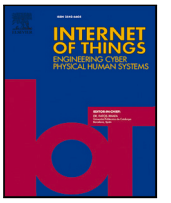
Internet of Things 27 (2024) 101287

Contents lists available at ScienceDirect

Internet of Things

journal homepage: www.elsevier.com/locate/iot



Research article

# TSFed: A three-stage optimization mechanism for secure and efficient federated learning in industrial IoT networks

Made Adi Paramartha Putra [a], Nyoman Bogi Aditya Karna [b], Ahmad Zainudin [c], Dong-Seong Kim [d], Jae-Min Lee [d,*]

[a] Faculty of Information Technology and Design, Primakara University, Denpasar, Bali, Indonesia
[b] School of Electrical Engineering, Telkom University, Bandung 40257, Indonesia
[c] Department of Electronic Engineering, Kumoh National Institute of Technology, Gumi, South Korea
[d] Department of IT Convergence Engineering, Kumoh National Institute of Technology, Gumi, South Korea

ARTICLE INFO

*Keywords:*
Adaptive training
Blockchain
Client selection
Federated learning optimization
Industrial internet of things

ABSTRACT

This paper presents a three-stage optimization mechanism designed to enhance Federated Learning (FL) in Industrial Internet of Things (IIoT) networks. Traditional FL optimizations, which typically focus on a single aspect, fall short in IIoT environments. Our approach integrates a multi-criteria enhancement: first, an Ensembled Client Selection Mechanism (ECSM) selects participants based on accuracy, reputation, and randomness. Second, Adaptive Distributed Client Training (ADCT) dynamically adjusts based on participant performance. Lastly, a Secure and Efficient Communication Channel (SECC), backed by blockchain, meets IIoT's stringent security demands. The evaluation shows TSFed outperforms baseline methods, enhancing FL performance by increasing accuracy and F1-score. Importantly, TSFed improves the efficiency of achieving 80% accuracy on the MNIST dataset by 29.09% over baseline methods, showcasing significant gains in both security and efficiency. This mechanism also exhibits robustness against malicious attacks, setting a new benchmark for FL in IIoT environments.

## 1. Introduction

The significant impact of the massive adoption of Internet of Things (IoT) technology in several fields has been very clear in recent years [1]. For example, the industrial sector can increase efficiency in the production process [2]. The implementation of IoT in the industrial sector is also called Industrial IoT (IIoT), where the devices used in the production process are equipped with IoT, which can provide real-time information about the condition of each device used. To support a robust system, data processing and storage capabilities, secure networks, and integration with Artificial Intelligence (AI) are needed in the IIoT domain [3]. In 2022, a total of 14.4 billion active IoT devices are connected through the Internet, which is expected to be 27 billion connected devices in 2025 [4]. This increase also depends on advances in AI to assist decision-making in various sectors, especially the industrial sector [5]. The implementation of IIoT demands secure communication networks, advanced data processing and storage, and AI algorithms for reliability and efficiency in industrial processes [6]. To fully realize the potential of IIoT, robust communication networks, efficient data processing, and AI-powered data analysis are crucial. Ensuring these prerequisites are in place is vital for the successful implementation of IIoT and the optimization of industrial operations [7,8].

Well-known approach for training AI applications in IIoT and various fields done in a centralized manner. This technique requires data to be stored in a single server before Machine Learning (ML) and Deep Learning (DL) models are trained. The centralized

* Corresponding author.
*E-mail address:* ljmpaul@kumoh.ac.kr (J.-M. Lee).

https://doi.org/10.1016/j.iot.2024.101287
Received 13 May 2024; Received in revised form 23 June 2024; Accepted 9 July 2024
Available online 22 July 2024

learning mechanism can achieve high-performance accuracy due to the vast amount of data used for training. Consequently, the client is required to forward their data and related information to the server. The data gathered on the server poses privacy issues and is vulnerable to client security, especially for sensitive data. In addition, centralized learning leaves the model generalization since the training is performed within a specific dataset. Therefore, the model performance is only generalized for a specific job and may fail to process slightly different tasks [9].

Federated Learning (FL) was developed to address the drawbacks of centralized learning by enabling distributed training without sharing sensitive information with the server [10]. In federated learning, each participant trains the model on their device and then sends the updated model parameters to the server. The server aggregates the parameters from multiple clients to produce a new global model. However, this approach leads to a slight reduction in performance due to the lack of Independence and Identical Distribution (IID) among the different training datasets on each device [11].

Additionally, the conventional FL approach faces difficulties in terms of efficiency. This is because the selection of clients for training is random, leading to uncertainty and an increased possibility of training bias. As a result, the number of rounds required to perform FL increases, which also increases the computational load on clients and the cost of communication, leading to inefficiency. Furthermore, the security of traditional FL is vulnerable to attack because locally updated parameters are sent directly to the server without any protection [12]. Despite these challenges, FL remains the preferred option for a variety of use cases, such as smart automotive systems, healthcare, agriculture, and industries [13].

