Research Article

An Efficient Intrusion Detection Method Based on Federated Transfer Learning and an Extreme Learning Machine with Privacy Preservation

Kunpeng Wang 1, Jingmei Li 1, and Weifei Wu 2

1 College of Computer Science and Technology, Harbin Engineering University, Harbin 150001, China
2 Beijing Institute of Remote Sensing Equipment, Beijing 100080, China

Correspondence should be addressed to Kunpeng Wang; 13351007612@hrbeu.edu.cn

Received 21 December 2021; Revised 21 May 2022; Accepted 28 September 2022; Published 11 October 2022

Copyright © 2022 Kunpeng Wang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Current network security is becoming increasingly important, and intrusion detection is an effective method to protect the network from malicious attacks. This study proposes an intrusion detection algorithm FLTrELM based on federated transfer learning and an extreme learning machine to improve the effect of intrusion detection, which implements data aggregation through federated learning and facilitates the construction of personalized transfer learning for all organizations. FLTrELM first builds a transfer extreme learning machine model to solve the problem of insufficient samples and probability adaptation, then uses the model to learn to protect data privacy without sharing training data under the federated learning mechanism, and finally obtains an intrusion detection model. Experiments on the NSL-KDD, KDD99, and ISCX2012 datasets verify that the proposed method can achieve better detection results and robust performance, especially for small samples and new intrusions, and protects data privacy.

1. Introduction

Currently, computer networks and the Internet are the fundamental components of our society, making great contributions to the economy and impacting people’s work and lifestyle [1]. The network not only has become the foundation of society and modern life but also stores a large amount of data related to people’s private information and national security. At the same time, with the rapid development of network technology, attacks on networks are increasing at an alarming rate. When a network intrusion or an attack occurs, it will inevitably affect our normal activities, threaten national security, and weaken the security of personal information. Therefore, network security has become increasingly important, and the problem of cybersecurity has been the focus of a growing number of people [2–4]. As an emerging security defense technology, the intrusion detection system (IDS) [5, 6] can improve the reliability and security of the system by detecting and responding to various malicious behaviors, while actively protecting the network system from illegal external attacks. IDS aims to classify network access as normal or intrusive based on rules or models. The system that has proven to be efficient and powerful examines network traffic data on computer networks to determine harmful activities and issues alert when such activities are detected. In addition, IDS can improve the reliability and security of cyberspace by detecting and responding to various malicious behaviors and has become an important technical means of protecting cyberspace security against network attacks and intrusions [7].

In recent years, with the rapid development of machine learning, deep learning and artificial intelligence are used for pedestrian detection [8], and their application in intrusion detection has become a commonly studied research topic in the field of network security [9]. Sumaiya Thaseen and
Aswani Kumar [10] proposed an intrusion detection model based on chi square feature selection and a multiclass support vector machine (SVM). The model uses parameter tuning technology to optimize the kernel parameters of the radial basis function. The main idea is to construct a multiclass support vector machine that has not been used for intrusion detection thus far to reduce training and testing time and improve the individual classification accuracy of network attacks. Ikram et al. [11] integrated different deep neural network (DNN) models, such as multilayer perceptron (MLP), backpropagation network (BPN), and long-term and short-term memory (LSTM), to establish a robust anomaly detection model. The model uses XGBoost to integrate the results of each deep learning model to achieve higher accuracy [12]. Based on migration learning and whether IOV can provide a small amount of marker data for new attacks in time, a new model update scheme is proposed. Vehicles can complete the update without obtaining any marker data through the IOV cloud. The experimental results show that compared with the existing methods, the detection accuracy of our model scheme is improved by at least 23%. Xu et al. [13] designed a multisource transfer learning intrusion detection system for privacy protection. First, the system uses Paillier homomorphism to encrypt the models that are trained from different source domains and uploaded to the cloud and then proposes a multisource transfer learning intrusion detection system based on encryption XGBoost (e-XGBoost) based on a privacy protection scheme. Cheng et al. [14] used federal learning (FL) to protect privacy through a local training model and used transfer learning (TL) to improve training efficiency through knowledge transfer.

Compared with other commonly used machine learning algorithms, such as SVM, DBN, and CNN, extreme learning machine (ELM) [15] has the advantages of better performance, faster learning speed, and minimal human intervention, which has quickly gained the attention of researchers. ELM was applied to the network intrusion detection system, effectively improving the detection effect of the network intrusion detection system. Cheng et al. [16] proposed a basic ELM method based on random features and a kernel-based ELM classification method, which is superior to the commonly used SVM method in classification in terms of training and testing speed and detection accuracy. Singh et al. [17] proposed an intrusion detection technology based on an online sequential extreme learning machine (OS-ELM), which uses alpha analysis to reduce time complexity while using feature selection based on filtering, correlation, and consistency discards irrelevant features. Wang et al. [18] proposed a kernel-based extreme learning machine (KELM) with supervised learning capabilities to replace the BP algorithm in DBN to shorten the training cycle. Wang et al. [19] applied an equation-constrained optimization ELM (C-ELM) to network intrusion detection, proposed an adaptive incremental learning strategy to derive the optimal number of hidden neurons, and developed optimization standards and a method that adaptively added hidden neurons through a binary search.

Although ELM has a good application effect in the field of intrusion detection, improves the detection effect, and reduces the false alarm rate, it still faces some following problems. (1) The intrusion detection labeled data generated by a single organization is limited in terms of data volume and data diversity. Additionally, the data itself have a direct impact on the effect of the model, and sufficiently available and high-quality data are required [20]. (2) The distribution of intrusion detection data generated by each organization is different, but for a machine learning algorithm to achieve better learning results and for the data to meet the condition of independent and identical distribution, if the training data of all organizations is directly aggregated, the detection performance of the intrusion model is not ideal [21]. (3) In reality, data usually exist in the form of isolated islands, although each organization contains a wealth of data, which is stored independently of each other and cannot centrally set up data pools to collect and share user data. Therefore, only the performance of the machine learning models trained with independent data from various departments cannot achieve global optimization [22]. (4) Data privacy and security are gaining increasing attention, each organization has a strict privacy policy to protect its own data, and it is forbidden to exchange data without clear user approval [23].

Recently, federated learning [24] has become one of the most promising directions for the future development of machine learning. The purpose of federated learning is to conduct collaborative training without sharing private data. It does not need to aggregate the data required for model training for centralized calculation but transmits encrypted gradient-related data and uses multisource data to collaboratively train the same model [25]. Its emergence allows traditional machine learning models to achieve better training results while ensuring data security and privacy, which has the advantages of distributed collaboration, good scalability, strong privacy protection capabilities, and low cost. Even federated learning is positioned as the last mile of artificial intelligence [23].

