Journal of Applied Mathematics and Physics, 2024, 12, 1247-1262
https://www.scirp.org/journal/jamp
ISSN Online: 2327-4379
ISSN Print: 2327-4352

Fully Distributed Learning for Deep Random Vector Functional-Link Networks

Huada Zhu[1], Wu Ai[1,2*]

[1]School of Mathematics and Statistics, Guilin University of Technology, Guilin, China
[2]Guangxi Colleges and Universities Key Laboratory of Applied Statistics, Guilin, China
Email: *aiwu818@gmail.com

How to cite this paper: Zhu, H.D. and Ai, W. (2024) Fully Distributed Learning for Deep Random Vector Functional-Link Networks. *Journal of Applied Mathematics and Physics*, **12**, 1247-1262.
https://doi.org/10.4236/jamp.2024.124077

Received: March 18, 2024
Accepted: April 25, 2024
Published: April 28, 2024

Abstract

In the contemporary era, the proliferation of information technology has led to an unprecedented surge in data generation, with this data being dispersed across a multitude of mobile devices. Facing these situations and the training of deep learning model that needs great computing power support, the distributed algorithm that can carry out multi-party joint modeling has attracted everyone's attention. The distributed training mode relieves the huge pressure of centralized model on computer computing power and communication. However, most distributed algorithms currently work in a master-slave mode, often including a central server for coordination, which to some extent will cause communication pressure, data leakage, privacy violations and other issues. To solve these problems, a decentralized fully distributed algorithm based on deep random weight neural network is proposed. The algorithm decomposes the original objective function into several sub-problems under consistency constraints, combines the decentralized average consensus (DAC) and alternating direction method of multipliers (ADMM), and achieves the goal of joint modeling and training through local calculation and communication of each node. Finally, we compare the proposed decentralized algorithm with several centralized deep neural networks with random weights, and experimental results demonstrate the effectiveness of the proposed algorithm.

Keywords

Distributed Optimization, Deep Neural Network, Random Vector Functional-Link (RVFL) Network, Alternating Direction Method of Multipliers (ADMM)

1. Introduction

In recent years, with the rapid development of digital technology and network

technology, the scale of data we can collect is unprecedented, and it presents three characteristics: one is the large scale of data, the other is the high dimension of data, and the third is the distributed storage of data. These characteristics of data bring us a lot of challenges in processing data [1]. Traditional centralized machine learning is limited to a single machine for processing calculations, which has revealed many drawbacks. The problem of limited training data size and long training time makes centralized learning unable to meet the requirements of processing today's big data, so it is necessary to deploy the data to be processed to multiple machines for joint modeling in a distributed manner, which also corresponds to this feature of data distributed storage [2]. Therefore, it is of great significance to apply fast and efficient distributed learning algorithm to the original neural network.

On the other hand, in recent years, deep neural networks have become a very popular research direction in the field of machine learning, and have made major breakthroughs in many fields. Although deep neural networks are favored by everyone because of their excellent performance, with the rapid development of digitalization and the three characteristics of data presentation, stand-alone can no longer meet the training requirements of deep neural networks. Therefore, the application of distributed optimization algorithms to deep neural networks has become a new research trend. As early as 2012, Dean *et al.* [3], a researcher at Google, developed two distributed training algorithms, Downpour SGD and Sandblaster L-BFGS, in the training of a large-scale deep neural network. It is of great significance. Of course, there is a gradual increase in research on distributed deep neural networks, and many frameworks that support distributed training have emerged, such as the TensorFlow framework proposed by Abadi *et al.* [4] and the Horovod framework proposed by Sergeev *et al.* [5]

To realize distributed training of models, two distributed frameworks are generally adopted [6], one is master-slave mode, and the other is point-to-point mode. In master-slave mode, there is a central node, which is responsible for collecting and aggregating data or model parameters sent by other child nodes for processing and calculation, and then sending the calculated results to them respectively [7] [8]. Such a communication architecture may cause problems with communication stress on the one hand [9], and risks of data leakage and misuse on the other [10]. In a point-to-point distributed architecture, there is no central node in the network, and the state between nodes is the same. Depending on the topology of the network, a node communicates with one or more other nodes, and after several rounds of communication, the entire network eventually reaches the goal of consistency. This decentralized, fully distributed architecture not only saves some communication overhead, but also data or model parameters are communicated only between adjacent nodes, thus preserving data privacy [11]. Due to the advantages of this framework, there have been many researches and applications on this distributed framework in recent years, and the application examples in deep learning are [12] [13] and so on.

In addition, the choice of algorithm for deep neural network also has an im-

portant impact on the efficiency of the model. Gradient algorithm is that most widely used neural networks learn algorithms in deep neural networks. However, traditional gradient algorithms have some disadvantage, such as easy to fall into local minimum points, slow convergence speed, strong dependence on initial parameters, etc. [14]. For deep networks, gradient algorithms also have gradient vanishing or gradient explosion problems, which will affect the training efficiency and make it difficult to exert the strong learning ability of the deep neural network [15]. In order to solve these problems, this paper proposes a distributed learning method based on deep random weight neural network. Compared with traditional neural network, random weight neural network has a very fast training speed, reduces the probability of falling into local minimum point, and ensures good approximation and generalization ability. Representative deep random neural networks, such as multi-hidden layer feedforward neural networks (MLFN) [16], limit learners for deep structures (H-ELM) [17], deep random vector functional-link neural networks based on stacked autoencoders (sdRVFL) [18] [19], etc., where sdRVFL has faster and better generalization ability than the above deep random networks.

