. Training time of the parameters.

the accuracy is the smallest, and the loss function is the largest. This is because θ_3 is the parameter trained for the new data and does not contain the characteristics of the original data. Therefore, the original data are trained to obtain the worst classification result. The experimental results show that the accuracy G and the loss function of the parameter θ_1 obtained by the incremental deep learning model training and the parameter θ_2 containing both the original data and the newly added data training are slightly worse than θ_0, indicating that the algorithm is correct. The feature learning of historical data has good stability, and it can still effectively learn historical data features after updating the model parameters.

The above experiments show that the proposed algorithm can learn the feature information of the newly added data after updating the parameters and can effectively maintain the learning ability of the historical data characteristic information.

We also compare the time spent on the training of the four parameters and the superiority of the algorithm proposed in this paper. As shown in Fig.10 and Table III, the incremental deep learning model obtained by the proposed algorithm

TABLE 3. Five experimental parameters average training time.

Parameter	Average running time(mins)
θ_0	15.796
θ_1	1.644
θ_2	17.12
θ_3	2.116

takes the least time to train parameter θ_1. This is because the incremental deep learning model passes the first-order approximation above the original parameters of the model. The concept is to directly identify the incremental parameters of the model and update the model parameters without multiple iterations. The other three parameters are obtained by performing the basic deep learning model, so the execution time is longer than θ_1, and θ_2 contains the largest number of samples and the longest time required, θ_3 contains the smallest number of samples, and the time required is relatively small. In summary, the proposed method is the most effective for network update models that only require parameter updates.

V. CONCLUSION

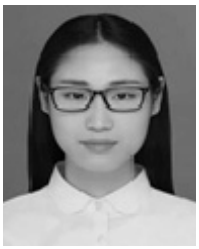
In this paper, an incremental high-order deep learning model based on parameter updating and structure updating is proposed to meet the requirements of dynamic big data online analysis and real-time processing. The experimental results show that the parameter update algorithm based on the first-order approximation idea of the handwritten data set MNIST does not need to retrain the original data, and does not need to iteratively update the parameters. Compared with other training parameter algorithms, only 1.644 mins is needed to improve the parameter update efficiency. The classification accuracy of the new data and the original data can achieve better accuracy, which fully demonstrates that the proposed model satisfies the three characteristics of incremental learning, incrementality, stability and plasticity. Based on the dynamic data stream CUAVE with relatively frequent dynamic changes, the structure update algorithm of the hidden layer neurons is used, and the original data training parameters are used as the original parameters. When learning new data features, the new parameters are trained by using the original parameters, so that the model can converge quickly and the number of convergence times is stable, which satisfies the real-time requirements of big data feature learning to the greatest extent. The next step will optimize the model proposed in this paper, so that after increasing the hidden layer neuron structure, as the number of connections increases, the network connection structure is optimized to avoid overfitting of the model. The model proposed in this paper will be further extended, so that the proposed model can effectively learn the characteristics of massive data, heterogeneity and incompleteness of big data.

REFERENCES

[1] M. Xiaoping, H. Yanjun, and M. Yanzi, "Application research of technologies of Internet of Things, big data and cloud computing in coal mine safety production," *Ind. Mine Automat.*, vol. 40, no. 4, pp. 5–9, 2014.