### *1.1. Motivation and contribution*

Improving FL for specific use cases requires a different approach, as different areas have unique requirements. In the case of IIoT networks, traditional FL models are not enough. The IIoT demands system robustness and efficiency to meet production processes, making efficient FL crucial for its applications. It is important to note that the efficiency of FL can take two forms: (1) the overall FL training time, which includes all communication rounds, and (2) the total number of communication rounds required to reach a specific accuracy, which only calculates the necessary rounds to achieve the desired accuracy. To enhance both model accuracy and FL efficiency, several optimization techniques have been introduced. The client selection mechanism has been improved by considering client resource constraints to speed up model improvement, as proposed in [14]. Other approaches, such as accuracy-based client selection [15,16], classes [17], and parameter similarity [18], has also been proposed to improve the FL process. The client training process has also been explored for improvement through the adaptive adjustment of the dropout layer [19,20], learning rate [21], batch size, and local epoch [22]. Finally, the secure aggregation process in FL has been investigated through the use of differential privacy [23] and blockchain implementation [24].

To the best of our knowledge, most state-of-the-art studies primarily focus on a single optimization aspect, neglecting others. Therefore, the need to provide multi-aspect optimization in FL systems is emerging. To further enhance FL performance, this work addresses three main aspects of the distributed learning approach: client selection, local training, and security. It is designed to meet the requirements of the Industrial Internet of Things (IIoT), which demands high performance and efficiency. The significant contributions of this paper are summarized as follows.

1. We propose a novel Three-Stage Federated Learning (TSFed) optimization mechanism for the IIoT environment that provides a secure and efficient learning process. Our mechanism combines ensembled client selection, adaptive local training, and a secure communication channel for improved model performance and efficiency.
2. We design an Ensembled Client Selection Mechanism (ECSM) for the first stage of TSFed that take into account multi-criteria client selection for FL, such as accuracy, reputation, and randomization. These technique aims to achieve high performance with model generalization under minimal FL rounds.
3. In the second stage, we propose Adaptive Distributed Client Training (ADCT) to conduct adaptive training on the FL clients using the hyperparameter tunning approach.
4. We establish a Secure and Efficient Communication Channel (SECC) to facilitate data exchange between the FL server and clients for the last stage of the TSFed optimization. The SECC is implemented using a private Proof of Authority (PoA) blockchain with a lightweight smart contract design for enhanced security with low complexity.
5. We perform extensive simulations with three well-known datasets and different FL optimization methods to evaluate the performance of our TSFed approach. Our findings demonstrate that TSFed significantly improves FL performance in terms of model accuracy and training efficiency.

The remainder of this article is structured as follows: The relevant studies in FL optimization are summarized in Section 2. The TSFed optimization mechanism proposed in this work is described in detail in Section 3. The results of the performance evaluation and analysis are presented in Section 4. Finally, Section 5 wraps up this paper and highlights potential future research.

## 2. Related work

This article investigated numerous solutions to improve the performance of FL in regard to client selection, local model training process, and security. The results are summarized in Table 1 and detailed as follows.

**Table 1**
Summary of state-of-the-art FL optimization.

| Author | Published year | C1 | C2 | C3 |
|---|---|---|---|---|
| S. AbdulRahman, et al. [25] | 2021 | ■ | □ | □ |
| P. Tian, et al. [18] | 2022 | ■ | □ | □ |
| L. Yu, et al. [26] | 2022 | ■ | □ | □ |
| F. Shi, et al. [27] | 2022 | ■ | □ | □ |
| M.A.P. Putra, et al. [15] | 2022 | ■ | □ | □ |
| Y. Shi, et al. [28] | 2023 | ■ | □ | □ |
| J. Zhang, et al. [29] | 2024 | ■ | □ | □ |
| N. Bouacida, et al. [19] | 2021 | □ | ■ | □ |
| H. Zang, et al. [30] | 2022 | □ | ■ | □ |
| F. Varno, et al. [21] | 2022 | □ | ■ | □ |
| J. Park, et al. [22] | 2022 | □ | ■ | □ |
| A.P. Kalapaaking, et al. [24] | 2022 | □ | □ | ■ |
| Y. Miao, et al. [31] | 2022 | □ | □ | ■ |
| Z. Yang, et al. [32] | 2023 | □ | □ | ■ |
| L. Li, et al. [33] | 2021 | ■ | ■ | □ |
| Ours (TSFed) | – | ■ | ■ | ■ |

Note: C1 represent the client selection optimization, whereas C2 and C3 denotes local training and security improvement, respectively.
■ covered | □ not covered.

### 2.1. Client selection optimization

Numerous approaches have been designed to select appropriate clients for the FL process. The first approach, called Weight Similarity Client Clustering (WSCC), computes the weighted similarity of each participant and groups clients with high similarity into clusters for further training. This approach also considers dataset distributions, as each client's weight is determined by its dataset. The author notes that clustering is dynamically adjusted for each communication round [18]. The second approach, called the Accuracy-based Client Selection (ACS), improves FL performance in the IIoT environment by selecting participants based on a centralized performance evaluation conducted on the FL server rather than randomly. The author states that incorporating ACS with various aggregation strategies leads to improved FL performance results [15]. In addition, optimizing volatile FL (VFedCS) with the client selection mechanism introduced in [27] by quantifying the client's data impact and resource heterogeneity. A resource management-based client selection proposed in [26] dynamically adjusts and optimizes the performance results with the total energy consumption based on the client's processing capability and transmission power. Lastly, multicriteria client selection (FedMCCS) was introduced to consider memory, energy, and processing time [25]. The findings show that FedMCCS outperforms resource-constrained client selection [14] in round completion.