Based on the solid foundation of the above models and theories, combining the advantages of current deep neural networks and distributed learning frameworks in various aspects, this paper creates a point-to-point fully distributed deep vector functional-link model algorithm called D-sdRVFL on the proposed deep random vector functional-link neural network (sdRVFL). Our proposed algorithm is based on the decentralized average consensus (DAC) [20] and alternating direction method of multipliers (ADMM) [21]. In the process of distributed model training, we first use ADMM algorithm to transform the global consistency optimization problem of the model into equivalent subproblems to solve. In the process of solving, we involve the values that need global information to calculate. We use DAC algorithm to achieve global consistency only through communication between nodes, avoiding the existence of central nodes, and finally realizing decentralized and completely distributed training of deep learning models. The main contributions of this paper are as follows:

- A peer-to-peer distributed learning algorithm based on deep RVFL is proposed, in which multiple nodes can jointly train modeling without a central server, while also protecting data privacy.
- According to two different connection variants of deep RVFL network, we propose corresponding distributed deep neural network algorithms.
- The proposed D-sdRVFL algorithm is comparable to the centralized deep RVFL algorithm in performance. The experimental results on multiple classification datasets show that the proposed algorithm has little difference in model accuracy with the centralized deep RVFL, and the training speed of the model is improved. Compared with the centralized algorithm, the point-to-point distributed algorithm has great advantages in dealing with large-scale high-dimensional data, and at the same time, it also protects data privacy to a cer-

tain extent.

The rest of this paper is structured as follows. **Section 2** briefly introduces the basic concepts and training optimization process of two kinds of deep RVFL networks. In **Section 3**, decentralized fully distributed optimization algorithms are proposed for two kinds of deep RVFL networks. In **Section 4**, we compare the performance of the proposed distributed algorithm with other centralized deep random weight algorithms. **Section 5** summarizes the paper.

2. Preliminary

In this section, we will introduce the basic structure of deep RVFL and its optimization problems, and introduce the concept of the decentralized average consensus (DAC) as the theoretical basis for our extension of the network to decentralized distributed deep networks.

2.1. Deep RVFL with Direct Links

In the Deep RVFL with direct links network, the original data first goes through L hidden layers for feature extraction to obtain complex high-level features, and then enters the RVFL classifier. The learning and optimization of the whole network are also divided into two parts, one is the optimization of the reconstruction matrix of the hidden layer encoder, and the other is the optimization of the weight matrix of the RVFL classifier.

The hidden layers in depth RVFL are composed of stacked self-encoded layers of L layers, and the output of each hidden layer represents $\mathbf{H}_l$. In the hidden layer, the output result of the previous layer is used as the input value of the next layer. The optimization problem for each coding layer is as follows:

$$\hat{\mathbf{U}}_l = \arg\min_{\mathbf{U}_l} \frac{1}{2}\left\|\mathbf{Z}_l\mathbf{U}_l - \mathbf{H}_{l-1}\right\|^2 + \lambda_l \left\|\mathbf{U}_l\right\|_1 \tag{1}$$

where $\mathbf{H}_{l-1}$ is the output of the coding layer of the $l-1$th layer, and is also the input of the encoder of the coding layer of the lth layer, $\mathbf{Z}_l$ is the output of the encoder of the coding layer of the lth layer obtained by the activation function, and $H_0 = X$, our goal is to optimize the weight matrix U of the decoder of the coding layer, λ_l is the regularization parameter of the lth layer.

After L self-encoding layers, the final feature representation $\mathbf{H}_L$ is obtained. We need to connect $\mathbf{H}_L$ with the original data $\mathbf{X}$ and then enter the classifier of RVFL. We use $\mathbf{X}_c$ to represent the input value of the classifier, and $\mathbf{X}_c$ can be defined as:

$$\mathbf{X}_c = \left[\mathbf{H}_L, \mathbf{X}\right]. \tag{2}$$

In the RVFL classifier, the learning objective is to optimize the weight matrix $\boldsymbol{\beta}$, and the optimization objective function is as follows:

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \frac{1}{2}\left\|\mathbf{X}_c\boldsymbol{\beta} - \mathbf{T}\right\|^2 + \frac{\lambda}{2}\left\|\boldsymbol{\beta}\right\|^2 \tag{3}$$

where $\mathbf{T}$ is the target matrix, λ is the regularization parameter.

2.2. Deep RVFL with Dense Direct Links

In the Deep RVFL with dense direct links network, the original data is first subjected to feature extraction through hidden layers, and in the self-encoding layers of the L layers in the hidden layers, each self-encoding layer is connected with the subsequent self-encoding layer, so that the input value of each subsequent hidden layer includes the output values of all the previous hidden layers, and each hidden layer input $\mathbf{X}_l$ can be represented as follows:

$$\mathbf{X}_l = \left[\mathbf{X}, \mathbf{H}_1, \cdots, \mathbf{H}_{l-1}\right] \tag{4}$$

where $\mathbf{X}$ is the original data, $\mathbf{H}_l$ is the output value of each hidden layer, and the form of the output value may refer to formula:

$$\mathbf{H}_l = \mathbf{Z}_l \hat{\mathbf{U}}_l, \tag{5}$$

except that the input value of each layer is changed.

After passing through L hidden layers, we get the output value $\mathbf{H}_L$ of the hidden layer. We connect the output value $\mathbf{H}_l$ of each hidden layer with the original data and enter the classifier of RVFL as a whole. We use $\mathbf{X}_c$ to represent the input value of the classifier, which can be expressed as:

$$\mathbf{X}_c = \left[\mathbf{X}, \mathbf{H}_1, \cdots, \mathbf{H}_L\right]. \tag{6}$$

The optimization problem in the RVFL classifier is the same as in Equation (3), we need to solve the optimal weight matrix $\boldsymbol{\beta}$.