In this study, benefiting from the advantages of federated learning, a framework FLTrELM based on federated transfer learning and extreme learning is proposed to solve the problems of data islands, scarce labeled samples, data privacy protection, and personalization in intrusion detection. FLTrELM uses federated learning [26] and homomorphic encryption [27] to build a powerful extreme learning machine model by aggregating data from independent institutions while protecting data privacy. After the model is built, FLTrELM reuses the transfer learning method to achieve personalized model learning for each organization, solves the lack of label samples and data distribution mismatch, and effectively improves the intrusion detection effect.

The algorithms proposed in [10, 11] do not use transfer learning and cannot use relevant data to help target learning tasks create an intrusion detection classifier. Moreover, references [12, 13] are intrusion detection models based on migration learning, but the application scenario of the former is a vehicle network, and they cannot solve the
problem of data islands. Although the literature [14] has the advantages of federated learning and transfer learning, it uses convolutional neural networks and reinforcement learning. The training efficiency of the algorithm is discounted, and the unselected client data are wasted. The final model is a global model, not a personalized model for each mechanism. Compared with them, the FLTrELM algorithm proposed in this study has higher training efficiency, makes full use of the data of each organization, effectively solves the data island, and obtains a personalized learning model for each organization.

Our contributions are highlighted as follows:

(1) To the best of our knowledge, we are the first to apply federated extreme learning transfer learning and ELM to intrusion detection and propose FLTrELM. It aggregates intrusion detection data from different organizations without sacrificing data privacy and security and simultaneously obtains a strong learning model that includes individualized behaviors suitable for each organization through knowledge transfer.

(2) The experimental results show that compared with traditional machine learning methods, FLTrELM substantially improves the accuracy of intrusion detection, especially for fewer samples and new intrusions. Additionally, at the same time, through knowledge transfer, a strong learning model that includes individualized behaviors applicable to various institutions is obtained.

The rest of the study is arranged as follows. Section 2 reviews the related works of federated transfer learning and an extreme learning machine; in Section 3, an intrusion detection algorithm based on federated transfer learning and an extreme learning machine are proposed in this study; in Section 4, the effectiveness of the algorithm is verified on NSL-KDD, KDD99, and ISCX2012 datasets; Section 5 summarizes the main work of this study.

2. Related Works

2.1. Federated Transfer Learning. Federated learning was first proposed by Konen et al. [24] in 2016 and was used by Google to train machine learning models based on mobile phones distributed around the world. Compared with traditional machine learning algorithms that require the last large amount of high-quality data collected from various institutions to be trained on the cloud server for centralized training, it allows each user to train the model on the local machine and upload the model to the server for aggregation after being encrypted. Finally, a global learning model is obtained through multiple iterations. This learning method not only protects the users’ privacy but also does not require data aggregation to cause uncontrollable data flow and sensitive data leaks. The process of federated learning is shown in Figure 1.

Figure 1 shows that the learning process of federated learning is as follows:

(1) The organization downloads the global model \( m_t \) from the central server

(2) Organization \( k \) trains on local data to obtain the local model \( m_{t,k} \) (local model updated in the \( t \)th iteration of \( k \)th organization)

(3) Each organization uploads the locally updated model to the central server

(4) The central server performs a weighted aggregation operation after receiving each mechanism model to obtain the global model \( m_t \) (update the global model at the \( t \)th iteration)

To ensure data privacy, federated learning only allows all remote devices to exchange model gradients with a central server. In this process, each distributed device uses local data to train its own model and then uploads the local model to the central server. After aggregating all the collected models, the server returns the new global model to each device.

According to the different distribution patterns of samples and data feature spaces, federated learning can be divided into three categories, namely, horizontal federated learning, vertical federated learning, and federated transfer learning [26]. Horizontal federated learning is suitable for situations where the user features of the two datasets overlap substantially, but the user overlap is small; vertical federated learning is applicable to the situation where the user features of the two datasets overlap very little, but the user overlaps considerably. Federated transfer learning [28] is different from the previous two federated learning algorithms. It is used when the user and user characteristics of the two datasets rarely overlap, without segmenting the data, but using transfer learning [29] (the transfer of knowledge from an existing field to a new field related to it) to overcome the lack of data or labels is often used to solve the problem of different feature spaces of datasets and the scarcity of label samples. Therefore, the federated transfer learning formed by federated learning + transfer learning has formed a mechanism that can protect privacy and the jointly model as
2.2. Extreme Learning Machine. An extreme learning machine (ELM) is a new learning algorithm proposed by Huang et al. in 2006 [32]. It is used to train the single hidden layer feedforward neural network (SLFN). By randomly selecting the input layer weight and the hidden layer bias, the output layer weight is calculated according to Moore Penrose (MP) generalized inverse matrix theory by minimizing the loss function composed of the training error term and the regular term of the output layer weight norm. As a nonlinear model, ELM has good generalization and nonlinear mapping and can be used to solve dimensional disaster problems. Compared with other machine learning algorithms, such as BP neural networks and SVMs, it has the advantages of fast learning speed, less intervention, and good computational scalability [33].

Given a training dataset \( \{x_i, t_i| x_i \in R^D, t_i \in R^m, i = 1, 2, \ldots, N\} \), \( w_i \) is the weight connecting the \( i \)th neuron, \( b_i \) is the bias of the \( i \)th hidden neuron, \( \beta \) is the output weight vector connecting the neuron, and \( h(x) \) is the output function of the hidden layer neuron. The number of hidden layer nodes of the limit learning machine is \( L \), and \( t_i \) indicates the mark corresponding to the \( i \)th data example. The network structure of the ELM is shown in Figure 2.

