Voting Consensus Based Decentralized Federated Learning

Yan Gou, *Student Member, IEEE,* Shangyin Weng, *Student Member, IEEE,* Muhammad Ali Imran, *Fellow, IEEE,* and Lei Zhang, *Senior Member, IEEE*

***Abstract*—With the fourth industrial revolution, the construction of the Internet of Things (IoT) has developed vigorously, and machine learning is also widely used in IoT management and data processing. Given the existence of massive distributed and private datasets generated by a large number of IoT devices, centralized machine learning is unsatisfactory. Therefore, federated learning (FL), as a distributed learning method, becomes a promising solution. In FL, clients can train models by transferring model parameters to the aggregation server while keeping private data locally. However, FL still relies on a central server, which has questionable reliability. The single point of failure and limited communication resources also hinder the application of FL in the IoT. In this paper, we propose a voting consensus based decentralized federated learning method (VCDFL) by incorporating the leader-candidate-follower hierarchical management method and the consensus based leader election mechanism to solve the single point of failure and exclude outlier models for accelerating convergence during aggregation. Then, we propose a joint decision method to exchange decision information rather than model transfer between clients to further protect privacy and reduce communication overhead while ensuring accuracy. Furthermore, we mathematically derive the probability of successfully electing a leader, the communication efficiency and the joint decision accuracy. We conduct our method in an image recognition scenario. The results show that our joint decision mechanism promotes the accuracy of both system and local decision-making. Meanwhile, the proposed scheme greatly reduces communication costs compared to benchmark learning methods.**

***Index Terms*—Decentralized federated learning, consensus, hierarchical, communication efficient, fault tolerant**

I. Introduction

In the fourth industrial revolution, Internet of Things (IoT) applications that have been proposed in recent years, such as smart cities [1], autonomous vehicles [2], intelligent transportation [3], smart industry [4], intelligent health care [5] are being realized. The deployment of these smart services involves billions of IoT devices [6] that will generate billions of zettabytes of data [7]. Along with the development of mobile network communication technology, artificial intelligence and machine learning are provided with good conditions for rapid development. Against this background, the development of decentralized federated learning (FL) has attracted much attention.

Compared with other centralized machine learning, such as deep learning, FL allows users to store personal training data, train local models, and only transfer model parameters to the central server for global model aggregation. This not only greatly reduces the risk of personal privacy exposure during data transmission but also greatly reduces communication overhead. However, traditional FL is still a centralized architecture that relies on the computing capability of a fixed central server, which leaves the problem of a single point of failure, i.e., any failure of the central server could cause a process outage, stopping the entire learning process. Given that the computing ability of IoT devices has greatly increased in recent years with the development of hardware, end devices can also take on the responsibility of model aggregation to get rid of the dependence on the central server and solve the single point of failure. Therefore, decentralized FL is an appropriate solution.

There are two ways of decentralization; one way is that all users have the function of a central server. For example, in [8], [9], each client in the system broadcasts the local model updates to the other clients and obtains the other models for model aggregation. The disadvantage of this approach is high communication cost and inefficient computation workload. Another way is hierarchical management, where different users have different responsibilities to cooperate [10]–[12]. For example, in [11], the authors divide clients into clusters and introduce leader-follower relationships in each cluster for hierarchical management. In each round, a leader is chosen in every cluster for model collection and aggregation. Then, all leaders cooperatively aggregate one global model and provide feedback to their followers to achieve global information sharing. Compared with the first method, the hierarchical framework has fewer communication costs, higher efficiency, and faster convergence performance.

However, the hierarchical architecture still has some problems that need to be researched. For example, since the aggregated client is responsible for the model collection, aggregation and broadcasting. How to choose the aggregated client (leader) is a very important issue. Some authors choose leaders based on the actual situation of the user. In [12], the authors select one client with high reputation and communication capabilities as the leader for model aggregation management. Nevertheless, other authors select leaders through a voting mechanism. In [13], the authors leverage raft consensus algorithm [14] and introduce the leader-candidate-follower

Yan Gou, Shangyin Weng, Muhammad Ali Imran and Lei Zhang (corresponding author) are all with the James Watt School of Engineering, University of Glasgow, Glasgow, G12 8QQ, UK.
E-mail: y.gou.1@research.gla.ac.uk, s.weng.2@research.gla.ac.uk, Lei.Zhang@glasgow.ac.uk, Muhammad.Imran@glasgow.ac.uk

relationship. In the beginning, all clients are followers who train and provide local models. Those who realize that there is no viable leader become candidates to request votes, and the one who gets enough votes becomes the leader for model aggregation during its tenure. In [15], all clients vote for the new leader they support by secret ballot without a list of candidates. The current leader tallies the votes, and the client with the most votes becomes the next leader. Although all of the above methods can elect suitable clients to serve as leaders, they ignore the quality of the leaders' model, which should also be an important factor in measuring the leader's suitability.

Besides, the accuracy of the local model is different due to the difference in the personal database. If the local model with low accuracy is involved in model aggregation, the convergence speed of the system is delayed, and the accuracy of the system is reduced. Hence, how to select models for aggregation is crucial for FL. In [16] and [17], by calculating phase deviation or cosine similarity between models, only relatively benign models can participate in model aggregation so as to eliminate the outliers of the models. In [18], a formula to calculate reputation values for all clients in the system and set the clients with the highest reputation as the leader group. Then, each leader evaluates the model accuracy of the remaining clients using their local data, unifies all accuracy against other leaders and ranks them to select the models with the highest accuracies for model aggregation. Besides, in [19], clients are ranked based on their ratings and majority vote aggregation is performed. In [20], authors use blockchain technology to combine model aggregation and voting mechanism in smart contracts for model selection and fault tolerance improvement through consensus. Nevertheless, those model selection methods all require the model exchange and then perform screening based on the model effect through model calculation, which not only consumes huge communication resources but also increases the computational burden due to the complexity of the model.

Furthermore, the communication cost is always a bottleneck in decentralized learning networks since there may be millions of IoT devices generating data and training local models at high speed in the network [21]. On the issue of improving communication efficiency and alleviating bandwidth pressure, one way is to reduce the size of the messages communicated at each round. For example, in [22], the authors encode the updated gradients before sending them to the server in a lossy compression way. Then, the server decodes the updated gradients before aggregating. The other way of improving communication efficiency is to reduce the number of model aggregations or model updates. For example, in [23], the authors reduce the number of model aggregations with a dynamic sampling method. However, these methods bring a worse accuracy performance. In addition, the transmission of the model can not fully protect the users' privacy since private data can still be leaked as it can be recovered from the model through reconstruction attacks by malicious clients or servers [24], [25]. Hence, further research is needed to reduce model transfer or transfer information in a more secure manner.

To cope with the questions mentioned above, our answer is a novel voting consensus based decentralized FL algorithm (VCDFL) along with a joint decision mechanism. In this paper, we assume all clients in our system are honest but curious. The following are the major contributions:

1) We propose a voting consensus based leader election mechanism. Based on the leader-candidate-follower hierarchical management framework, followers decide whether to vote based on decision-making information, and only candidates who receive enough votes will become the leader. Local models are also filtered based on voting information for aggregation to speed up convergence. Furthermore, there is no model interaction during the entire voting process, which not only protects the model's privacy but also reduces communication overhead while transmitting information.
2) We propose a joint decision-making mechanism. By following the majority principle, the decisions of the few are replaced by the majority, which will improve fault tolerance and the accuracy of decision-making. In addition, replacing model transmission with decision interaction can greatly reduce communication costs while protecting personal privacy.
3) In the algorithm analysis, we derive the probability of leader election under different local decision distributions. The mathematical derivations of communication efficiency and joint decision accuracy are also presented.
4) We conduct simulations based on IID data and non-IID data in an image recognition scenario to evaluate the performance of our method. Results demonstrate that our method can save up to 88% (64.4%) communication costs using IID (non-IID) data and increase the accuracy compared with the traditional centralized FL.

The rest of the paper is organized as follows. A detailed voting consensus based decentralized federated learning framework and descriptions are presented in Section II, followed by Section III, where we analyse the performance of our algorithm theoretically. The simulation results are presented in Section IV. Finally, the paper concludes in Section V. A summary of basic notations is provided in Table I.