2.3. Decentralized Average Consensus (DAC)

DAC [15] is an algorithm that iterates continuously over the parameters of each node to reach a global average, requiring only communication between nodes. Here, we assume that there are N nodes in the network. In the kth iteration, the parameter of a node i is ψ_i, and the update of the DAC of each local node is as follows:

$$\psi_i(k) = \sum_{j=1}^{N} b_{ij} \psi_i(k-1) \tag{7}$$

where $B = \left[b_{ij}\right]$ B is an adjacency matrix of size $N \times N$, The parameters will gradually converge to the global average value through continuous iteration, as follows:

$$\lim_{k \to +\infty} \psi_i(k) = \frac{1}{N} \sum_{i=1}^{N} \psi_i(0), \quad \forall i \in \mathcal{N}. \tag{8}$$

3. Fully Distributed Deep RVFL Network

In this section, we extend the previous two forms of deep RVFL to the peer-to-peer distributed learning framework. By using DAC and ADMM methods to optimize the weights of each hidden layer of each node and the weights of their RVFL classifiers. The following describes two distributed deep RVFL networks and their solving processes.

3.1. Problem Description

In a distributed learning network based on a point-to-point architecture, we assume that the network has N nodes that are connected to their neighbors and can communicate with each other. The whole dataset is randomly distributed among nodes. Here, we assume that the dataset local to the ith node is $\mathbf{X}^i$ and $\mathbf{Y}^i$, and each node is trained locally for the deep RVFL network. Then in distributed scenarios, the whole global optimization problem becomes minimizing the sum of the loss functions at each node. The following formula is used to express, assuming that the loss function at the ith node is $f_i(\mathbf{z})$, then the global objective function is:

$$\mathbf{z}^* = \arg\min\left\{F(\mathbf{z}) := \sum_{i=1}^{N} f_i(\mathbf{z})\right\}. \tag{9}$$

3.2. Fully Distributed Deep RVFL Network with Direct Links

From the introduction of the second part, we know that in deep RVFL directly connected networks, the optimization of the model is divided into two parts, one is the optimization of the decoder reconstruction weight matrix of the self-encoder in the hidden layer, and the other is the optimization of the RVFL classifier weight matrix. We extend the optimization problem to distributed scenarios. Suppose we are in a topological network of N nodes, and each node only communicates with its neighbors. From the analysis and deduction in the previous subsection, the optimization problem (1) is decomposed into N subproblems for cumulative solution:

$$\hat{\mathbf{U}}_l = \arg\min_{\mathbf{U}_l^i} \frac{1}{2}\sum_{i=1}^{N}\left\|\mathbf{Z}_l^i\mathbf{U}_l^i - \mathbf{H}_{l-1}^i\right\|^2 + \lambda_l\left\|\mathbf{U}_l^i\right\|_1 \tag{10}$$

where λ_l is the regularization parameter of the hidden layer of the lth layer. By solving the above objective function, we obtain the optimal reconstruction matrix $\mathbf{U}_l$ of each hidden layer, and each node can use $\mathbf{U}_l$ to extract the features of the hidden layer. After optimizing the RVFL classifier weight matrix, we assume the same distributed topology scenario, and then problem (3) naturally becomes the following form:

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \frac{1}{2}\sum_{i=1}^{N}\left\|\mathbf{X}_c^i\boldsymbol{\beta}^i - \mathbf{T}^i\right\|^2 + \frac{\lambda}{2}\left\|\boldsymbol{\beta}^i\right\|^2 \tag{11}$$

where $\mathbf{X}_c^i = \left[\mathbf{H}_L^i, \mathbf{X}^i\right]$, $\mathbf{H}_L^i$ is the output value of the hidden layer for each node, $\mathbf{X}^i$ is the original data for each node and $\mathbf{T}^i$ is the target matrix for each node.

3.3. Fully Distributed Deep RVFL Network with Dense Direct Links

Here, the deep RVFL with dense direct links is also extended to a distributed scenario. The difference from the fully distributed deep RVFL network with direct links lies in the connection between the hidden layers. Each hidden layer in the

front and all hidden layers in the back are connected, so that features with lower complexity can be used multiple times, so that the features extracted by the hidden layers are more representative and meaningful. Suppose that on a certain node, the input value $\mathbf{X}_l^i$ of a certain hidden layer can be represented as follows:

$$\mathbf{X}_l^i = \left[\mathbf{X}^i, \mathbf{H}_1^i, \cdots, \mathbf{H}_{l-1}^i\right]. \tag{12}$$

The distributed optimization problem can then look at problems (10) and (11) for the optimization problem of the entire network, as in the case of direct connections.

3.4. Fully Distributed Solutions

For the above objective function to solve the global optimal weight matrix, there are two aspects of the problem, one is to minimize the sum of loss functions, the other is to achieve global consistency, which is actually an optimization problem with constraints. For such problems, ADMM method can be used to solve. Below we outline the principles of ADMM.

ADMM algorithm combines Lagrangian multiplier method and dual decomposition, and solves the original problem by optimizing the original problem and dual problem alternately. ADMM is typically applied to constrained optimization problems of the form:

$$\begin{aligned} &\min \quad f_1(\boldsymbol{\theta}_1) + g_2(\boldsymbol{\theta}_2) \\ &\text{s.t.} \quad \mathbf{P}_1\boldsymbol{\theta}_1 + \mathbf{P}_2\boldsymbol{\theta}_2 - \mathbf{R} = 0. \end{aligned} \tag{13}$$

The core idea of ADMM is to transform constrained optimization problems into equivalent unconstrained ones, and this process realizes the interpretation of constraints by introducing Lagrangian multiplier terms. In this way, we obtain the augmented Lagrangian function of the above problem, and then find its partial derivative to obtain the specific iterative formula of variables.

In addition, there have been many literatures on the convergence analysis and convergence rate judgment of distributed ADMM algorithm, and it has been proved in [22] that this algorithm converges at the rate $O\left(\frac{1}{k}\right)$.