As seen in Figure 2, the input of the neural network from left to right is a training sample set \( x = \{x_i, i = 1, 2, \ldots, D_k\} \), there is a hidden layer in the middle, the input layer is fully connected to the hidden layer, and the output matrix \( H(x) \) of the hidden layer is recorded as follows:

\[
H(x) = [h_1(x), \ldots, h_L(x)]^T. \tag{1}
\]

In (1), the form of \( h_j(x) \) is shown as follows:

\[
h_j(x) = g(w_j, b_j, x) = g(w_j x + b_j). \tag{2}
\]

In (2), the function \( g(\cdot) \) represents the activation function [15], which is a nonlinear piecewise continuous function, and commonly used functions include the sigmoid function and the Gaussian function. In this study, the sigmoid function is used. After passing through the hidden layer, it enters the output layer. According to Figure 2 and formula (2), the output of the “generalized” single hidden layer feedforward neural network ELM is as follows:

\[
f_L(x) = \sum_{j=1}^{L} \beta_j h_j(x) = H(x) \beta = t, j = 1, 2, \ldots, m, \tag{3}
\]

where \( \beta = [\beta_1, \ldots, \beta_L] \). The unknown quantities in (2) and (3) are \( w, b, \) and \( \beta \). The ELM adjusts the weights and deviations between neurons according to the training data, and what we actually learn is contained in the connection weights and deviations. Equation (3) can be converted to (4) as follows:

\[
H \beta = T, T = [t_1, \ldots, t_m]^T. \tag{4}
\]

The purpose of ELM is to minimize the training error and obtain the parameters with a better learning effect on the training dataset \( \beta \). In the ideal state, (4) holds, but in practice, the output \( H \beta \) can only be as close to the sample label \( T \) as possible. In this study, the method of minimizing the approximate square difference is used to connect the weight of the hidden layer and the output layer \( \beta \). The objective function is as follows:

\[
\min \|H \beta - T \|^2, \beta \in R^{L \times m}. \tag{5}
\]

According to [34], the Moore Penrose generalized inverse matrix \( H^+ = H^T (H H^T)^{-1} \) is calculated by an orthogonal projection matrix, and the optimal solution is as follows:

\[
\beta^* = H^+ T. \tag{6}
\]

Although ELM has made some achievements, there is still room for improvement. Some scholars have also proposed algorithms to optimize ELM [35, 36]. At present, the traditional extreme learning machine is limited by the fact that the training and the test data meet the constraints of independent and identically distributed and sufficiently available training data. In reality, it is often hoped to use only a small amount of new data and a large amount of historical data to learn an accurate model. With the emergence of transfer learning, it is possible for the further development of ELM.

3. The Proposed FLTrELM Algorithm

3.1. Definition of Problems. Given data \( \{D_1, \ldots, D_k\} \) from \( K \) different organizations (users), the feature spaces of the data are different from each other. An extreme learning machine model \( M_{\text{ALL}} \) is trained on the data of all organizations. In this study, the federated transfer learning model \( M_{\text{FED}} \) is trained through all data collaboration. Among them, no organization will disclose its data to other organizations. For
convenience, binary classification problems are considered, the class label set is \( Y = \{-1, 1\} \), and the multiclassification problem can be extended to the binary classification problem. Assuming that the accuracy rate is expressed as \( Acc \), then the goal of learning is to ensure that the accuracy of federated learning \( Acc_{FED} \) is close to or better than that of the traditional learning model, \( Acc_{ALL} \), as follows:

\[
Acc_{FED} - Acc_{ALL} > \psi,
\]

(7)

where \( \psi \) represents a very small nonnegative real number. Suppose the classification space is \( Y = \{-1, 1\} \). For convenience and without loss of generality, this section only considers the binary classification problem, and the multiclassification problem can be extended to the binary classification problem.

3.2. Framework of FLTrELM. FLTrELM aims to achieve accurate detection of malicious network intrusions through federated transfer learning without sacrificing the privacy and the security of institutional data. Without loss of generality, we assume that there are 3 organizations (users) and 1 server, and more organizations can be expanded under actual conditions. Figure 3 provides an overview of the framework.

In Figure 3, three organizations, A, B, and C, cannot share their own data due to the “wall” of data privacy protection. Their respective datasets are \( D_1, D_2, \) and \( D_3 \), and the models are Model A, Model B, and Model C, respectively. Each organization cannot use each other’s data training models. They can only exchange model knowledge with the cloud model through knowledge migration. In this way, each model establishes contact through the cloud model and breaks the “separation wall.” The framework protects the data privacy of various institutions and can use each other’s model knowledge to help train their own models.

The framework mainly includes the following four procedures:

1. The cloud model on the server is trained according to the public dataset
2. The server distributes the cloud model to all institutions, and each institution can train their own model on their own data (Figure 4 shows knowledge migration process)
3. Each organization’s model is uploaded to its own model to the cloud server, which helps train new cloud models through model aggregation
4. Repeat procedures (1)–(3) until convergence is achieved, and each organization can obtain a personalized model with a good learning effect

Due to the great distribution difference between the server data and the data of each organization, probability adaptation must be carried out with migration learning to obtain a model more suitable for each organization, that is, the migration knowledge process in Figure 3. The detailed process of this procedure is shown in Figure 4. The detailed knowledge migration process is as follows: the cloud model uses the cloud dataset DC training to obtain the model cloud model. This model knowledge is used to help organization \( i \) train model \( m_i \), to migrate the knowledge in the cloud model that is very similar to the model \( m_i \), we select the migrated knowledge through probability adaptation to obtain a new model \( m_i \). It is worth noting that this process encrypts all parameters using data homomorphic encryption. The sharing process does not involve any data leaks [27], meaning that all institutions do not share data. Learning is completed through encrypted shared parameters to protect data privacy.

The learning process of FLTrELM involves model establishment and parameter sharing. After the cloud model is established, it can be directly applied to various organizations. In the actual situation, it is obvious that the samples in the server and the data of various organizations have highly distinct probability differences. Therefore, traditional intrusion detection models fail in personalization, and transfer learning can adapt to the probability differences between models to achieve personalization. In addition, due to the
3.3. Implementation of FLTrELM

3.3.1. Construction of the Transfer Learning Model.

Federal learning solves the problem of data islands between institutions. Therefore, the data of all organizations can be used to build a cloud model, and then each organization can directly use the cloud model. However, due to differences in the distribution of data from various organizations and the cloud data, it is obvious that the model does not perform well for specific users, meaning that it cannot provide users with personalized features. In this study, transfer learning is used to build a personalized model for each user (organization), as shown in Figure 4. In this way, through the acquired cloud model parameters, transfer learning is performed on users to learn their personalized models.