II. The voting consensus based decentralized federated learning framework

This section introduces the training process of VCDFL as shown in Figure 1, which has four stages: 1) local learning stage (Step 1); 2) voting consensus based leader election stage (Steps 2-6); 3) model aggregation stage (Steps 7-10); and 4) joint decision stage (Steps 11-14). Every client in the system can switch among three roles: Leader, Candidate and Follower.

A. Local learning stage

In the local learning stage, clients train their local model with their local dataset. In order to accelerate the learning process, there can be multiple stochastic gradient descent (SGD) training rounds within this step. After q local updating rounds ($q \geq 1, q \in Z$), clients make local decisions with their local models. Local decisions are referred to the results associated with or applied to the local models in an application. For

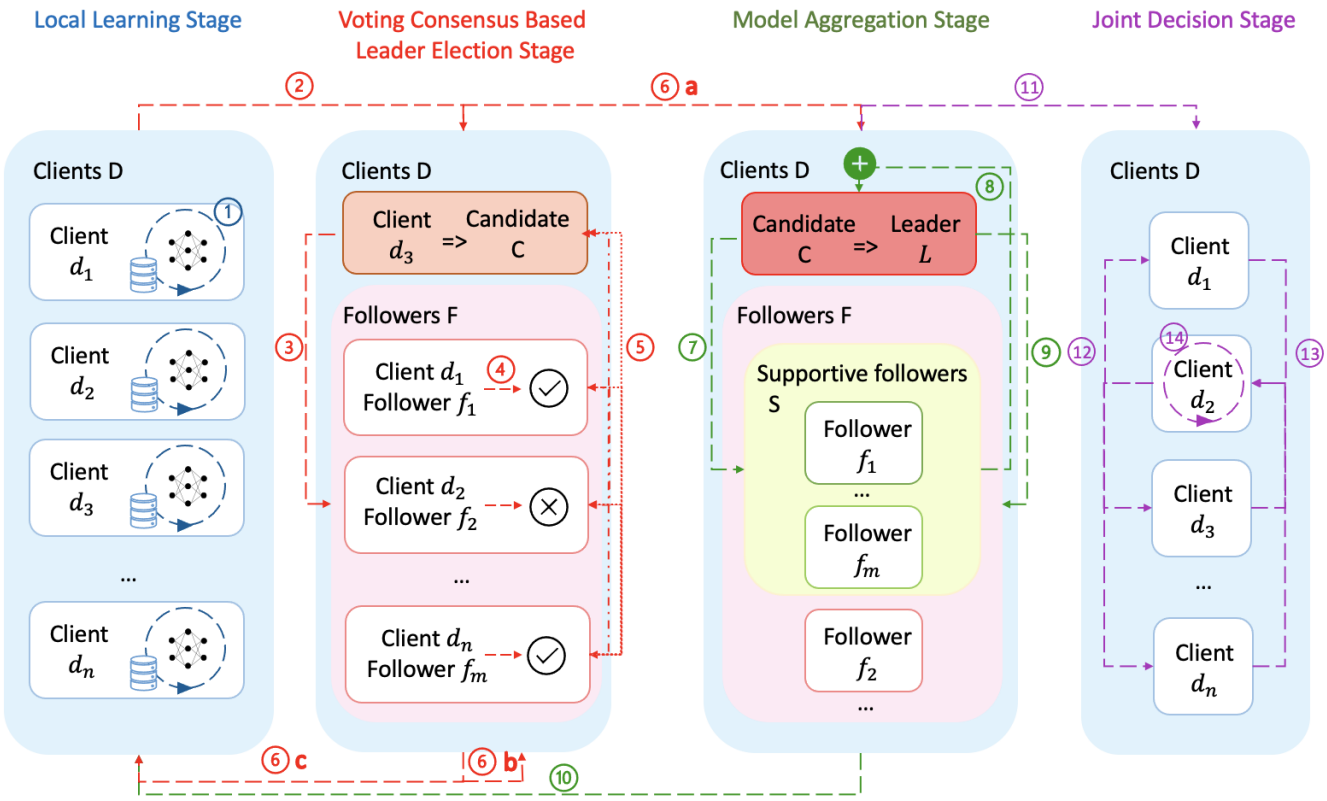

Fig. 1: The diagram of VCDFL: Local Learning Stage: (1) Every client d_i trains their local model with their local dataset; Voting Consensus Based Leader Election Stage: (2) A client becomes the candidate C, the remaining clients become followers f_j; (3) The candidate C broadcasts voting request messages to the followers; (4) Each follower determines whether it reach a consensus with the candidate by comparing the decision information contained in the voting request; (5) Followers who agree with the candidate broadcast the vote; (6a) According to the majority principle, candidate who reach more than $\frac{m}{2}$ followers becomes the leader L; (6b) The candidate fail to becomes the leader, change another client as the candidate and repeat step 3 to step 5; (6c) No candidate is able to become the leader, return to Step 1 for local learning; Model Aggregation Stage: (7) Leader send the model collection requests to the supportive followers; (8) Supportive followers send their local models to the leader for model aggregation; (9) After model aggregation, leader L broadcast the global model; (10) Model update, begin the next iteration; Joint Decision stage: (11) Stop model transmission and aggregation; (12) Broadcast joint decision requests; (13) Return decision to the requester; (14) Replace individual decision with joint decision

example, in an image recognition scenario, the model is used to distinguish images. Therefore, the decisions can be the labels or logics, which are represented in a matrix or a serial number.

B. Voting consensus based leader election stage

As present in Figure 1 and algorithm 1, every client in the system can switch among three roles: Leader, Candidate and Follower. At the beginning of the voting consensus based leader election stage, we set up a random client as the candidate to contest the election. Let the number of all clients in the network be n. D represents the set of clients in the system. r represents the local decision. After one client becomes the candidate, the remaining $m = n - 1$ clients will be followers. Therefore, we have $D = (d_1, d_2, d_3, ...d_n) = C \cup F$, where $d_i, 1 \leq i \leq n$ represents the i-th client, C means the candidate and $F = (f_1, f_2, ...f_m)$ is the set of followers. Since the candidate is chosen at random, it needs to get enough votes to become the leader. First, the candidate broadcasts voting request messages that carry its own local decision r_C. Then, after each follower f_i receives the voting request message, it calculates the difference between its decision r_{f_i} and the candidate's decision r_C, and decides whether to vote for the current candidate by comparing the difference $|r_C - r_{f_i}|$ with the margin of error e. The margin of error e is a value the follower sets to judge whether a candidate meets its voting criteria. Since the candidate does not know and cannot interfere with the value of e, it cannot control the election's result. If the difference $|r_C - r_{f_i}|$ is smaller than the margin of error e, we call that the candidate C and the follower f_i have reached a consensus, and then f_i votes for C to support it as the leader. On the contrary, if $|r_C - r_{f_i}| > e$, this situation is referred as that the candidate C and the follower f_i do not reach a consensus, and then f_i does not vote for C.

After receiving all voting information, the candidate counts the votes. Since our voting system obeys the majority principle, if the number of followers, s, who reach a consensus with the candidate is larger than $\frac{m}{2}$, then it can be called the system

TABLE I: Notations.

Notation	Definition
D	The set of clients
n	The number of clients
F	The set of followers
m	The number of followers ($m = n - 1$)
C	The candidate
L	The leader
S	The set of supportive followers
s	The number of supportive followers
q	The number of local updating rounds
e	The margin of error
x	The number of types of non-IID data owned by each client
r	The symbol of decision
W	The symbol of model
R	The set of decisions collected by the requester
P_{CFC}	The probability of the candidate reach a consensus with followers
P_{SC}	The probability of the system reaches a consensus
T	The set of rounds
t	The number of rounds
T_{CS}	The set of rounds before leader election
T_{c1}	The set of rounds where the first candidate becomes the leader
T_{c2}	The set of rounds where the second candidate becomes the leader
T_f	The set of rounds where the leader is unsuccessfully elected
T_{CE}	The set of joint decision rounds
O_{CS}, O_{c1}, O_{c2}, O_f, O_{CE}	Communication cost in each corresponding round
O_{all}	All communication overhead
S_m	The size of model
S_r	The size of request
S_v	The size of vote
S_d	The size of decision
V_{ij}	The total number of votes for the j-th voting time of T_i round
a	The accuracy of local decisions
a_{jd}	The accuracy of joint decision

reaches a consensus. Then, the candidate becomes the leader, and the system switches to the Model aggregation stage.