According to the principle of ADMM above, we set the auxiliary variable $\mathbf{V}_l$ so that the parameters of each node converge to the same value. Then, problem (10) is rewritten as follows:

$$\begin{aligned} &\min \quad \frac{1}{2}\sum_{i=1}^{N}\left\|\mathbf{Z}_l^i\mathbf{U}_l^i - \mathbf{H}_{l-1}^i\right\|^2 + \lambda_l\left\|\mathbf{V}_l\right\|_1 \\ &\text{s.t.} \quad \mathbf{U}_l^i - \mathbf{V}_l = 0, i = 1, 2, \cdots, N \end{aligned} \tag{14}$$

where λ_l denotes the regularization parameter for each hidden layer, then we obtain the augmented Lagrangian for the above problem as follows:

$$\begin{aligned} L_{\rho_l}\left(\left\{\mathbf{U}_l^i\right\}, \mathbf{V}_l, \left\{\boldsymbol{\mu}_l^i\right\}\right) = &\frac{1}{2}\sum_{i=1}^{N}\left\|\mathbf{Z}_l^i\mathbf{U}_l^i - \mathbf{H}_{l-1}^i\right\|^2 + \lambda_l\left\|\mathbf{V}_l\right\|_1 + \sum_{i=1}^{N}\left(\boldsymbol{\mu}_l^i\right)^{\mathrm{T}}\left(\mathbf{U}_l^i - \mathbf{V}_l\right) \\ &+ \frac{\rho_l}{2}\sum_{i=1}^{N}\left\|\mathbf{U}_l^i - \mathbf{V}_l\right\|^2. \end{aligned} \tag{15}$$

For each of the hidden layers, where $\boldsymbol{\mu}_l^i$ is the dual variable of the ith node,

ρ_l is the penalty term. In each iteration process, the local objective functions of $\mathbf{U}_l^i$ and $\mathbf{V}_l$ are first optimized alternately, and then the dual variable $\boldsymbol{\mu}_l^i$ is updated, and the iteration formula is as follows:

$$\mathbf{U}_l^i(t+1) = \arg\min_{\mathbf{U}_l^i} L_{\rho_l}\left(\mathbf{U}_l^i, \mathbf{V}_l(t), \boldsymbol{\mu}_l^i(t)\right) \tag{16}$$

$$\mathbf{V}_l(t+1) = \arg\min_{\mathbf{V}_l^i} L_{\rho_l}\left(\mathbf{U}_l^i(t+1), \mathbf{V}_l, \boldsymbol{\mu}_l^i(t)\right) \tag{17}$$

$$\boldsymbol{\mu}_l^i(t+1) = \boldsymbol{\mu}_l^i(t) + \rho_l\left(\mathbf{U}_l^i(t+1) - \mathbf{V}_l(t+1)\right) \tag{18}$$

where t represents the tth iteration. Equations (16) and (17) can be calculated to obtain closed solutions. Then, we can obtain the iterative steps as follows:

$$\mathbf{U}_l^i(t+1) = \left(\left(\mathbf{Z}_l^i\right)^{\mathrm{T}}\mathbf{Z}_l^i + \rho_l\mathbf{I}\right)^{-1}\left(\left(\mathbf{Z}_l^i\right)^{\mathrm{T}}\mathbf{H}_{l-1}^i + \rho_l\mathbf{V}_l(t) - \boldsymbol{\mu}_l^i(t)\right) \tag{19}$$

$$\mathbf{V}_l(t+1) = S_{\lambda_l/N\rho_l}\left(\hat{\mathbf{U}}_l + \hat{\boldsymbol{\mu}}_l\right) \tag{20}$$

$$\boldsymbol{\mu}_l^i(t+1) = \boldsymbol{\mu}_l^i(k) + \rho_l\left(\mathbf{U}_l^i(t+1) - \mathbf{V}_l(t+1)\right) \tag{21}$$

where $\hat{\mathbf{U}}_l = \frac{1}{N}\sum_{i=1}^{N}\mathbf{U}_l^i(t+1)$, $\hat{\boldsymbol{\mu}}_l = \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{\mu}_l^i(t)$, are the average of global nodes. In master-slave mode, this requires a central node to aggregate information from all nodes to compute. Here, we use the decentralized average consensus (DAC) algorithm to achieve global average consistency only by communication between nodes, instead of the role of central nodes, thus avoiding the existence of central nodes and realizing decentralized distributed optimization. We obtain an estimate of the mean value by (7) and (8).

In addition, $\mathcal{S}_\kappa(\cdot)$ stands for the element-wise soft threshold operator [23], which is defined as follows:

$$\mathcal{S}_\kappa(a) = \begin{cases} a-\kappa, & a > \kappa \\ 0, & |a| \le \kappa \\ a+\kappa, & a < -\kappa. \end{cases} \tag{22}$$

Through the above calculation, we find the optimal reconstruction matrix $\hat{\mathbf{U}}_l^i$ of each hidden layer, the data enters each hidden layer to find the optimal reconstruction matrix and then enters the next layer, and the optimization of the hidden layer is completed before the optimization of the RVFL classifier.

For problem (11), we also use ADMM combined with DAC to solve it, set auxiliary variable $\mathbf{V}$, so (11) is rewritten as follows:

$$\begin{aligned} &\min \quad \frac{1}{2}\sum_{i=1}^{N}\left\|\mathbf{X}_c^i\boldsymbol{\beta}^i - \mathbf{T}^i\right\|^2 + \frac{\lambda}{2}\|\mathbf{V}\|^2 \\ &\text{s.t.} \quad \boldsymbol{\beta}^i - \mathbf{V} = 0, i = 1, 2, \cdots, N. \end{aligned} \tag{23}$$