According to [30], the server is the source domain $DS = \{(x'_i, y'_i), i = 1, 2, \ldots, n_1\}$, and organization $i$ is the target domain $DT = \{(x'_i, y'_i), i = 1, \ldots, m\}$. On $DT = \{(x'_i, y'_i), i = 1, \ldots, m\}$, we assume that the parameters of ELM are $\beta_i$, combined with transfer learning and extreme learning machine structure risk minimization theory and the ELM optimization algorithm proposed in [37] to construct the objective function as follows:

$$\arg \min_{\omega, \beta_i} \frac{1}{2} \| \beta_i \|^2 + \frac{C_i}{2} \| \beta_i - \beta_s \|^2 + C_2 \epsilon^i,$$

$$s.t.,$$

$$\epsilon^i \geq 0,$$

$$y'_i(\beta_i \cdot h(x'_i)) \geq 1 - \epsilon^i, i = 1, 2, \ldots, m,$$

$$y'_i(\beta_s^T \cdot h(x'_i) - \beta_i^T \cdot h(x'_i)) \geq 0.$$

The constants $C_i$ and $C_2$ represent the penalty parameters of domain adaptation contribution and target classification error, respectively; $\| \beta_i - \beta_s \|^2$ represents the difference term between the classifier of source and the target domains; constraint $y'_i(\omega_i \cdot h(x'_i)) \geq 1 - \epsilon^i$ indicates that the target domain classifier can classify correctly; constraint $y'_i(\omega_s^T \cdot h(x'_i) - \omega_i^T \cdot h(x'_i)) \geq 0$ ensures that the effect of transfer is not worse than the classification effect only learned on the dataset of the target domain and limits the possibility of a negative transfer. By calculating (7), a classifier is found in the target domain, which can correctly classify the samples in the target domain.

The Lagrangian form of (8) is shown as follows:

$$L(\beta, \epsilon^i, a, \mu, \gamma) = \frac{1}{2} \beta^T \beta_i + \frac{C_i}{2} \| \beta_i - \beta_s \|^2 + C_2 \sum_{i=1}^{m} \epsilon^i$$

$$- \sum_{i=1}^{m} \alpha_i \left( y'_i (\beta_i^T \cdot h(x'_i) - 1 + \epsilon^i) \right)$$

$$- \sum_{i=1}^{m} \mu_i \left( y'_i (\beta_i^T \cdot h(x'_i) + \epsilon^i - \beta_s^T \cdot h(x'_i)) \right)$$

where $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_m)$, $\mu = (\mu_1, \mu_2, \ldots, \mu_m)$, and $\gamma = (\gamma_1, \gamma_2, \ldots, \gamma_m)$ are Lagrange multipliers. $\alpha_i \geq 0$, $\gamma_i \geq 0$, $\mu_i \geq 0$, and $m$ are the number of samples in the target domain.

We find the derivative of the parameter $\beta_i$ sum $\epsilon^i$ of the Lagrange function in (8) and set the derivative to zero to obtain the following formula:

$$\frac{\partial L}{\partial \beta_i} = \beta_i + C_i (\beta_i - \beta_s) - \sum_{i=1}^{m} \alpha_i y'_i h(x'_i)$$

$$- \sum_{i=1}^{m} \mu_i y'_i h(x'_i) = 0,$$

$$\frac{\partial L}{\partial \epsilon^i} = C_2 - \alpha_i - \gamma_i - \mu_i = 0.$$

Simplify (10) and (11) to get the following:

$$\beta_i = \frac{C_i \beta_s}{1 + C_i} + \frac{1}{1 + C_i} \sum_{i=1}^{m} (\alpha_i + \mu_i) y'_i h(x'_i),$$

$$\alpha_i + \gamma_i + \mu_i = C_2.$$

Substituting (12) and (13) into (9), solving ELM is equivalent to solving the dual problem as follows:

$$\min_{\beta} \frac{1}{1 + C_i} \sum_{i=1}^{m} \sum_{j=1}^{m} (\alpha_i + \mu_j) (\alpha_j + \mu_j) y'_i (h(x'_i) \cdot h(x'_j))$$

$$+ \sum_{i=1}^{m} \left( \frac{C_i y'_i (h(x'_i))}{1 + C_i} - 1 \right) \beta_i - \frac{1}{1 + C_i} \| \beta_s \|^2$$

$$0 \leq \alpha_i < C_2, i = 1, 2, \ldots, m,$$

$$0 \leq \mu_i < C_2, i = 1, 2, \ldots, m.$$

Equation (14) is solved by [38], and the result is substituted into (14) to obtain the parameter value of the target domain and obtain the decision function.
3.3.2. Federal Learning Process. FTLTrELM uses federated learning to implement encryption model training and sharing. The learning process of federated learning is mainly composed of two key parts, namely, cloud model learning and user model learning. The basic learning model is an extreme learning machine. \( f_s \) represents the server model to be learned, and the objective function of learning is as follows:

\[
\arg\min_{\Theta} L = \frac{1}{n} \sum_{i=1}^{n} l(y_i, f_S(x_i)),
\]

where \( l(\cdot) \) represents the loss function of the network, \( n \) represents the number of samples on the server, and \( \Theta \) represents the network parameters that need to be learned.

After obtaining the cloud model, \( f_0 \), it is distributed to all organizations. As seen in Figure 3, the organization’s "wall" prohibits the direct sharing of information. This process uses the existing homomorphic encryption technology [27] to avoid information leaks. Through federated learning, user data can be aggregated without affecting privacy and security.

The organizational learning model is expressed as follows:

\[
\arg\min_{\Theta_i} L_t = \frac{1}{n_t} \sum_{i=1}^{n_t} l(y'_i, f_t(x'_i)),
\]

In all organizational models, \( f_t \) is based on a shared cloud model and uploaded to the server for aggregation. In federated learning, good performance can be obtained by sharing initialization and averaging models. Therefore, this study uses the model average to align the user model and averages \( K \) user models in each round of training for cloud model updating. Note that here, we are only averaging the user model, and the updated cloud model is expressed as follows:

\[
f'_S(w) = \frac{1}{M} \sum_{k=1}^{K} f_{1,k}(w),
\]

where \( w \) is a parameter of the network, which represents the learning model of the \( k \)th organization. After enough iterations, the updated server model has better generalizability. Subsequently, new users can participate in the next round of server model training, so FTLTrELM also has the ability to incrementally learn.

3.4. Training FTLTrELM. From Section 3.3, the learning process of the FTLTrELM is given in Algorithm 1. The improved algorithm can continue to work with newly emerging organizational data and update the user model and the cloud model at the same time when faced with new data. Therefore, the longer the FTLTrELM is used, the more personalized the model in each organization, and the better the effect of intrusion detection.