However, if $s < \frac{m}{2}$, it can be considered that the system has not reached a consensus. Hence, the candidate cannot become the leader. Below are three reasons for this situation. **1. It is too early to reach a consensus (q is small).** As the time for local learning is insufficient, the decision results are too scattered, and a consensus cannot be reached within the system. **2. A wrong candidate.** In this case, clients in the system who are eligible to become the leader, that is, can reach a consensus with more than half of the followers, are not selected as the candidate. **3. e is too small.** If e is too small, similar decisions cannot be regarded as a consensus. Therefore, if the first candidate fails to become the leader, we will repeat the leader election stage and choose another client to become the candidate. Given the time and communication costs involved in the leader election stage and the failure may be due to insufficient local training and premature consensus, we will only repeat the election once in this paper. Hence, if the second candidate still does not receive enough votes, we will return to the local learning stage and enlarge the parameter of the error range e during the next time in the Voting consensus based leader election stage. The e here can be used in follow-up research to design adaptive algorithms for adjusting the value according to individual requirements

Algorithm 1: Voting consensus based leader election stage.

```
Initialization: h = 1
repeat
    Randomly select a client as the candidate C, and
     the others become followers F = (f_1, f_2, ...f_m)
    The candidate C broadcasts voting request
     message carrying r_C
    foreach f_i, 1 ≤ i ≤ m do
        Receive voting request message
        Calculate |r_C − r_{f_i}|
        Compare |r_C − r_{f_i}| with the margin of error e
        if |r_C − r_{f_i}| > e then
            f_i do not reach a consensus with C
        else
            f_i reach a consensus with C
            Vote for C
        end
    end
    Candidate C count the votes
    if Vote number s > m/2 then
        The system reaches a consensus
        Candidate C becomes the Leader L
        Go to the Model aggregation Stage
    else
        The system does not reach a consensus
        t ← h + 1
    end
until h = 3;
Enlarge e
Go to the Local learning stage
```

and selecting the most suitable leader. However, this is not the main research direction of this paper, so it is uniformly represented by e in this paper, and the relationship between e and the system consensus probability is provided in Section IV.

C. Model aggregation stage

In the Voting consensus based leader election stage, only when their own decision and the candidate's decision reach a consensus will they vote for the candidate. Therefore, while the leader is elected, the followers are also screened. Hence, as shown in algorithm 2, in the Model aggregation stage, the leader only sends model collection requests to the followers ($f_i \in S$) who have voted for it, where S is the set of supportive followers who reach a consensus with the leader and s is the number of those followers. After followers receive the request, they send their local model W_i, back to the leader. Then, the leader aggregates the received models by the weighted average method to calculate a global model:

$$W_g = \frac{\sum_{f_i \in S} W_i + W_L}{s + 1}. \tag{1}$$

In this way, by aggregating only the models of clients whose decisions reach a consensus, outliers will not interfere with the global model, which not only greatly reduces the transmission

Algorithm 2: Model aggregation stage.

The leader L sends the model collection request to $f_i \in S$
foreach $f_i \in S$ **do**
  Receive the request from L
  Send local model W_i to L
end
L aggregates the received models $W_g = \frac{\sum_{f_i \in S} W_i + W_L}{s+1}$
L evaluates the loss value of W_g and judge whether the FL has converged
if *FL has converged* **then**
  Stop model transmission and aggregation
  Go to the Joint decision stage
else
  L sends W_g to the followers
  Model updates
  Go to the Local learning stage
end

of models but also accelerates the convergence speed of the system. In addition, after FL converges, the accuracy of the model will not be improved much during model aggregation. On the contrary, user privacy is at risk of being leaked during model transmission. Therefore, after the leader obtains the global model, it uses part of its local dataset as the test dataset to evaluate the loss value of the global model. If the loss value is low to a certain extent, it can be judged that FL has converged, where the model aggregation is stopped and the system switches to the Joint decision stage. If the model has not converged yet, the leader sends the global model to the followers. Then, the model is updated by both the leader and followers. After the model updates, the system returns to the Local learning stage and starts the next iteration.

D. Joint decision stage

In this stage, we begin decision exchange and decision consensus building instead of model exchange. In the application scenario, a client d_i who needs to use the model for decision-making broadcasts a joint decision request. Clients who receive the request use the local model to make a responding decision about this information and send its decision back to the requester. Then, the decision which is agreed upon by the largest number of clients in the system is referred to as a joint decision:

$$r_{d_i} = r_{joint} = argmax_{r_i} P(r_i : r_i \in R), \tag{2}$$

where R is the set of decisions collected by the requester and $P(r_i : r_i \in R)$ represents the probability of the decision value of r_i in R. Therefore, the fault tolerance rate of the system and the accuracy of the joint decision can be improved in this stage.

III. Analysis of the framework

This section presents the analysis of the voting probability and the system consensus probability in the Voting consensus based leader election stage. The communication efficiency and joint decision accuracy of VCDFL are also proposed.

A. The voting probability and the system consensus probability in Voting consensus based leader election stage

In order to calculate the probability of system consensus. First, we need to calculate the probability P_{CFC} that the followers and the candidate reach a consensus, which is also the probability that followers will vote :

$$\begin{aligned} P_{CFC} =& P(|r_C - r_{f_i}| < e) \\ =& P(r_C - r_{f_i} < e, r_C - r_{f_i} > 0) + \\ & P(r_{f_i} - r_C < e, r_C - r_{f_i} < 0). \end{aligned} \tag{3}$$

Assuming probabilities of all local decisions are IID and continuous probability distribution, we consider three common distributions as examples: a Uniform distribution, an Exponential distribution and a Gaussian distribution. The following are the corresponding formulas:

$$P_{CFC} = \begin{cases} \dfrac{-e^2 - 2ae + 2be}{(b-a)^2}, \\ \text{with a Uniform distribution } r_C, r_{f_i} \sim U(a,b) \\ 1 - exp(-\lambda e), \\ \text{with an Exponential distribution } r_C, r_{f_i} \sim E(\lambda) \\ erf(\dfrac{e}{\sqrt{2}\sigma}), \\ \text{with a Gaussian distribution } r_C, r_{f_i} \sim N(\mu, \sigma^2) \end{cases} \tag{4}$$

The proof of Equation 4 is given in Appendix, and the simulation results are shown in Figure 2. The system consensus is only reached when the candidate reaches a consensus with more than half of the clients in the system. Therefore, the probability of the system reaching a consensus P_{SC} is:

$$P_{SC} = \sum_{k=\frac{m}{2}}^{m} C_m^k P_{CFC}^k (1 - P_{CFC})^{m-k}, \tag{5}$$

which is similar to the consensus reliability of raft when the links are assumed reliable [26]. The simulation results are shown in Figure 3.

B. Communication efficiency

Communication costs are always a vital index in decentralized learning. In order to evaluate the communication overhead of our method, we divide the whole procedure into five types according to the information transfer process. T represents the set of all t rounds. T_{CS}, T_{c1}, T_{c2}, T_f and T_{CE} are the sets of rounds of five different communication processes where the meaning are shown in Table I. $T = T_{CS} \cup T_{c1} \cup T_{c2} \cup T_f \cup T_{CE}$. Besides, $t_{CS}, t_{c1}, t_{c2}, t_f, t_{CE}$ represent the number of rounds in each kind so that $t = t_{CS} + t_{c1} + t_{c2} + t_f + t_{CE}$, where $t_{CS}, t_{c1}, t_{c2}, t_f, t_{CE} \in N$. Figure 4 shows the communication procedures in different situations.