We get the augmented Lagrange function as follows:

$$\begin{aligned} L_\rho\left(\left\{\boldsymbol{\beta}^i\right\}, \mathbf{V}, \left\{\mu^i\right\}\right) = &\frac{1}{2}\sum_{i=1}^{N}\left\|\mathbf{X}_c^i\boldsymbol{\beta}^i - \mathbf{T}^i\right\|^2 + \frac{\lambda}{2}\|\mathbf{V}\|^2 + \sum_{i=1}^{N}\left(\boldsymbol{\mu}^i\right)^{\mathrm{T}}\left(\boldsymbol{\beta}^i - \mathbf{V}\right) \\ &+ \frac{\rho}{2}\sum_{i=1}^{N}\left\|\boldsymbol{\beta}^i - \mathbf{V}\right\|^2. \end{aligned} \tag{24}$$

Then, the ADMM iterations are as follows:

$$\boldsymbol{\beta}^i(t+1)=\left(\left(\mathbf{X}_c^i\right)^{\mathrm{T}}\mathbf{X}_c^i+\rho\mathbf{I}\right)^{-1}\left(\left(\mathbf{X}_c^i\right)^{\mathrm{T}}\mathbf{T}^i+\rho\mathbf{V}(t)-\boldsymbol{\mu}^i(t)\right) \tag{25}$$

$$\mathbf{V}(t+1)=\frac{\rho\hat{\boldsymbol{\beta}}+\hat{\boldsymbol{\mu}}}{\rho+\frac{\lambda}{N}} \tag{26}$$

$$\boldsymbol{\mu}^i(t+1)=\boldsymbol{\mu}^i(t)+\rho\left(\boldsymbol{\beta}^i(t+1)-\mathbf{V}(t+1)\right) \tag{27}$$

where $\hat{\boldsymbol{\beta}}=\frac{1}{N}\sum_{i=1}^{N}\boldsymbol{\beta}^i(t+1)$ and $\hat{\boldsymbol{\mu}}=\frac{1}{N}\sum_{i=1}^{N}\boldsymbol{\mu}^i(t)$ in (26) are the average value of the global nodes, and the DAC algorithm is also used to obtain the average value, and the calculation is carried out according to Formulas (7) and (8). Through the calculation of the above formula, the global optimal value of the RVFL classifier weight matrix is finally obtained.

In order to understand the training process of the distributed algorithm more clearly, the pseudocode of Algorithm 1 shows the iterative steps of the decentralized distributed algorithm in the directly connected deep RVFL network. The algorithm for dense connections is similar to Algorithm 1 and will not be repeated here.

Algorithm 1: D-sdRVFL

Input: $\mathbf{X}^i$, $\mathbf{Y}^i$, λ_l, ρ_l, Number of hidden layers L, maximum number of iterations T
Output: $\hat{\mathbf{U}}_l$, $\hat{\beta}$
Initialization: $\mathbf{U}_l^i(0)=0$, $\boldsymbol{\beta}^i(0)=0$, $\mathbf{V}_l(0)=0$, $\mathbf{V}(0)=0$, $\boldsymbol{\mu}_l^i(0)=0$, $\boldsymbol{\mu}^i(0)=0$
`/* Distributed optimization for each hidden layer l */`
Randomly select $\mathbf{W}_l^i$, b_l^i;
(For node i in the network)
for $i=1,\ldots,N$ **do**
  Calculate $\mathbf{Z}_l^i$ according to the corresponding activation function;
  for $t=1,\ldots,T$ **do**
    Calculate $\mathbf{U}_l^i(t+1)$ according to (19);
    Calculating $\hat{\mathbf{U}}_l$ and $\hat{\boldsymbol{\mu}}_l$ according to (7) and (8) (Through communication between nodes);
    Calculate $\mathbf{V}_l(t+1)$ according to (20);
    Calculate $\boldsymbol{\mu}_l^i(t+1)$ according to (21);
    Calculate residuals and check the termination criterion.;
  Obtain $\hat{\mathbf{U}}_l$;
  Calculate $\mathbf{H}_l^i$ according to (5);
Obtain $\mathbf{H}_L^i$;
`/* Distributed optimization for RVFL output layer */`
Calculate $\mathbf{X}_c^i=\left[\mathbf{H}_L^i,\mathbf{X}^i\right]$;
for $t=1,\ldots,T$ **do**
  Calculate $\boldsymbol{\beta}^i(t+1)$ according to (25);
  Calculating $\hat{\boldsymbol{\beta}}$ and $\hat{\boldsymbol{\mu}}$ according to (7) and (8) (Through communication between nodes);
  Calculate $\mathbf{V}(t+1)$ according to (26);
  Calculate $\boldsymbol{\mu}^i(t+1)$ according to (27);
  Calculate residuals and check the termination criterion.;
return $\hat{\beta}$

4. Experiments and Analysis

In order to verify the effectiveness and feasibility of the proposed algorithm, and

the robustness of the algorithm in the face of network layer number changes. We designed two experiments. The first part of the experiment is mainly to compare with other algorithms in terms of performance, by comparing the model accuracy and training time of each model algorithm on the same data set. In the second part, we change the number of hidden layers of the depth model to observe the accuracy and training time of the proposed distributed algorithm, and verify its robustness.

We will introduce the experimental setup below, including a brief description of the dataset, metrics to measure the accuracy of the model, a description of the training time of the model, and the selection and parameter setting of the model algorithm compared with it. Make the superiority of the proposed algorithm more convincing.

4.1. Experimental Setup

4.1.1. Training Datasets

In the selection of data, we use the data sets used for classification tasks on the classical UCI dataset, carefully selected according to the size of the data set, there are large data sets with a total data volume of more than one million, and there are small data sets with a total data volume of less than ten thousand. Minmax normalization is performed on the data, and the performance of the observation model on different orders of magnitude data sets is better. Details about the dataset are presented in Table 1, and further descriptions of the data can be found on the UCI dataset website.