4. Experimental Results

4.1. Experimental Setting Evaluation Criteria. All experiments in this study are performed on a PC machine with a processor Intel Core (TM), 3.6 GHz, 8 GB RAM, and Windows 10 operating system. To verify the effectiveness and the generalization performance of the proposed algorithm in intrusion detection FTLTrELM, three intrusion detection datasets, namely, NSL-KDD, KDD99, and ISCX2012, are used as experimental datasets. The benchmark algorithms selected in the experiment are ELM [32], TrAdaBoost [13, 14, 39], and SVM, among which SVM is implemented using the LIBSVM [37] toolkit.

The 10-fold cross-validation method is a standard method for evaluating machine learning algorithms, and this study uses the intrusion detection model proposed by its evaluation. Specifically, we randomly sample the original dataset into 10 mutually exclusive subsets of equal size. In each run of the model, nine subsets are selected to train the intrusion detection model, and the remaining subset is used to test the model. Therefore, by repeating the above process 10 times, each subset has an equal chance of being selected to train and test the model. Finally, the performance of the proposed detection model is obtained by averaging the results of the test subset. The average of the results of all experiments repeated ten times is used as the final comparison result.

Commonly used evaluation indicators for detection include precision, detection rate, accuracy, false-positive rate, and miss rate. Precision reflects the proportion of correctly classified samples to the total number of samples; the larger the value, the better. Accuracy reflects the proportion of true positive samples to the total number of samples classified as positive, and the larger the value, the better. The detection rate reflects the proportion of positive samples classified as positive in all positive samples. Accuracy and detection rate are a pair of contradictory indicators. The higher the accuracy, the fewer false-positives, and the higher the detection rate, the fewer false-negatives. If there is more precision, the detection rate will increase, but the accuracy will decrease, and vice versa. In intrusion detection, the false-positive rate refers to the proportion of misclassified positive samples to all negative samples. The smaller the value is, the better and the higher it is; it will be prone to "the wolf is coming."

The formal description of the precision, detection rate, accuracy, false-positive rate, and miss rate is as follows:

\[
\text{Precision} : CR = \frac{TP}{TP + FP} \times 100\%,
\]
\[
\text{Detection rate} : DR = \frac{TP}{TP + FN} \times 100\%,
\]
\[
\text{Accuracy} : ACC = \frac{TP + TN}{TN + FP + FN + TP} \times 100\%,
\]
\[
\text{False – positive rate} : FR = \frac{FP}{TN + FP} \times 100\%,
\]
\[
\text{Miss rate} : MR = \frac{FN}{TP + FN} \times 100\%.
\]

where TP represents the number of positive samples that are correctly classified as positive samples, FP represents the
number of negative samples that are incorrectly classified as positive samples, TN represents the number of negative samples that are correctly classified as negative samples, and FN represents the number of positive samples that are incorrectly classified as negative samples.

In the work of this study, the average accuracy rate and false alarm rate of the experimental results obtained by the 10-fold cross-validation method are used as overall evaluation indicators to verify the effectiveness and the accuracy of the algorithm.

4.2. Dataset. This section describes the NSL-KDD, KDD99, and ISCX2012 datasets and preprocesses them.

4.2.1. Dataset

(1) ISCX2012 Dataset. Researchers noticed that the attack types considered in the KDD99 intrusion detection dataset are now out of date. In 2012, the center of Information Security Excellence (ISCX) of the University of New Brunswick (UNB) released an intrusion detection dataset named ISCX2012 [34]. This dataset contains seven days of original network traffic data, including normal traffic and four intrusion types, namely, DoS, Prob, R2L, and U2R (see Table 1, for details). In the experiment, 2% of the data are selected from the training dataset, most of the labeled information is deleted as the source domain dataset, the remaining labeled data are composed of the target domain dataset, and the two datasets together constitute the training dataset; similarly, 1% of the data are taken from the test dataset as the test dataset.

(2) KDD 99. KDD99 is a widely used competition data for intrusion detection provided by the Lincoln Laboratory of Massachusetts Institute of Technology. It is an intrusion detection dataset with the best influence and credibility in academia [35]. The dataset has 5 \times 10^6 pieces of data, and each piece of data has 41 characteristic attributes and 1 class identifier. There are approximately 38 attack types, of which 21 attack types appear in the training dataset, and another 17 unknown attack types appear only in the test dataset. The purpose of this design is to test the generalizability of the classifier model. The ability to detect unknown attack types is also one of the important indicators to evaluate the effect of classifiers in intrusion detection applications.

Thus far, the most used by researchers is the 10% KDD99 dataset (including the training dataset and the test dataset), which is a sample of 10% of all datasets of the KDD99 dataset, and this dataset is also used in the article. The 10% dataset contains 1 type normal with normal signs and 4 major network attack types, namely, DOS, probing, U2R, and R2L. In the two 10% datasets, the four types of cyber-attacks contain different amounts of attack behavior. Table 2 details 22 attack behaviors in the training dataset and 39 attack behaviors in the test dataset, and normal data are also counted as one type of attack in the table.

For the intrusion detection algorithm to be able to recognize new attack behaviors by learning from the training dataset, the test dataset in Table 3 contains more new attack behaviors than the training dataset. In Table 3, the proportion of normal in the two datasets in the 10% dataset is basically the same, but the proportions of the other four attack types are significantly different; because U2R and R2L have very small proportions, most of the current detection algorithms have difficulty detecting these two types of attacks.

(3) NSL-KDD. NSL-KDD [40] is an optimization of the KDD99 dataset, deleting some duplicate records, including different classification difficulty levels, and the number is more balanced so that it can be used as an effective benchmark dataset to correct and effectively detect the ability of the model. The NSL-KDD dataset includes 4 sub-datasets as follows: KDDTrain+, KDDTrain+20Precent, KDDTest+, and KDDTest+21. This study uses KDDTrain+ for training and KDDTest+ for testing. The dataset contains 4 anomaly types, which are subdivided into 39 attack types, of which 17 unknown attack types appear in the test set. Each record includes 41 characteristics and 1
Among the 41 features, there are 9 basic TCP connection features, 13 TCP connection content features, 9 time-based network traffic statistics features, and 10 host-based network traffic statistics features. The details of the NSL-KDD dataset are shown in Table 4.

4.2.2. Data Preprocessing. In the intrusion detection dataset, there are nonnumerical data and the dimension difference between the values, and these data need to be converted into numerical data and unified dimension processing. Therefore, the data preprocessing operation includes two steps, namely, character type digitization and data normalization.