When $T_i \in T_{CS}$, in rounds without consensus voting, clients use local data to learn local models, and there is no decision exchange, voting and model communication. Therefore, the communication cost O_{CS} in the round before leader election is:

$$O_{CS} = 0, T_i \in T_{CS}. \tag{6}$$

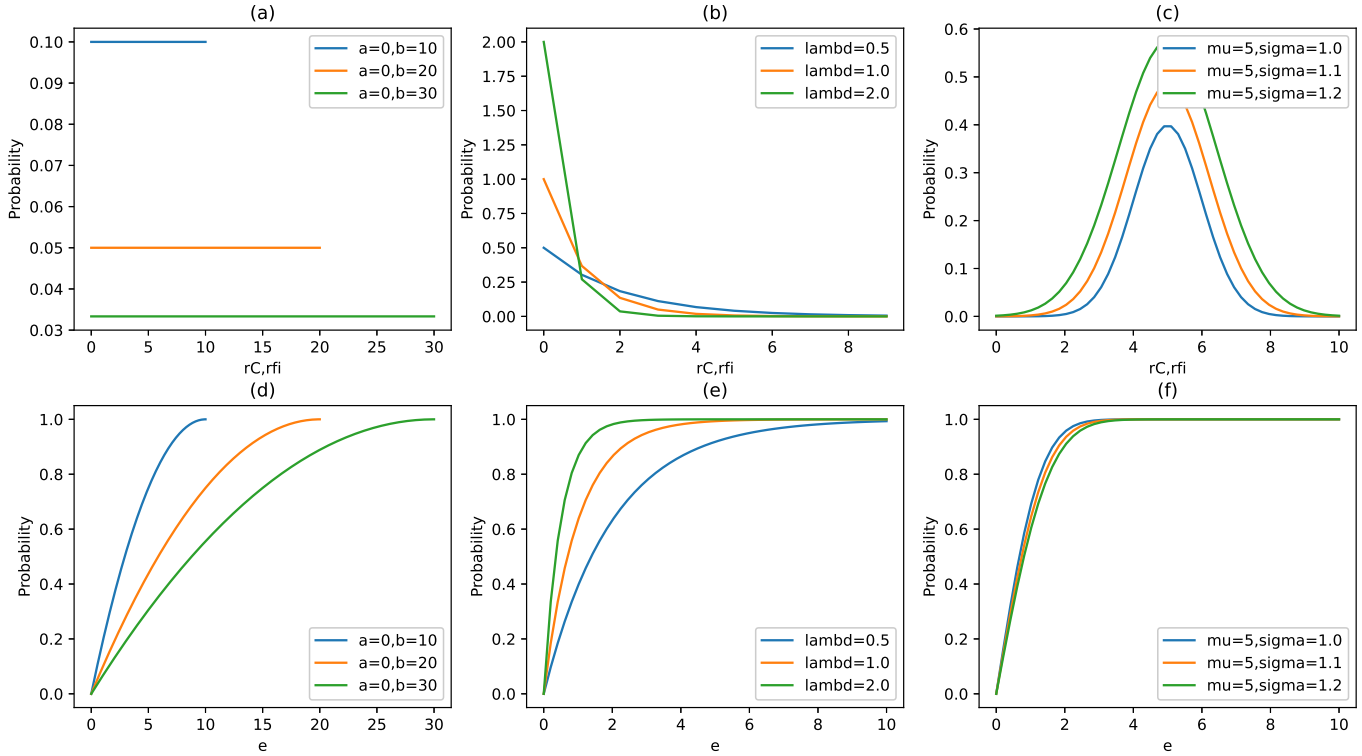

Fig. 2: The probability of the candidate and each follower reaching a consensus P_{CFC}. (a) the probabilities of the candidate's and follower's local decisions satisfy a uniform distribution, and the corresponding P_{CFC} is shown in (d); (b) the probabilities of the candidate's and follower's local decisions satisfy an exponential distribution, and the corresponding P_{CFC} is shown in (e); (c) the probabilities of candidate's and follower's local decisions satisfy a Gaussian distribution, and the corresponding P_{CFC} is shown in (f).

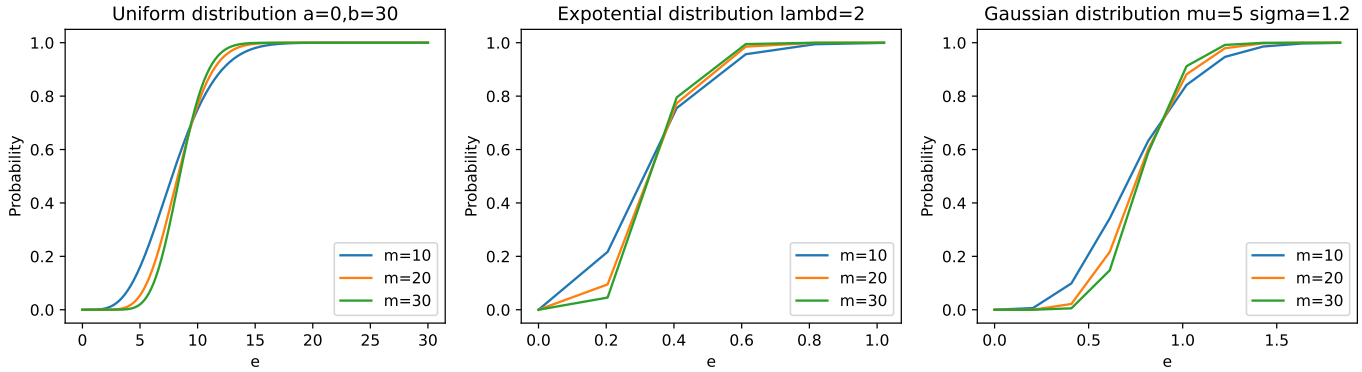

Fig. 3: The system consensus probability P_{SC}.

As shown in Figure 4 (a), when $T_i \in T_{c1}$, the procedure of communication begins with the decision and the voting request from the first candidate. After comparing the candidate's decision with their own, followers who reach a consensus with the candidate will vote for the candidate. Then, because we have the total number of votes in the first voting round, V_{i1}, larger than $\frac{m}{2}$, the system reaches a consensus, and the first candidate becomes to be the leader. The approved leader sends the model collection request to its followers S and then collects the local models from them. After model aggregation by the leader, the global model will return feedback to followers. Hence, the communication overhead O_{c1} in each round when the first candidate becomes the leader is:

$$\begin{aligned} O_{c1} =&(S_d + S_r) \cdot m + V_{i1} \cdot S_v \cdot m + S_r \cdot V_{i1} \\ &+ V_{i1} \cdot S_m + S_m \cdot V_{i1}, T_i \in T_{c1}. \end{aligned} \tag{7}$$

When $T_i \in T_{c2}$, as shown in Figure 4 (b), since the first candidate does not receive more than half of the votes, a second candidate is required. Therefore, there is an additional round of voting processes. Therefore, the communication cost O_{c2} in each round that the second candidate becomes the leader is:

$$\begin{aligned} O_{c2} =&(S_d + S_r) \cdot m + V_{i1} \cdot S_v \cdot m + (S_d + S_r) \cdot m \\ &+ V_{i2} \cdot S_v \cdot m + S_r \cdot V_{i2} + V_{i2} \cdot S_m \\ &+ S_m \cdot V_{i2}, T_i \in T_{c2}. \end{aligned} \tag{8}$$