4.1.2. Evaluation Index

In the accuracy evaluation of the model, we select the classification accuracy as the evaluation index. The closer the classification prediction of the model is to the actual situation, the higher the accuracy of the model. The calculation formula for the classification accuracy is as follows:

$$\text{CAR} = \frac{\text{the number of correctly classified samples}}{\text{the total number of samples}} \times 100\%. \tag{28}$$

In terms of training time, we measure the training time of each node. For example, in a centralized model, there are no redundant nodes, so the training time

Table 1. Overview of the UCI datasets.

Dataset	#Patterns	#Features	#Classes
bank	4521	17	2
credit-approval	690	15	2
glass	214	9	6
musk-2	6598	166	2
statlog-image	2310	18	7
waveform	5000	21	3

of its nodes is the training time of the model. In a distributed model, because multiple nodes participate in each optimization, the training time of each node needs to be divided by the corresponding number of nodes and then compared with the centralized model.

4.1.3. Testing Models and Parameter Setting

For comparison model selection, we not only compare the proposed distributed algorithm model with the corresponding centralized model, but also select two representative deep random weight neural networks H-ELM and ML-KELM and centralized deep RVFL models sdRVFL (d) and sdRVFL (dense) as comparison objects for vertical and horizontal comparison.

We set all the models for comparison, and they keep consistent in the number of hidden layers and neurons to ensure the rationality of comparison. In this paper, the number of hidden layers is set to 3, the number of neurons is fixed to 32, and other parameters are simulated according to the optimal values mentioned in the paper where the model is located. For centralized depth RVFL and distributed depth RVFL, we uniformly adjust regularization term λ and Lagrangian parameter ρ synchronously, λ is set to λ = 0.01, 0.1, 1.10, 100, ρ is set to ρ = 0.01, 0.1, 1, 10, 100. The maximum iteration number of DAC algorithm is 500, and the iteration termination limit of DAC algorithm is 0.001.

4.2. Performance

4.2.1. Classification Accuracy

Through experimental verification on 6 classification data, as shown in Table 2 above, we find that our proposed distributed depth models D-sdRVFL(d) and D-sdRVFL (dense) have good performance on classification tasks, and participate in the comparison of centralized depth models sdRVFL(d − I_1/I_2) and sdRVFL(dense − I_1/I_2) and H-ELM models differ only 3% to 4% in classification accuracy on average, and ML-KELM models differ less than 1% in classification accuracy on average, indicating that our proposed distributed depth model can match the performance of centralized models. In addition, the classification accuracy of D-sdRVFL(dense) model is higher than that of D-sdRVFL(d) model.

Table 2. CAR (%) for different algorithms on the test datasets.

Dataset	H-ELM	ML-KELM	sdRVFL(d − I_1/I_2)	sdRVFL(dense − I_1/I_2)	D-sdRVFL(d)	D-sdRVFL(dense)
musk-2	94.65	84.60	94.49	95.20	88.38	88.59
waveform	86.27	85.60	84.67	85.73	81.47	82.40
bank	89.08	88.63	89.00	89.15	88.63	88.56
glass	58.46	49.23	58,46	60.00	56.92	56.92
statlog-image	93.91	82.61	91.74	92.17	87.97	89.28
credit-approval	81.46	85.85	81.48	87.04	77.78	79.26
Mean Acc.	83.97	79.42	83.30	84.88	80.19	80.84

4.2.2. Training Time

As shown in Table 3, we observe that for the D-sdRVFL(d) and D-sdRVFL(dense) models with 5 agents and 3 hidden layers, the actual training time per agent is slightly higher than that of the centralized model, but the training time is greatly reduced compared to the ML-KELM model. In the following experiments, we discussed the change of training time of each agent in distributed model after changing the number of hidden layer network layers in the network. We found that with the increase of network layers and the number of agents, the training time of single agent will decrease continuously. On the contrary, the training time of centralized model will increase continuously.

4.3. Correlation Analysis of Model Robustness

In this experiment, we change the number of hidden layers in the network to observe the changes in classification accuracy and training time. Three representative data sets were selected as the data sets of this experiment, namely musk-2, waveform and credit-approval. These three data sets also represent large, medium and small data sets.

As shown in Figure 1, in this experiment we compared the classification accuracy and training time of two centralized depth models and two proposed distributed models, and the number of hidden layers changed from 3 to 7. For the model classification accuracy, on the dataset Waveform, the model classification accuracy of centralized deep RVFL model and distributed deep RVFL model does not change significantly with the increase of network layers, the difference between the highest and lowest is less than 2%, there is no obvious increase and decrease, and the highest accuracy does not appear in the model with the most layers. In Musk-2, the classification accuracy of centralized deep RVFL model and D-sdRVFL(d) model does not change significantly with the increase of network layers, while in D-sdRVFL(dense) model, the classification accuracy of model increases with the increase of network layers, and reaches the highest when the number of hidden layers reaches 6, and decreases after reaching 7 layers. In credit-approval dataset, the classification accuracy of centralized deep RVFL model and D-sdRVFL(d) model increases first and then decreases with the increase of network layers, while in D-sdRVFL(dense) model, the classification

Table 3. Average training time (s) per node for different algorithms on training datasets.

Dataset	H-ELM	ML-KELM	sdRVFL(d – I_1/I_2)	sdRVFL(dense – I_1/I_2)	D-sdRVFL(d)	D-sdRVFL(dense)
musk-2	0.138	19.275	0.097	0.231	0.430	0.840
waveform	0.040	7.583	0.031	0.046	0.114	0.240
bank	0.041	5.424	0.019	0.037	0.060	0.092
glass	0.062	0.012	0.014	0.023	0.083	0.160
statlog-image	0.033	2.135	0.032	0.052	0.116	0.234
credit-approval	0.057	0.116	0.014	0.024	0.102	0.212