(1) Character-Type Digitization. The ISCX2012, NSL-KDD, and KDD99 dataset processing methods are also the same. In each record, their symbol characteristics are converted into numerical data by using 1 to N−1 encoding. For example, for KDD99, we convert 3 network connection types, 70 network service types, 23 attack types (including normal type Normal), and 11 network connection states of the character type of the dataset into numerical types. The converted forms of the 11 network connection types are shown in Table 5, and the other character types are similar.

(2) Data Normalization. After digitization, the TrELM algorithm uses MMD to select the data. In the MMD, the distance is calculated. For the continuous feature attributes in the dataset, the measurement methods of each attribute are different, so the calculation of the distance between the data has a greater impact, which in turn affects the accuracy of the calculation results. To avoid the above situation, the difference between the different features can be eliminated. For discrete features, the method of MMD normalization is adopted. For continuous features, the Z-Score method is used to fix the value at [0, 1], as shown in formulas (18) and (19):

\[
x_{mn} = \frac{x - x_{\text{min}}}{x_{\text{max}} - x_{\text{min}}},
\]

\[
x_{zc} = \frac{x - x_{\text{av}}}{\sigma},
\]

where \(x\) is the original sample data, \(x_{\text{max}}\) is the maximum value, \(x_{\text{min}}\) is the minimum value, \(x_{\text{av}}\) is the average value, \(\sigma\) is the standard deviation, and \(x_{mn}\) and \(x_{zc}\) are the normalized results of the original data.

4.3. Experimental Results and Analysis. This section analyzes the experimental results of the SVM, ELM, TrAdaBoost, and FLTrELM algorithms proposed in this study on the NSL-KDD, KDD99, and ISCX2012 datasets to verify their effectiveness. In addition, the influence of the adjustable parameter \(C_1\) on the results of FLTrELM is discussed and analyzed.

Tables 6–8 show the average accuracy, false-positive rate, and miss rate of the algorithm on the NSL-KDD, KDD99, and ISCX2012 datasets. The following conclusions can be obtained:
Table 6: Average accuracy rate, false-positive rate (%), and miss rate on the NSL-KDD dataset (%).

| Algorithm   | DoS   | Normal | Probe | R2L  | U2R  |
|-------------|-------|--------|-------|------|------|
|             | ACC   | FR     | ACC   | FR   | ACC  | FR   | ACC  | FR   | ACC  | FR   | ACC  | FR   | ACC  | FR   | ACC  | FR   | ACC  | FR   | ACC  | FR   | ACC  | FR   | ACC  | FR   | ACC  | FR   | ACC  | FR   | ACC  | FR   | ACC  | FR   |
| FLTrELM     | 99.99 | 2.12   | 1.81  | 99.76 | 2.97 | 2.28 | 99.73 | 3.25 | 3.21 | 90.12 | 5.15 | 6.96 | 70.56 | 5.98 | 6.85 |
| TrAdaBoost  | 98.12 | 2.32   | 2.25  | 98.86 | 3.23 | 2.43 | 98.10 | 3.63 | 3.42 | 73.25 | 9.76 | 9.13 | 47.65 | 9.33 | 10.12 |
| [13]        | 98.33 | 2.25   | 2.36  | 99.15 | 3.27 | 2.38 | 98.64 | 3.68 | 3.45 | 75.47 | 9.58 | 8.87 | 59.43 | 8.47 | 7.12 |
| [14]        | 97.44 | 2.56   | 2.43  | 98.55 | 3.38 | 2.57 | 98.88 | 3.73 | 3.54 | 73.45 | 9.75 | 8.97 | 50.24 | 9.36 | 9.46 |
| ELM         | 97.23 | 2.46   | 3.02  | 97.47 | 4.17 | 3.37 | 97.12 | 4.15 | 4.37 | 68.17 | 10.24 | 9.97 | 32.34 | 10.24 | 11.02 |
| SVM         | 97.18 | 2.97   | 3.21  | 97.43 | 4.24 | 3.47 | 97.24 | 4.42 | 4.46 | 64.42 | 11.13 | 9.86 | 31.26 | 10.75 | 11.21 |

Table 7: Average accuracy rate, false-positive rate (%), and miss rate on the KDD99 dataset (%).

| Algorithm   | DoS   | Normal | Probe | R2L  | U2R  |
|-------------|-------|--------|-------|------|------|
|             | ACC   | FR     | ACC   | FR   | ACC  | FR   | ACC  | FR   | ACC  | FR   | ACC  | FR   | ACC  | FR   | ACC  | FR   | ACC  | FR   | ACC  | FR   | ACC  | FR   | ACC  | FR   | ACC  | FR   |
| FLTrELM     | 99.65 | 2.32   | 2.15  | 99.98 | 3.11 | 2.28 | 99.53 | 3.49 | 3.26 | 88.48 | 5.23 | 6.11 | 70.35 | 5.29 | 6.89 |
| TrAdaBoost  | 97.23 | 2.45   | 2.37  | 98.45 | 3.34 | 2.65 | 98.73 | 3.82 | 3.56 | 72.34 | 9.85 | 9.23 | 46.65 | 9.42 | 10.23 |
| [13]        | 98.33 | 2.25   | 2.36  | 99.15 | 3.27 | 2.38 | 98.64 | 3.68 | 3.45 | 75.47 | 5.58 | 6.87 | 59.43 | 5.87 | 7.12 |
| [14]        | 97.44 | 2.56   | 2.43  | 98.55 | 3.38 | 2.57 | 98.88 | 3.73 | 3.54 | 73.45 | 9.75 | 8.97 | 50.24 | 9.36 | 9.46 |
| ELM         | 96.65 | 2.53   | 2.98  | 97.47 | 4.11 | 3.47 | 97.65 | 4.26 | 4.43 | 67.34 | 11.23 | 10.12 | 31.87 | 10.35 | 11.13 |
| SVM         | 96.33 | 3.11   | 3.26  | 97.21 | 4.27 | 3.53 | 96.56 | 4.56 | 4.52 | 63.56 | 12.15 | 10.65 | 30.12 | 10.84 | 11.37 |

1) Sufficient available intrusion detection training samples are the basis of high-accuracy classifier training. On the intrusion detection datasets, NSL-KDD and KDD99, there are a large number of these three types of attacks, namely, normal, Probe, and DOS. All algorithms have high accuracy for these three types of attacks, reaching over 96%. In the same way, on the ISCX2012 dataset, the accuracy of all algorithms against a large number of normal, infiltration, and DDoS attacks reached 90%.