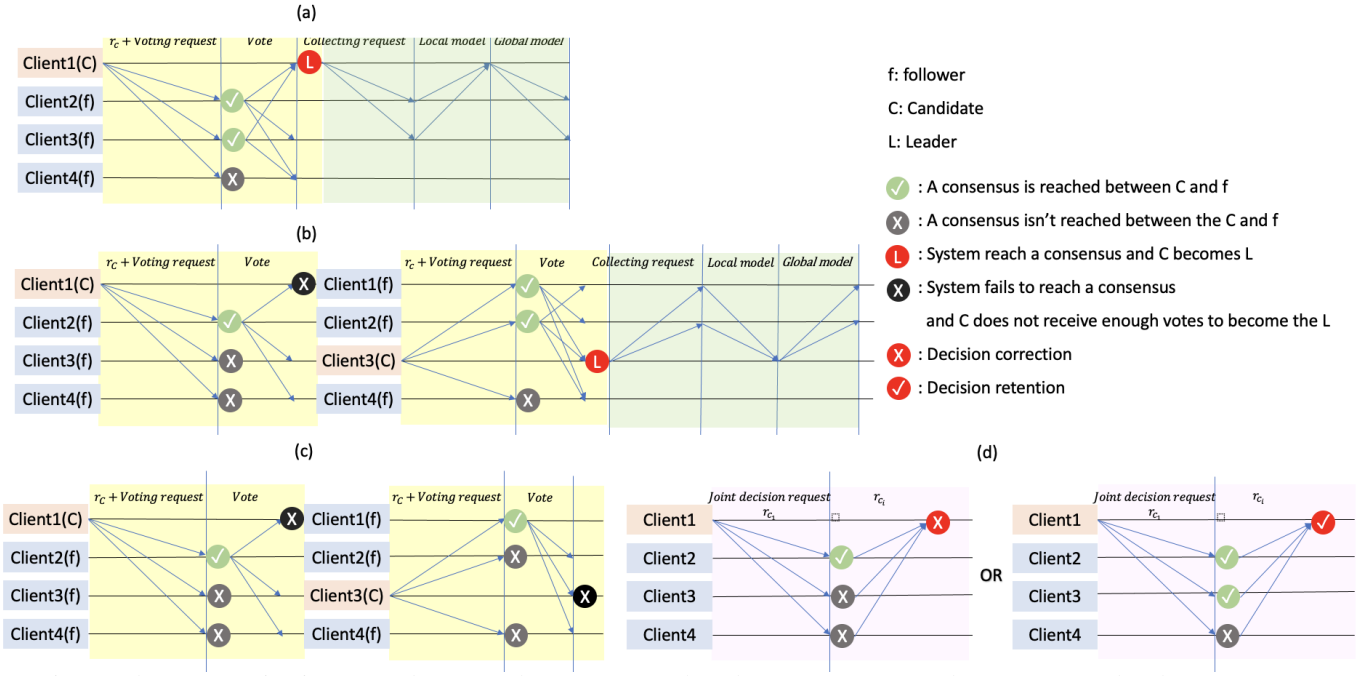

Fig. 4: The communication procedure (a) when $T_i \in T_{c1}$, (b) when $T_i \in T_{c2}$, (c) when $T_i \in T_f$, (d) when $T_i \in T_{CE}$.

In Figure 4 (c), when $T_i \in T_f$, both candidates failed to become the leader. Hence, each round of communication overhead O_f when the leader is not successfully elected is computed as follows:

$$O_f = (S_d + S_r) \cdot m + V_{i1} \cdot S_v \cdot m + (S_d + S_r) \cdot m + V_{i2} \cdot S_v \cdot m, T_i \in T_f. \quad (9)$$

Shown in Figure 4 (d), in joint decision rounds, when $T_i \in T_{CE}$, only decisions are transmitted between clients. Therefore, the communication overhead O_{CE} in each joint decision round is:

$$O_{CE} = S_r \cdot m + S_d \cdot m, T_i \in T_{CE}. \quad (10)$$

To conclude, the communication overhead of our method O_{all} is computed as:

$$O_{all} = O_{CS} \cdot t_{CS} + O_{c1} \cdot t_{c1} + O_{c2} \cdot t_{c2} + O_f \cdot t_f + O_{CE} \cdot t_{CE}. \quad (11)$$

In our system, model transmission only happens in the Model aggregation stage. In the Voting consensus base leader election stage and Joint decision stage, we transmit decisions for information exchange. Since the size of decisions is only a few bytes and the number of rounds of model transfer is also reduced compared with traditional FL, our method saves communication overhead to a large extent.

C. Joint decision accuracy

Even if the joint decision is a local decision agreed upon by the largest number of clients in the system, there is still a possibility that the joint decision is wrong. Therefore, the question we discuss below is whether the joint decision can improve accuracy. In some application scenarios like image recognition or automatic driving, the decision can be the label of the picture or vehicle control decisions like turning left or stopping whose distribution is discrete, and only when the decisions are the same ($e = 0$) can both clients reach a consensus. Hence, the accuracy of the joint decision is related to the accuracy of the clients' local decisions. Since the system has converged in the joint decision stage, the decisions are highly similar, so we assume that all local decisions are collected, the joint decision is agreed by at least $\frac{n}{2}$ clients, and the accuracies of local decisions are the same, which is represented by a. Therefore, the accuracy of joint decision a_{jd} is:

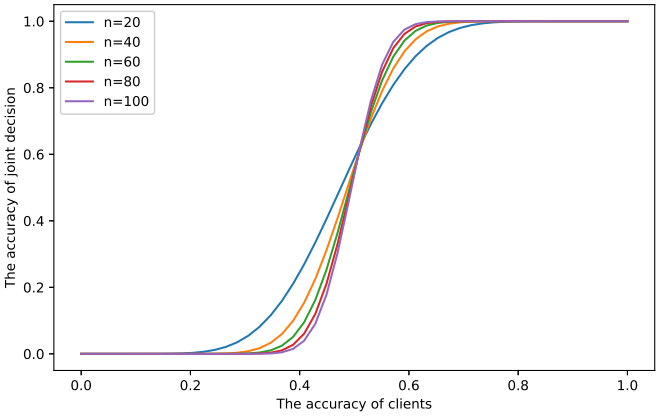

Fig. 5: The relationship between joint decision accuracy and individual accuracy when the number of clients n is 20, 40, 60, 80, 100.

$$a_{jd} = \sum_{k=\frac{n}{2}}^{n} C_n^k a^k (1-a)^{n-k}. \quad (12)$$

In Figure 5, we choose the number of clients in the system as $n = 20, 40, 60, 80, 100$ to show the result. We can see that when the accuracy of clients a is larger than 0.5, the joint decision mechanism has a positive effect on the system, which improves the accuracy of the system. In this way, even if the

training results of the client devices are not very good and the decision-making accuracies are not high, the system accuracy can still be greatly improved with very little communication overhead through the joint decision mechanism. The more clients in the system, the better the improvement of joint decision-making on the accuracy and the more difficult it is to unify the decisions. When $n = 20$ and $a = 0.8$, the accuracy of the joint decision is about 0.98. When $n = 100$, and even though the local decision accuracy is low (i.e., $a = 0.6$), the accuracy of the joint decision is larger than 0.99.

IV. Performance Evaluation

A. Evaluation setup

To simulate the performance of our method, we apply our method in an image recognition scenario. We use the standard MNIST dataset[1], which consists of 60000 images for training and 10000 for testing. We also randomly selected some images to serve as a public database without labels for clients to make decisions based on their models. All images are handwritten digits from zero to nine. We set 100 clients in the system and assume all clients are benign and honest. The artificial neural network includes two hidden layers with 200 units using ReLu activation. We study both IID and non-IID sampling of the local training data. In the IID setting, all data is shuffled and divided into 100 clients. Each client has data in all kinds of labels. In the non-IID setting, we allocate x types of data for each client ($1 \leq x \leq 10, x \in Z$). Hence, all training data is first sorted by labels, and the data of each label is divided into $10x$ shards without overlapping.

B. Performance using IID data

Decision similarity. Since whether the model can be aggregated is related to the difference between local decisions, we analyze the client's decision similarity with the help of box plots, as shown in Figure 6. In each box plot, the circles represent outliers, the upper and lower horizontal lines correspond to the maximum and minimum values, and the upper and lower edges of the box correspond to the Q3 value (75th percentile) and Q1 value (25th percentile). In addition to this, we show the average as a green triangle and the median as an orange line. The first box plot on the left shows the decision similarities of clients after only 5 epochs of local learning. Without any model exchange or aggregation, in the IID data situation, the local decisions are at least 75% similar so that systematic consensus can be reached in the first round. The remaining three boxes show, from left to right, the similarity of local decisions within the system after one, five, and ten rounds of the model aggregation stage. It can be seen that with the information exchange through model aggregation, the mean similarity is gradually improved from 75% to about 92.5%. And the system can reach a consensus in every round.