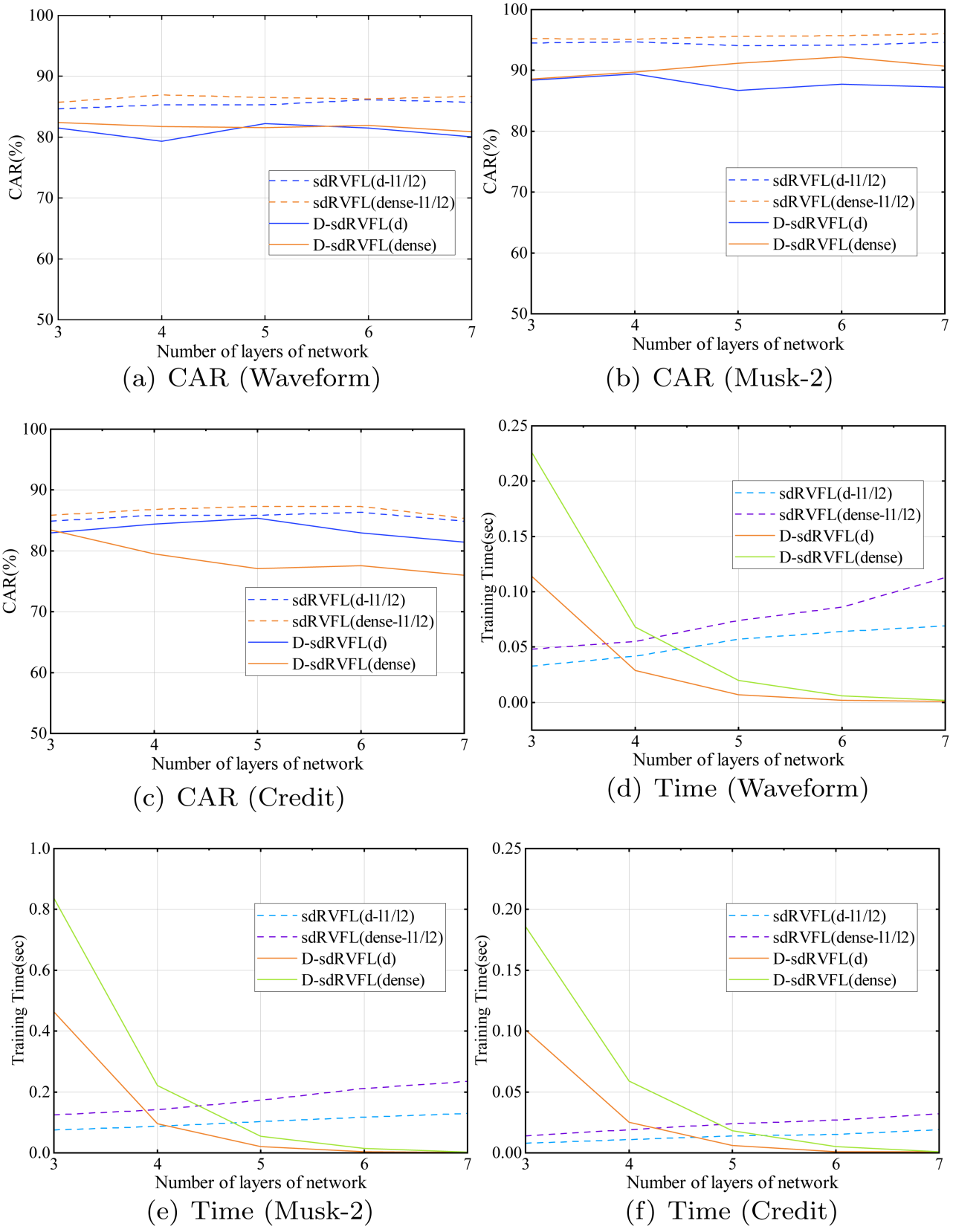

Figure 1. Comparison of CAR and training time between distributed deep network model and centralized deep network model with the number of layers of the network changing, other parameters unchanged.

accuracy decreases gradually with the increase of network layers. Each model shows different characteristics on different data sets, but when other parameters are fixed and only the number of layers is changed, the classification accuracy of the model does not change greatly, the maximum change is not more than 7%, most of them are concentrated in about 2%, and the change of distributed model is slightly larger than that of centralized model, thus verifying the robustness of the model.

As for the training time of the model, it can be seen from the training results of the three data sets that the training time of a single node in the centralized model will increase with the increase of the number of hidden layers of the network, while the training time of each node in the distributed model will gradually decrease with the increase of the number of layers of the network, and the average training time of each node in the distributed network will be lower than that of the centralized network when the number of layers of the network is greater than 4. With the increase of the number of layers of the network, distributed networks have more and more obvious advantages in training time, but can maintain robustness in training effect.

5. Conclusions

Based on the deep RVFL model, this paper proposes a completely distributed deep RVFL algorithm. In the fully distributed framework, agents in the network topology only communicate with each other, and do not need to interact with the original data. At the same time, DAC and ADMM algorithms are used to achieve collaborative optimization between agents in hidden layer and output layer, avoiding the existence of central servers and effectively protecting data privacy. Through experiments on several representative classification data sets show that the proposed algorithm has good classification accuracy and can greatly save the training time of each agent. At the same time, the robustness of the model is verified by changing the number of hidden layers.

The outlook for future work is mainly divided into two aspects. Firstly, in the aspect of algorithm, DAC and ADMM algorithms are used for collaborative optimization, which needs two iterations and consumes more training time. In the later research, other collaborative optimization methods will be selected to reduce the number of iterations in the process, thus further reducing the training time. Second, in terms of model application, relevant experiments have been carried out only on classification tasks to verify the effectiveness of the model, while experiments on other tasks of machine learning need to be expanded and verified.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China (No. 62166013), the Natural Science Foundation of Guangxi (No. 2022GXNSFAA035499) and the Foundation of Guilin University of Technol-

ogy (No. GLUTQD2007029).