2) On the intrusion detection datasets, NSL-KDD and KDD99, for the attack types U2R and R2L with a small number of samples, traditional intrusion detection algorithms are not enough to train and obtain a high-accuracy detection model. Therefore, they have low accuracy against the two types. FLTrELM [13, 14] and TrAdaBoost are transfer learning algorithms that use knowledge from a large number of well-labeled intrusion detection samples to train the detection types for U2R and R2L. Therefore, their detection rates for U2R and R2L will be improved. In Tables 6 and 7, it can be seen that FLTrELM [13, 14] and TrAdaBoost have improved accuracy on U2R and R2L, especially the accuracy rate of R2L. R2L is above 70%, and U2R is above 46%. On the ISCX2012 dataset in Table 8, the transfer learning algorithms, FLTrELM [13, 14] and TrAdaBoost, are more accurate than ELM and SVM for the smaller number of attack types HttpD0s and BFSSH, where FLTrELM performed best. Therefore, FLTrELM has a significant effect on improving the detection rate of U2R and R2L attack types that contain a small number of samples.

3) In terms of the false alarm rate, Tables 6 and 7 show that the false alarm rates of the three intrusion attack behaviors, namely, normal, Probe, and DOS in the intrusion detection datasets NSL-KDD and KDD99 do not exceed 5%. Among them, FLTrELM has the lowest false alarm rate, which is below 6%. In the intrusion detection behavior U2R and R2L, the three nontransfer benchmark algorithms performed poorly. The false alarm rates of R2L and U2R on the NSL-KDD dataset reach 10% and 9% or more, respectively, and on the KDD99 dataset reaches more than 9%. However, TrELM performs relatively well on these two datasets below 6%. On the ISCX2012 dataset, Table 8 shows that TrELM has the lowest false-positive rate among all attack types.

4) In terms of the miss rate, it can be seen from Tables 6–8 that FLTrELM is the lowest among the nine attack behaviors compared with the benchmark algorithm.

The experimental results show that, on the KDD 99 and the NSL-KDD datasets, TrELM’s accuracy for the five attack types is higher than that of the benchmark algorithm, and the accuracy of the attack type with a small number of samples is also considerably improved. In the ISCX2012 dataset, FLTrELM’s accuracy for the 5 attack types is also better than that of the benchmark algorithms.

Therefore, FLTrELM improves the detection rate of all 9 attack behaviors, especially for the R2L attack behavior detection rate with sparse samples; there is no problem where the detection rate of a certain attack behavior is too low, and the detection rate is very different. We effectively reduce the imbalance problem of the attack type detection in the machine learning algorithm; FLTrELM has shown substantial advantages in the false-positive rate and the false-negative rate. In other words, FLTrELM achieves the best classification accuracy for all intrusions. This is because federated learning can indirectly use more information in
distributed data to train better models, and the model becomes more consistent through transfer learning. The characteristics of each organization, compared with traditional methods (SVM and ELM), greatly improve the recognition results. Compared with [13, 14] and TrAdaBoost algorithms, the proposed algorithm still has advantages in terms of accuracy. In short, the experiment proves the effectiveness of FLTrELM, and the proposed algorithm can also protect the privacy of data; thus, it will not be worse than the benchmark algorithm. The experimental results on the three datasets also show that the FLTrELM algorithm has better generalization performance.

Table 9 shows the average training time of the algorithm on three intrusion detection datasets. Compared with the transfer learning algorithms, TrAdaBoost and FLTrELM, the transfer learning algorithms, ELM and SVM, have no transfer learning process and do not need to process additional data, so their training time is relatively low. However, TrAdaBoost and FLTrELM [13, 14] need to transfer the knowledge in the auxiliary data to help build the target learning task, and the training time will increase. FLTrELM [14] needs data from multiple institutions, so the training time will also increase; FLTrELM proposed in this study inherits the characteristics of the fast solution of ELM; compared with similar algorithms [14] that use the CNN, training time is shorter and the algorithm efficiency is higher.

It can be seen from (12) that the objective function of FLTrELM includes the parameter $C_1$, and its value will affect the learning effect, so it is necessary to analyze the sensitivity. On the NSL-KDD, KDD, and ISCX2012 datasets, according to parameter $C_1$, it takes [0, 1] different values to record the changes in the classification effect of the FLTrELM algorithm. The average accuracy rate is shown in Figures 5–7, according to the $C_1$ parameter sensitivity analysis of the experimental results presented by the temperature to illustrate the influence of parameters on the learning effect of FLTrELM.

In Figures 5–7, the conclusions obtained through analysis are as follows:

Using the parameter grid search method provided by [41], the value of $C_1$ is determined, and the experimental results on the real dataset are recorded at the same time.
Taking different values within a certain range, the classification effect of FLTrELM is considerably different. It can be seen that the closer the domain relationship, the greater the value and the higher the accuracy. Therefore, the algorithm is sensitive to the regularization parameter, $C_1$, within a certain value range, and the parameter value when the algorithm classification effect is the best that can be obtained on different cross-domain tasks.

5. Conclusions

In this study, the first algorithm, FLTrELM, based on federated transfer learning and an extreme learning machine for intrusion detection is proposed. FLTrELM aggregates data from different organizations without compromising privacy and security, realizes the adaptation to each user through knowledge transfer, and conducts personalized model learning. The FLTrELM provides a method for future research on intrusion detection. The effectiveness of the algorithm is evaluated in the experiment. The experimental results show that FLTrELM effectively solves the traditional intrusion detection algorithm for small samples and emerging attack behaviors with low detection accuracy, privacy protection, and data islanding problems and improves the detection effect. In the future, our future work is as follows: (1) the difference in the conditional probability of each organization’s data in the process of transfer learning will be given to measuring; (2) the improvement of training efficiency.

Data Availability

The data used to support the findings of this study can be obtained from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest regarding the publication of this study.

Acknowledgments

This work was partially supported by Major Special Fund Projects in Heilongjiang Province, China (Grant no. 2020-230100-54-01-000358).