Joint decision effect. Although, in the framework design, the joint decision mechanism will only run after the FL converges. Due to different convergence standards, in order to demonstrate the effect of joint decision-making, this mechanism starts from the first round of simulation. In Figure 7, we compare the accuracy of joint decisions with the accuracy of the global model and local models, where the joint decision mechanism is initiated by the leader in each round. Besides, the maximum accuracy in each kind is shown in Table II. It can be seen that the accuracy of the joint decisions is higher than one of the global models and much higher than any other local models. The highest joint decision accuracy is 0.9885. The reason is that on the basis of obtaining the global model, the joint decision is the result of local decisions filtered again based on the majority principle, thereby promoting the accuracy of system decision-making and improving local decision-making to some extent.

[1]http://yann.lecun.com/exdb/mnist/

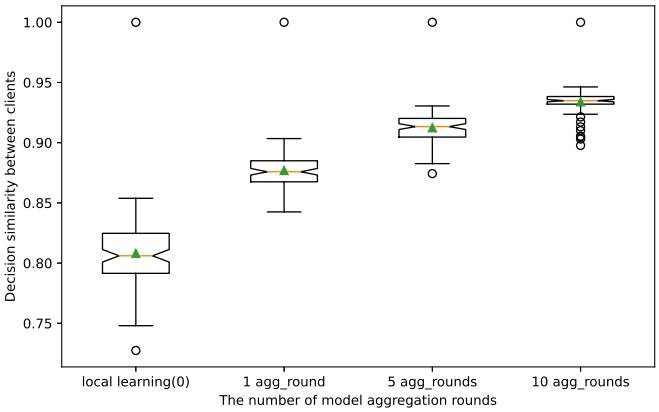

Fig. 6: The similarity of decisions when each client uses IID data.

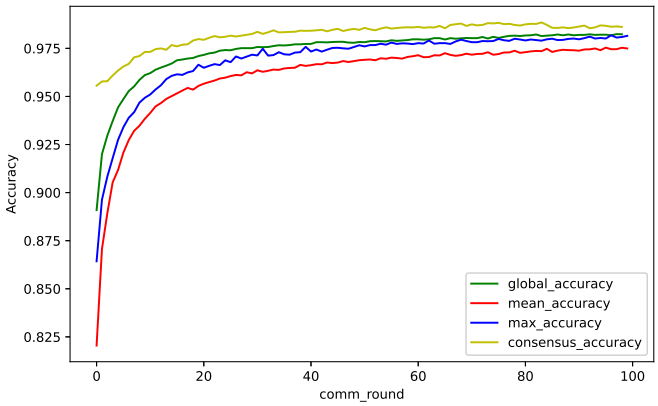

Fig. 7: The joint decision effect for IID data in 100 communication rounds.

TABLE II: Maximum accuracy with IID and non-IID data.

Category	Value			
	IID	x=7	x=8	x=9
Global accuracy	0.9824	0.9793	0.9797	0.9796
Mean local accuracy	0.9754	0.9158	0.9289	0.9426
Max local accuracy	0.9818	0.9741	0.9766	0.9758
Joint decision (consensus) accuracy	**0.9885**	**0.9918**	**0.9913**	**0.9894**

Communication cost. Based on Formulas (6)-(11) and the simulation result when clients use IID data, we calculate the communication cost of our algorithm and compare it with the communication overhead of the traditional FL, as shown in Figure 8, to demonstrate the advantage of our algorithm in

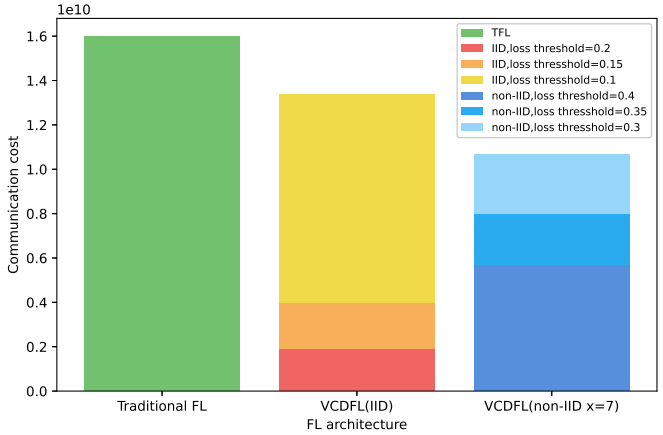

Fig. 8: The communication cost comparison between traditional FL and VCDFL when each client uses IID and non-IID data.

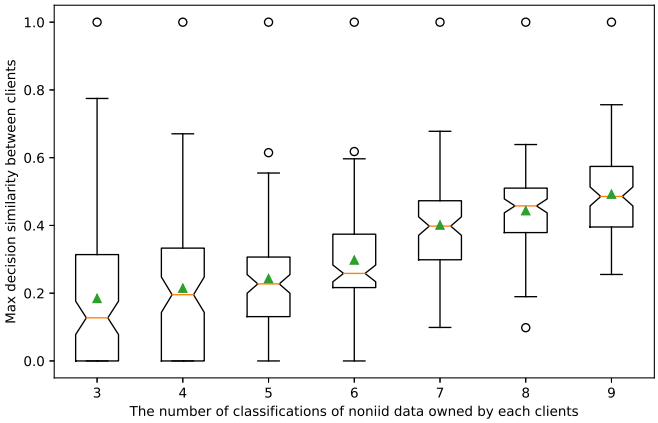

Fig. 9: The max similarities of local decisions in 100 rounds of local training when each client has $x = 3$ to $x = 9$ types of local training data.

saving communication resources. According to the simulation parameter set above, the size of each model is about 8×10^5 bytes. Hence, if all clients in traditional FL are involved in model transmission in 100 rounds, the communication cost is about 1.6×10^{10} bytes, which is shown as the green bar in Figure 8 and named as traditional FL. In our network, we set the size of voting requests, votes, decisions and collecting requests as $S_r = S_v = S_d = 8$ bytes, which are negligible compared with the model size. When using IID data, the decision similarities of clients are high enough to make the system continuously reach a consensus after the first communication round. Moreover, nearly all followers in the system can reach a consensus with the first candidate in each round. Therefore, $t_{CS} = t_{c2} = t_f = 0$ and $V_{i1} = m = 99$. Since the magnitude of t_{CE} is determined by when the system converges, we set the loss threshold to judge whether the system converges. In other words, if the loss of the global model is less than the threshold, the system converges and enters the Joint Decision Stage. If the loss threshold is 0.2, the system converges after 12 rounds, and we have $t_{c1} = 12$ and $t_{CE} = 88$. Therefore, we have $O_{all} = O_{c1} \cdot t_{c1} + O_{CE} \cdot t_{CE} \approx 1.92 \times 10^9$, which is the red bar in the Figure 8 that saves 88% communication overhead compared with the centralized FL method. If the system convergences when the global model loss is less than 0.15, and we have $t_{c1} = 25, t_{CE} = 75$ and $O_{all} = O_{c1} \cdot t_{c1} + O_{CE} \cdot t_{CE} \approx 4 \times 10^9$ that represent as the orange bar in Figure 8 which saves about 75% of communication cost compared with the centralized FL method. If the threshold is 0.1, then we have $t_{c1} = 84$ and $t_{CE} = 16$. The communication cost is $O_{all} \approx 1.34 \times 10^{10}$, which is shown as the yellow bar in Figure 8. Our system saves about 16.25% communication cost compared to the centralised learning method.

C. Performance using non-IID data

Decision similarity. In Non-IID simulation, the distribution of local data has strong inference on the decision similarity of clients, which is closely linked to the probability of the system reaching a consensus. Therefore, as with the IID experiments above, we still use box plots to show the maximum similarities of local decisions in 100 rounds of local training when each client has $x = 3$ to $x = 9$ types of local training data. It can be seen from the median similarity represented by the orange line in Figure 9 that with the increase of data types owned by individuals, the similarity of data increases, and the median similarity of individual decisions also increases from about 10% when $x = 3$ to about 50% when $x = 9$.