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Gupta, D. and Rani, R. (2019) A Study of Big Data Evolution and Research Challenges. *Journal of Information Science*, **45**, 322-340. https://doi.org/10.1177/0165551518789880

[2] Peteiro-Barral, D. and Guijarro-Berdinas, B. (2013) A Survey of Methods for Distributed Machine Learning. *Progress in Artificial Intelligence*, **2**, 1-11. https://doi.org/10.1007/s13748-012-0035-5

[3] Dean, J., *et al*. (2012) Large Scale Distributed Deep Networks. *NIPS*'12: *Proceedings of the* 25*th International Conference on Neural Information Processing Systems*, **1**, 1223-1231.

[4] Abadi, M., *et al*. (2016) TensorFlow: A System for Large-Scale Machine Learning. 12*th USENIX Symposium on Operating Systems Design and Implementation*, Savannah, 2-4 November 2016, 265-283.

[5] Sergeev, A. and Del Balso, M. (2018) Horovod: Fast and Easy Distributed Deep Learning in Tensorflow. arXiv: 1802.05799.

[6] Zhang, D., Chen, X., Wang, D. and Shi, J. (2018) A Survey on Collaborative Deep Learning and Privacy-Preserving. 2018 *IEEE* 3*rd International Conference on Data Science in Cyberspace* (*DSC*), Guangzhou, 18-21 June 2018, 652-658. https://doi.org/10.1109/DSC.2018.00104

[7] Li, P., Li, J., Huang, Z., Li, T., Zhi Gao, C., Yiu, S. and Chen, K. (2017) Multi-Key Privacy-Preserving Deep Learning in Cloud Computing. *Future Generation Computer Systems*, **74**, 76-85. https://doi.org/10.1016/j.future.2017.02.006

[8] Kwabena, O., Qin, Z., Zhuang, T. and Qin, Z. (2019) Mscryptonet: Multi-Scheme Privacy-Preserving Deep Learning in Cloud Computing. *IEEE Access*, **7**, 29344-29354. https://doi.org/10.1109/ACCESS.2019.2901219

[9] Nedic, A., Olshevsky, A. and Rabbat, M. (2017) Network Topology and Communication-Computation Tradeoffs in Decentralized Optimization. *Proceedings of the IEEE*, **106**, 953-976. https://doi.org/10.1109/JPROC.2018.2817461

[10] Liang, Y., Cai, Z., Yu, J., Han, Q. and Li, Y. (2018) Deep Learning Based Inference of Private Information Using Embedded Sensors in Smart Devices. *IEEE Network*, **32**, 8-14. https://doi.org/10.1109/MNET.2018.1700349

[11] Cattivelli, F. and Sayed, A.H. (2010) Diffusion LMS Strategies for Distributed Estimation. *IEEE Transactions on Signal Processing*, **58**, 1035-1048. https://doi.org/10.1109/TSP.2009.2033729

[12] Chang, K., *et al*. (2018) Distributed Deep Learning Networks among Institutions for Medical Imaging. *Journal of the American Medical Informatics Association*: *JAMIA*, **25**, 945-954. https://doi.org/10.1093/jamia/ocy017

[13] Jiang, Z., Balu, A., Hegde, C. and Sarkar, S. (2017) Collaborative Deep Learning in Fixed Topology Networks. 31*st Conference on Neural Information Processing Systems* (*NIPS* 2017), Long Beach, 4-9 December 2017, 5904-5914.

[14] Alhamdoosh, M. and Wang, D. (2014) Fast Decorrelated Neural Network Ensembles

with Random Weights. *Information Sciences*, **264**, 104-117. https://doi.org/10.1016/j.ins.2013.12.016

[15] Pascanu, R., Mikolov, T. and Bengio, Y. (2013) On the Difficulty of Training Recurrent Neural Networks. *International Conference on Machine Learning*, Atlanta, 17-19 June 2013, 1310-1318.

[16] Widrow, B., Greenblatt, A., Kim, Y. and Park, D. (2013) The No-Prop Algorithm: A New Learning Algorithm for Multilayer Neural Networks. *Neural Networks*, **37**, 182-188. https://doi.org/10.1016/j.neunet.2012.09.020

[17] Tang, J., Deng, C. and Huang, G.-B. (2015) Extreme Learning Machine for Multilayer Perceptron. *IEEE Transactions on Neural Networks and Learning Systems*, **27**, 809-821. https://doi.org/10.1109/TNNLS.2015.2424995

[18] Zhang, Y., Wu, J., Cai, Z., Du, B. and Philip, S.Y. (2019) An Unsupervised Parameter Learning Model for RVFL Neural Network. *Neural Networks*, **112**, 85-97. https://doi.org/10.1016/j.neunet.2019.01.007

[19] Katuwal, R. and Suganthan, P.N. (2019) Stacked Autoencoder Based Deep Random Vector Functional Link Neural Network for Classification. *Applied Soft Computing*, **85**, Article ID: 105854. https://doi.org/10.1016/j.asoc.2019.105854

[20] Olfati-Saber, R., Fax, J.A. and Murray, R.M. (2007) Consensus and Cooperation in Networked Multi-Agent Systems. *Proceedings of the IEEE*, **95**, 215-233. https://doi.org/10.1109/JPROC.2006.887293

[21] Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J., *et al.* (2011) Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers. *Foundations and Trends® in Machine Learning*, **3**, 1-122. https://doi.org/10.1561/2200000016

[22] Wei, E. and Ozdaglar, A. (2012) Distributed Alternating Direction Method of Multipliers. 2012 *IEEE* 51*st IEEE Conference on Decision and Control* (*CDC*), Maui, 10-13 December 2012, 5445-5450. https://doi.org/10.1109/CDC.2012.6425904

[23] Bredies, K. and Lorenz, D.A. (2008) Linear Convergence of Iterative Soft Thresholding. *Journal of Fourier Analysis and Applications*, **14**, 813-837. https://doi.org/10.1007/s00041-008-9041-1