[2] H. Yuzhou and H. Chuanfeng, "Construction of intelligent building health information service management system based on Internet of Things and big data," *Building Econ.*, no. 5, pp. 101–106, 2015.
[3] L. Guojie and C. Xueqi, "Big data research: The future of science and technology and socio-economic development of major strategic areas-big data research and scientific thinking," *Bull. Chin. Acad. Sci.*, vol. 27, no. 6, pp. 647–657, 2012.
[4] Z. Xueyin, Z. Li, and W. Xiaoqi, "Real-time stream data processing framework for complex equipment state monitoring," *Comput. Integr. Manuf. Syst.*, vol. 19, no. 12, pp. 2929–2939, 2013.
[5] D.-W. Sun, G.-Y. Zhang, and W.-M. Zheng, "Big data stream computing: Technologies and instances," *J. Softw.*, vol. 25, no. 4, pp. 839–862, 2014.
[6] Z. Qu, Z. Li, and S. Zhang, "Data processing of Hadoop-based wide area measurement system," *Autom. Electr. Power Syst.*, vol. 37, no. 4, pp. 92–97, 2013.
[7] Z. Guoliang, Z. Yongli, and W. Guilan, "Application of real-time big data processing technology in condition monitoring," *Trans. China Electrotech. Soc.*, vol. 29, no. s1, pp. 432–437, 2014.
[8] H. Yuzhou, F. Bin, and G. Xuedao, "Application of real-time data processing services in automatic clearing collection system based on cloud computing for storm," *Comput. Appl.*, vol. 34, no. s1, pp. 96–99, 2014.
[9] H. Rong, "Research on incremental machine learning algorithm," Ph.D. dissertation, Dept. School Autom., Nanjing Univ. Sci. Technol., Nanjing, China, Tech. Rep., 2013.
[10] S. Shalev-Shwartz, "Online learning and online convex optimization," *Found. Trends Mach. Learn.*, vol. 4, no. 2, pp. 107–194, 2012.
[11] C. P. Lim and R. F. Harrison, "Online pattern classification with multiple neural network systems: An experimental study," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 33, no. 2, pp. 235–247, May 2003.
[12] A. H. L. West and D. Saad, "On-line learning with adaptive back-propagation in two-layer networks," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 56, no. 3, pp. 3426–3445, 1997.
[13] S. Wan and L. E. Banta, "Parameter incremental learning algorithm for neural networks," *IEEE Trans. Neural Netw.*, vol. 17, no. 6, pp. 1424–1438, Nov. 2006.
[14] Y. Yi, J. Wu, and W. Xu, "Incremental SVM based on reserved set for network intrusion detection," *Expert Syst. Appl.*, vol. 38, no. 6, pp. 7698–7707, 2011.
[15] P. Campolucci, A. Uncini, F. Piazza, and B. D. Rao, "On-line learning algorithms for locally recurrent neural networks," *IEEE Trans. Neural Netw.*, vol. 10, no. 2, pp. 253–271, Mar. 1999.
[16] R. Polikar, L. Upda, S. S. Upda, and V. Honavar, "Learn++: An incremental learning algorithm for supervised neural networks," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 31, no. 4, pp. 497–508, Nov. 2001.
[17] B. Pérez-Sánchez, O. Fontenla-Romero, and B. Guijarro-Berdiñas, "An incremental learning method for neural networks in adaptive environments," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, Jul. 2010, pp. 1–8.
[18] B. Pérez-Sánchez, O. Fontenla-Romero, and B. Guijarro-Berdiñas, "An online learning algorithm for adaptable topologies of neural networks," *Expert Syst. Appl.*, vol. 40, no. 18, pp. 7294–7304, 2013.
[19] M. Rattray and D. Saad, "Globally optimal on-line learning rules for multi-layer neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, no. 22, 1997, pp. 771–776.
[20] N.-Y. Liang, G.-B. Huang, P. Saratchandran, and N. Sundararajan, "A fast and accurate online sequential learning algorithm for feedforward networks," *IEEE Trans. Neural Netw.*, vol. 17, no. 6, pp. 1411–1423, Nov. 2006.
[21] W.-F. Hsiao and T.-M. Chang, "An incremental cluster-based approach to spam filtering," *Expert Syst. Appl.*, vol. 34, no. 3, pp. 1599–1608, 2008.
[22] R. Elwell and R. Polikar, "Incremental learning of concept drift in nonstationary environments," *IEEE Trans. Neural Netw.*, vol. 22, no. 10, pp. 1517–1531, Oct. 2011.
[23] J.-T. Chien and H.-L. Hsieh, "Nonstationary source separation using sequential and variational Bayesian learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 5, pp. 681–694, May 2013.
[24] D. Chakraborty and N. R. Pal, "A novel training scheme for multilayered perceptrons to realize proper generalization and incremental learning," *IEEE Trans. Neural Netw.*, vol. 14, no. 1, pp. 1–14, Jan. 2003.
[25] J. Platt, "A resource-allocating network for function interpolation," *Neural Comput.*, vol. 3, no. 2, pp. 213–225, 1991.
[26] H. W. Coppock and J. E. Freund, "All-or-none versus incremental learning of errorless shock escapes by the rat," *Science*, vol. 135, no. 3500, pp. 318–319, 1962.
[27] Y. Bing, "Research and application of machine learning method based on tensor data," Ph.D. dissertation, Dept. College Sci., China Agricult. Univ., Beijing, China, Tech. Rep., 2014.
[28] T. Guo, L. Han, L. He, and X. Yang, "A GA-based feature selection and parameter optimization for linear support higher-order tensor machine," *Neurocomputing*, vol. 144, no. 1, pp. 408–416, 2014.
[29] Z. Hao, L. He, B. Chen, and X. Yang, "A linear support higher-order tensor machine for classification," *IEEE Trans. Image Process.*, vol. 22, no. 7, pp. 2911–2920, Jul. 2013.
[30] J. Zhang, Y. Han, and J. Jiang, "Semi-supervised tensor learning for image classification," *Multimedia Syst.*, vol. 23, no. 1, pp. 63–73, 2017.

YULIANG LI was born in Henan, China, in 1994. He received the B.S. degree from the Tianjin University of Technology and Education in 2015. He is currently pursuing the M.S. degree in communication engineering with Tianjin Normal University. His research interests include data mining, deep learning, and acceleration sensor.



MIN ZHANG was born in Shandong, China, in 1992. She is currently pursuing the M.S. degree in communication engineering with Tianjin Normal University. Her research interests include deep learning and data stream.

WEI WANG received the Ph.D. degree from Tianjin University. He is currently an Associate Professor with the College of Electronic and Communication Engineering, Tianjin Normal University, Tianjin, China. His main research interests include compressive sensing, radar signal processing, optical fiber sensing technology, and applications.