Joint decision effect. In Figure 10, we evaluate the joint decision effect of our method by comparing the accuracy of joint decisions with the accuracy of the global model and the max and mean accuracy of local models. As in the IID simulation above, to analyse the effect of joint decision-making, the joint decision mechanism is initiated by the leader in each round. There are three simulations with $x = 7, 8, 9$ kinds of non-IID data owned by clients for local learning. To visualize the results better, we zoom in on the performance from round 40 to 100. Therefore, as can be seen from Figure 10a and Figure 10b, when the client has 7 or 8 categories of non-IID data, the circumstance of the system failing to select a leader is mostly in the first twenty rounds. During that time, the uncertainty of the candidate success to be the leader and the low individual accuracy in the initial stage leads to great fluctuations in the accuracy of joint decisions. Then, with the increase of local learning and model aggregation, the system becomes more stable in reaching a consensus, and the accuracy of joint decisions increases rapidly, which is even higher than that of the global model. The reason is that in the Joint decision stage, minority clients whose decisions are wrong can be corrected by obeying the majority rule. In this way, we can improve the accuracy of decisions and individual fault tolerance rate by paying a very small amount of communication overhead for decision transmission. Therefore, as shown in Table II, the maximum joint decision accuracy is 0.9918, 0.9913 and 0.9894 when $x = 7, 8, 9$. By using the joint decision mechanism, the joint decision accuracy is higher than both global accuracy and max local accuracy. In addition, it can be seen in Figure 9 and Figure 10 that the higher the category similarity of individual data is, the higher the possibility of the system reaching a consensus.

Communication cost. As with the communication cost calculation above in the case of IID data, we calculate the

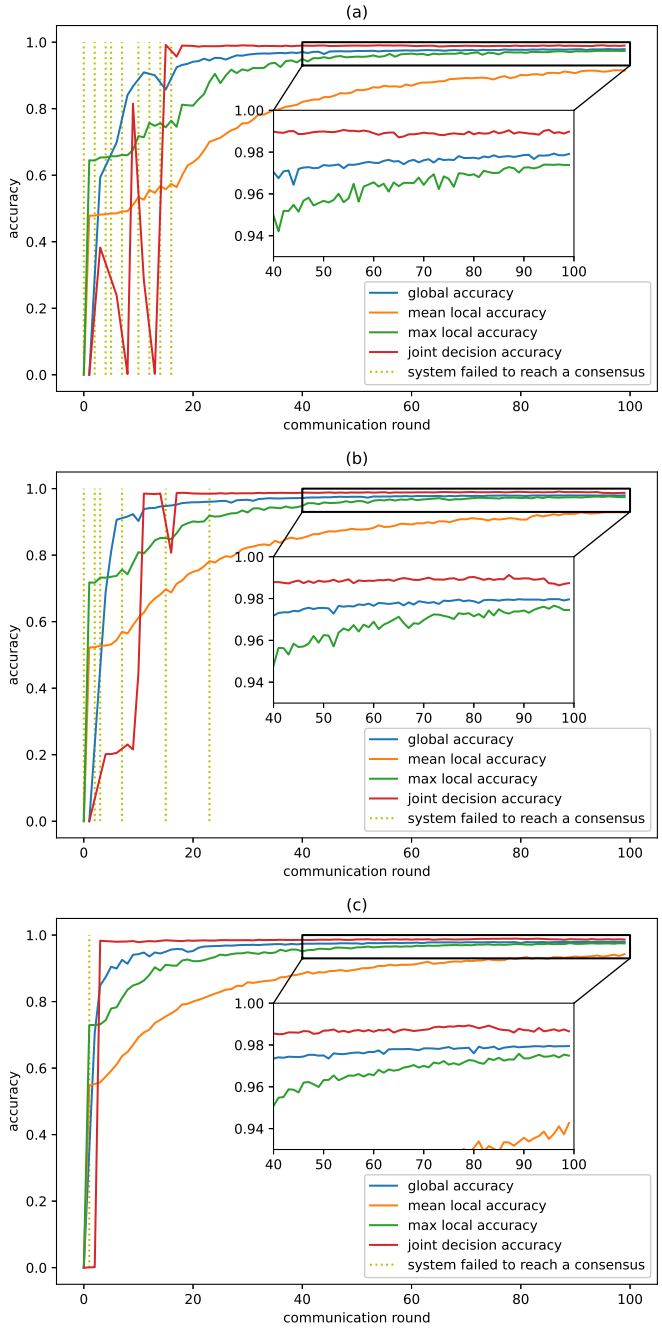

Fig. 10: The joint decision effect for non-IID data in 100 communication rounds. (a) The comparison between the global model accuracy, the max local model accuracy, the mean local model accuracy and the joint decision accuracy when $x = 7$; (b) The accuracy comparison when $x = 8$; (c) The accuracy comparison when $x = 9$.

communication cost of our algorithm by Equations 6-11 and compare it with the 1.6×10^{10} bytes of traditional FL communication overhead, which is shown as the green bar in Figure 8. As shown in Figure 10, it is not easy for the system to reach a consensus at the beginning of the Non-IID simulation. Moreover, since the similarity of local decisions in the early stage is not very high, even if the system reaches a consensus, it may be reached in the second round of voting when the first candidate fails. Therefore, according to the non-IID simulation result in Figure 10a, when we set $x = 7$ and $m = 99$, we have $t_{CS} = 0$ and $t_f = 9$. If loss threshold is 0.4, the system converges after 58 rounds, we have $t_{c1} = 48, t_{c2} = 10, t_{CE} = 42$. Hence, the communication overhead is $O_{all} = O_f \cdot t_f + O_{c1} \cdot t_{c1} + O_{c2} \cdot t_{c2} + O_{CE} \cdot t_{CE} \approx 5.7 \times 10^9$ bytes, shown as the dark blue bar in Figure 8, which saves about 64.4% of the communication overhead comparing with centralized FL. If the loss threshold is 0.35, the system will achieve convergence in about 75 rounds. At that time, the communication overhead is 8×10^9 bytes, represented as the blue bar in Figure 8, saving about 50% of the communication overhead compared to the centralized FL. Moreover, if the system converges when the loss value of the global model is less than 0.3, the convergence will be achieved in about 93 rounds. At this time, the communication overhead is 1.07×10^{10} bytes, which saves about 33% of the communication overhead compared with the centralized FL method, shown as the light blue bar in Figure 8.

V. Conclusion

This paper proposes a voting consensus based decentralized federated learning. The consensus discrimination mechanism between the candidate and followers based on decision communication and the voting mechanism in the system ensures that the local model does not leave the local device when the candidate is an outlier node and eliminates the interference of the outlier models on the model aggregation. The consensus probability, communication efficiency and joint decision accuracy have been analyzed. In addition, we simulate our method in an image recognition scenario with IID and non-IID partition of data. The simulation results are verified through the design of a joint decision mechanism, and the communication overhead is greatly reduced while the system accuracy is improved.

References

[1] S. Pandya, G. Srivastava, R. Jhaveri, M. R. Babu, S. Bhattacharya, P. K. R. Maddikunta, S. Mastorakis, M. J. Piran, and T. R. Gadekallu, "Federated learning for smart cities: A comprehensive survey," *Sustainable Energy Technologies and Assessments*, vol. 55, p. 102987, 2023.

[2] S. Kuutti, R. Bowden, Y. Jin, P. Barber, and S. Fallah, "A survey of deep learning applications to autonomous vehicle control," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 2, pp. 712–733, 2021.

[3] J. N. Njoku, C. I. Nwakanma, G. C. Amaizu, and D.-S. Kim, "Prospects and challenges of metaverse application in data-driven intelligent transportation systems," *IET Intelligent Transport Systems*, vol. 17, no. 1, pp. 1–21, 2023.

[4] P. Nirmala, S. Ramesh, M. Tamilselvi, G. Ramkumar, and G. Anitha, "An artificial intelligence enabled smart industrial automation system based on internet of things assistance," in *2022 International Conference on Advances in Computing, Communication and Applied Informatics (ACCAI)*. IEEE, 2022, pp. 1–6.

[5] S. Yang, F. Zhu, X. Ling, Q. Liu, and P. Zhao, "Intelligent health care: Applications of deep learning in computational medicine," *Frontiers in Genetics*, vol. 12, p. 444, 2021.