References

[1] J. Gu and S. Lu, “An Effective Intrusion Detection Approach Using SVM with Naïve Bayes Feature Embedding,” Computers & Security, vol. 103, 2021.
[2] L. Lv, W. Wang, Z. Zhang, and X. Liu, “A novel intrusion detection system based on an optimal hybrid kernel extreme learning machine,” Knowledge-Based Systems, vol. 195, Article ID 105648, 2020.
[3] H. W. Wang, J. Gu, and S. S. Wang, “An effective intrusion detection framework based on SVM with feature augmentation,” Knowledge-Based Systems, vol. 136, pp. 130–139, 2017.
[4] A. L. Buczak and E. Guven, “A survey of data mining and machine learning methods for cyber security intrusion detection,” IEEE Communications Surveys & Tutorials, vol. 18, no. 2, pp. 1153–1176, 2016.
[5] R. Yahalom, A. Steren, Y. Namari, M. Roytman, A. Porgador, and Y. Elovici, “Improving the effectiveness of intrusion detection systems for hierarchical data,” Knowledge-Based Systems, vol. 168, pp. 59–69, 2019.
[6] I. Benmessahel, K. Xie, and M. Chellali, “A new evolutionary neural networks based on intrusion detection systems using multiverse optimization,” Applied Intelligence, vol. 48, no. 8, pp. 2315–2327, 2018.
[7] J. Li, W. Wu, and D. Xue, “An intrusion detection method based on active transfer learning,” Intelligent Data Analysis, vol. 24, no. 2, pp. 363–383, 2020.
[8] Y. Jiang, G. Tong, and H. Yin, “A pedestrian detection method based on genetic algorithm for optimize XGBoost training parameters,” IEEE Access, vol. 99, pp. 118310–118321, 2019.
[9] A. L. Buczak and E. Guven, “A survey of data mining and machine learning methods for cyber security intrusion detection,” IEEE Communications Surveys & Tutorials, vol. 18, no. 2, pp. 1153–1176, 2016.
[10] I. Sumaiya Thaseen and C. Aswani Kumar, “Intrusion detection model using fusion of chi-square feature selection and multi class SVM,” Journal of King Saud University - Computer and Information Sciences, vol. 29, no. 4, pp. 462–472, 2017.
[11] S. T. Ikram, A. K. Cherukuri, B. Poorva et al., “Anomaly detection using XGBoost ensemble of deep neural network models,” Cybernetics and Information Technologies, vol. 21, no. 3, pp. 175–188, 2021.
[12] X. Li, Z. Hu, M. Xu, Y. Wang, and J. Ma, “Transfer learning based intrusion detection scheme for Internet of Vehicles,” Information Sciences, vol. 547, pp. 119–135, 2021.
[13] M. Xu, X. Li, Y. Wang, B. Luo, and J. Guo, “Privacy-preserving multisource transfer learning in intrusion detection system,” Transactions on Emerging Telecommunications Technologies, vol. 32, no. 5, 2021.
[14] Y. Cheng, J. Lu, D. Niyato, B. Lyu, J. Kang, and S. Zhu, “Federated transfer learning with client selection for intrusion detection in mobile edge computing,” IEEE Communications Letters, vol. 26, no. 3, pp. 552–556, 2022.
[15] G. B. Huang, Q. Y. Zhu, and C. K. Siew, “Extreme learning machine: theory and applications,” Neurocomputing, vol. 70, no. 1-3, pp. 489–501, 2006.
[16] C. Cheng, W. P. Tay, and G. Huang, “Extreme learning machines for intrusion detection,” in Proceedings of the 2012.
an intrusion detection system using network traffic profiling and online sequential extreme learning machine," Expert Systems with Applications, vol. 42, no. 22, pp. 8609–8624, 2015.

[18] Z. Wang, Y. Zeng, Y. Liu, and D. Li, "Deep belief network integrating improved kernel-based extreme learning machine for network intrusion detection," IEEE Access, vol. 9, no. 99, Article ID 16062, 2021.

[19] R. Singh, H. Kumar, and R. K. Singla, "An intrusion detection system using network traffic profiling and online sequential extreme learning machine," Expert Systems with Applications, vol. 42, no. 22, pp. 8609–8624, 2015.

[20] J. Li, W. Wu, and Di Xue, "Transfer Naive Bayes algorithm with group probabilities," Applied Intelligence, vol. 50, no. 1, pp. 61–73, 2020.

[21] J. Li, W. Wu, Di Xue, and P. Gao, "Multi-source deep transfer neural network algorithm," Sensors, vol. 19, no. 18, p. 3992, 2019.

[22] H. B. Mcmahan, E. Moore, and D. Ramage, "Communication-efficient learning of deep networks from decentralized data," Artificial Intelligence and Statistics, pp. 1273–1282, 2017.

[23] Q. Yang, "Federated learning: the last on kilometer of artificial intelligence," CAAI transactions on intelligent systems, vol. 15, no. 1, pp. 183–186, 2020.

[24] J. Koenen, H. B. Mcmahan, and D. Ramage, "Federated optimization: distributed machine learning for on-device intelligence," 2016, https://arxiv.org/abs/1610.02527.

[25] J. Koenen, H. B. Mcmahan, and F. X. Yu, "Federated learning: strategies for improving communication efficiency," 2016, https://arxiv.org/abs/1610.05492.

[26] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: concept and applications," ACM Transactions on Intelligent Systems and Technology, vol. 10, no. 2, pp. 1–19, 2019.

[27] C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li, and Y Gao, "A survey on federated learning," Knowledge-Based Systems, vol. 216, no. 1, Article ID 106775, 2021.

[28] Y. Liu, Y. Kang, C. Xing, T. Chen, and Q. Yang, "A secure federated transfer learning framework," IEEE Intelligent Systems, vol. 35, no. 4, pp. 70–82, 2020.

[29] S. J. Pan and Q. Yang, "A survey on transfer learning," IEEE Transactions on Knowledge and Data Engineering, vol. 22, no. 10, pp. 1345–1359, 2010.

[30] X. Liang, Y. Liu, and T. Chen, "Federated transfer reinforcement learning for autonomous driving," arXiv, 2019, https://arxiv.org/abs/1910.06001.

[31] S. Sharma, X. Chaoping, and Y. Liu, "Secure and efficient federated transfer learning," 2019, https://arxiv.org/abs/1910.13271.

[32] G. B. Huang, L. Chen, and C. K. Siew, "Universal approximation using incremental constructive feedforward networks with random hidden nodes," IEEE Transactions on Neural Networks, vol. 17, no. 4, pp. 879–892, 2006.

[33] X. Li, W. Mao, and W. Jiang, "Extreme learning machine based transfer learning for data classification," Neurocomputing, vol. 174, pp. 203–210, 2016.

[34] G. B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), vol. 42, no. 2, pp. 513–529, 2012.