[6] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE Communications Surveys Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[7] J. Hojlo, "Future of industry ecosystems: Shared data and insights." [Online]. Available: https://blogs.idc.com/2021/01/06/future-of-industry-ecosystems-shared-data-and-insights/

[8] T. C. Aysal, M. E. Yildiz, A. D. Sarwate, and A. Scaglione, "Broadcast gossip algorithms for consensus," *IEEE Transactions on Signal processing*, vol. 57, no. 7, pp. 2748–2761, 2009.

[9] J. Daily, A. Vishnu, C. Siegel, T. Warfel, and V. Amatya, "Gossipgrad: Scalable deep learning using gossip communication based asynchronous gradient descent," *CoRR*, vol. abs/1803.05880, 2018.

[10] Y. Deng, F. Lyu, J. Ren, Y. Zhang, Y. Zhou, Y. Zhang, and Y. Yang, "Share: Shaping data distribution at edge for communication-efficient hierarchical federated learning," in *2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*, 2021, pp. 24–34.

[11] Y. Gou, R. Wang, Z. Li, M. A. Imran, and L. Zhang, "Clustered hierarchical distributed federated learning," in *ICC 2022-IEEE International Conference on Communications*. IEEE, 2022, pp. 177–182.

[12] H. Yang, K.-Y. Lam, L. Xiao, Z. Xiong, H. Hu, D. Niyato, and H. Vincent Poor, "Lead federated neuromorphic learning for wireless edge artificial intelligence," *Nature communications*, vol. 13, no. 1, p. 4269, 2022.

[13] M. R. Behera, S. Shetty, R. Otter *et al.*, "Federated learning using peer-to-peer network for decentralized orchestration of model weights," 2021.

[14] D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm," in *2014 USENIX Annual Technical Conference (Usenix ATC 14)*, 2014, pp. 305–319.

[15] G. V. Crosby, N. Pissinou, and J. Gadze, "A framework for trust-based cluster head election in wireless sensor networks," in *Second IEEE Workshop on Dependability and Security in Sensor Networks and Systems*. IEEE, 2006, pp. 10–pp.

[16] X. Cao, M. Fang, J. Liu, and N. Z. Gong, "Fltrust: Byzantine-robust federated learning via trust bootstrapping," *CoRR*, vol. abs/2012.13995, 2020.

[17] F. Sattler, K.-R. Müller, T. Wiegand, and W. Samek, "On the byzantine robustness of clustered federated learning," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 8861–8865.

[18] X. Feng and L. Chen, "Data privacy protection sharing strategy based on consortium blockchain and federated learning," in *2022 International Conference on Artificial Intelligence and Computer Information Technology (AICIT)*. IEEE, 2022, pp. 1–4.

[19] H. Mozaffari, V. Shejwalkar, and A. Houmansadr, "Every vote counts: Ranking-based training of federated learning to resist poisoning attacks," in *32nd USENIX Security Symposium (USENIX Security 23)*, 2023.

[20] C. A. Lee, K. Chow, H. A. Chan, and D. P.-K. Lun, "Decentralized governance and artificial intelligence policy with blockchain-based voting in federated learning," *Frontiers in Research Metrics and Analytics*, vol. 8, p. 1035123, 2023.

[21] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[22] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *CoRR*, vol. abs/1610.05492, 2016.

[23] S. Ji, W. Jiang, A. Walid, and X. Li, "Dynamic sampling and selective masking for communication-efficient federated learning," *CoRR*, vol. abs/2003.09603, 2020.

[24] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, 2015, pp. 1322–1333.

[25] Z. Wang, M. Song, Z. Zhang, Y. Song, Q. Wang, and H. Qi, "Beyond inferring class representatives: User-level privacy leakage from federated learning," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, 2019, pp. 2512–2520.

[26] H. Xu, Y. Fan, W. Li, and L. Zhang, "Wireless distributed consensus for connected autonomous systems," *IEEE Internet of Things Journal*, vol. 10, pp. 7786–7799, 2022.

APPENDIX A
PROOF OF EQUATION 4

Considering the probabilities of the leader's and follower's local decisions satisfy a uniform distribution: $r_L, r_{f_i} \sim U(a,b)$, we have

$$P(r_L, r_{f_i}) = \frac{1}{(b-a)^2}, a \le r_{f_i}, r_L \le b. \tag{13}$$

Applying eq. 1, we have

$$\begin{aligned} P_{CFC} =& 2P(r_{f_i} < e + r_L, r_L < r_{f_i}) \\ =& 2[\int_a^{a+e} \int_a^{r_{f_i}} \frac{1}{(b-a)^2}\, dx\, dy \\ &+ \int_{a+e}^{b} \int_{r_{f_i}}^{r_{f_i}-e} \frac{1}{(b-a)^2}\, dx\, dy] \\ =& \frac{-e^2 - 2ae + 2be}{(b-a)^2}. \end{aligned} \tag{14}$$

Considering the probabilities of leader's and follower's local decisions satisfy an exponential distribution: $r_L, r_{f_i} \sim E(\lambda)$, we have

$$P(r_L, r_{f_i}) = \lambda^2 exp(-\lambda(r_L + r_{f_i})), 0 \le r_{f_i}, r_L. \tag{15}$$

Considering eq. 1, we have

$$\begin{aligned} P_{CFC} =& 2P(r_{f_i} < e + r_L, r_L < r_{f_i}) \\ =& 2[\int_0^{e} \int_0^{r_{f_i}} \lambda^2 exp(-\lambda(r_L + r_{f_i}))\, dx\, dy \\ &+ \int_e^{\infty} \int_{r_{f_i}-e}^{r_{f_i}} \lambda^2 exp(-\lambda(r_L + r_{f_i}))\, dx\, dy] \\ =& 1 - exp(-\lambda e). \end{aligned} \tag{16}$$

Considering the probabilities of leader's and follower's local decisions satisfy a Gaussian distribution:$r_L, r_{f_i} \sim N(\mu, \sigma^2)$, we have

$$P(r_L, r_{f_i}) \sim N(\mu_{r_L} - \mu_{r_{f_i}}, \sigma_{r_L}^2 + \sigma_{r_{f_i}}^2) = \sim N(0, 2\sigma^2). \tag{17}$$

Combining eq. 1, we have

$$\begin{aligned} P_{CFC} =& \Phi(\mu_{r_L} - \mu_{r_{f_i}}, \sigma_{r_L}^2 + \sigma_{r_{f_i}}^2; e) \\ &- \Phi(\mu_{r_L} - \mu_{r_{f_i}}, \sigma_{r_L}^2 + \sigma_{r_{f_i}}^2; 0) \\ &+ \Phi(\mu_{r_{f_i}} - \mu_{r_L}, \sigma_{r_L}^2 + \sigma_{r_{f_i}}^2; e) \\ &- \Phi(\mu_{r_{f_i}} - \mu_{r_L}, \sigma_{r_L}^2 + \sigma_{r_{f_i}}^2; 0) \\ =& \frac{1}{2}(1 + erf(\frac{e - (\mu_{r_L} - \mu_{r_{f_i}})}{\sqrt{\sigma_{r_L}^2 + \sigma_{r_{f_i}}^2}})) \\ &- \frac{1}{2}(1 + erf(\frac{e - (\mu_{r_L} - \mu_{r_{f_i}})}{\sqrt{\sigma_{r_L}^2 + \sigma_{r_{f_i}}^2}})) \\ &+ \frac{1}{2}(1 + erf(\frac{e - (\mu_{r_{f_i}} - \mu_{r_r})}{\sqrt{\sigma_{r_L}^2 + \sigma_{r_{f_i}}^2}})) \\ &- \frac{1}{2}(1 + erf(\frac{e - (\mu_{r_{f_i}} - \mu_{r_r})}{\sqrt{\sigma_{r_L}^2 + \sigma_{r_{f_i}}^2}})) \\ =& 1 + erf(\frac{e}{\sqrt{2}\sigma}) - (1 + erf(\frac{0}{\sqrt{2}\sigma})) \\ =& erf(\frac{e}{\sqrt{2}\sigma}). \end{aligned} \tag{18}$$

The "erf" in eq.18 represents the error function in mathematics, which is also called the Gauss error function. The function is defined as:

$$erfz = \frac{2}{\sqrt{\pi}} \int_0^z e^{-t^2} dt \tag{19}$$