### 2.2. Client training optimization

In terms of improving local clients, various techniques have been introduced to speed up training while also boosting FL performance. An adaptive FL dropout (AFD) approach was proposed to minimize communication costs from the client to the FL server. The suggested AFD, when combined with existing compression methods, significantly decreased convergence time [19]. Next, a dynamic update and adaptive pruning mechanism called FedDUAP was introduced to achieve efficient and high-performance FL. The dynamic update was used to determine the optimal steps for sending the local parameters to the server. At the same time, the pruning method reduced communication costs by pruning multiple layers of the model [30]. Furthermore, the adaptive bias estimation (AdaBest) method was introduced in [21] to accurately estimate the drift across FL clients, which further improved FL performance. Performance evaluations have shown that AdaBest enhances the model performance compared to conventional aggregation strategies. A technique called AMBLE was proposed in [22] to adjust the mini-batch and local epoch. The dynamic configuration of AMBLE also scales the learning rate, which might increase training time for FL. Despite the additional training time, AMBLE has relatively better performance and faster convergence time.

### 2.3. Security optimization

The final consideration for FL optimization is the security of the updated parameters being forwarded to the server. Various methods have been employed to ensure the security of the updated model while preserving client privacy. On the one hand, a verifiable weighted average aggregation is introduced by applying a masking technique to the local parameters and data size that is forwarded to the FL server. This masking mechanism is designed to comply with the aggregation tag, providing efficient verification results and improving FL performance [32]. On the other hand, homomorphic encryption is used along with cosine similarity to detect malicious gradient updates uploaded to the FL server by malicious clients in a privacy-preserving Byzantine-robust FL framework (PBFL) [31]. PBFL utilizes blockchain technology to facilitate a transparent implementation of the regulations. In addition, a blockchain-based FL framework with Intel Software Guard Extension (SGX) has been introduced for the IIoT scenario.

This proposed framework implements a consensus mechanism that verifies each participant's updated hash to be the same [24]. Other blockchain approaches have also been designed to store client metadata [34], reputation [35], and incentive mechanisms [36].

Most of the studies mentioned above focus on a single optimization problem to enhance the FL performance. Nevertheless, multi-criteria optimization can lead to higher FL performance in terms of accuracy and efficiency. Previous work has not adequately explored the combination of FL optimization and its implementation techniques, especially in the IIoT environment. This article proposes TSFed, a multi-criteria optimization mechanism for FL in IIoT. It provides a detailed description of the implementation process and compares its performance with state-of-the-art research.

## 3. Proposed system

In this section, we provide a detailed explanation of the problem formulation and the proposed system, encompassing all optimization mechanisms up to the integration mechanism to create a robust FL system.

### 3.1. Problem formulation

The cross-silo FL approaches reduce communication costs compared to centralized learning in the model updating strategy. FL's distributed manner requires an advanced strategy to ensure the learning process is conducted successfully. There are numerous aspects to enhance the performance of FL, such as client selection, local training, aggregation technique, DL/ML model, and security. Most authors mainly focused on single-aspect optimization for general-purpose FL applications in the current research work detailed in Section 2. Industrial IoT demanded a secure and efficient approach to ensure every process is performed in real-time. More than traditional FL to IIoT requirements is required; thus, a specific purpose optimization is demanded. Based on those critical aspects mentioned before, this article modifies three different processes: client selection, local device training, and security.

#### 3.1.1. Client selection

In the vanilla version of FL, the client who participated in the training process is randomly selected. This approach leads to model uncertainty with higher bias, which is unsuitable for IIoT implementation. Previously, some research focused on client selection technique by utilizing client classes [17], weight similarity [18], and client computing capability [14]. Those approaches successfully enhance the FL performance in terms of training accuracy. It is worth mentioning that the IIoT environment requires a robust client selection mechanism; accordingly, a multi-criteria client selection is the best option to provide better performance improvement by considering multiple aspects of the client characteristic.

#### 3.1.2. Local training

The configuration of the local training process on each device participating in the FL process is another important aspect to enhance the FL performance. Several methods have been proposed to adaptively adjust the mini-batch and local epoch by utilizing the linear scaling approach [22]. The learning rate is another important aspect of model training that needs to be considered to enhance the model's accuracy. This work considers adaptive hyperparameter modification using grid search based on determined accuracy. Model improvement can be achieved within low communication rounds by adopting this approach.

#### 3.1.3. Security

Ensuring model updates are delivered safely to the client and server is required in the IIoT network. Model poisoning attack is able to disturb the training process, resulting in lower model accuracy [37]. As detailed in Section 2, homomorphic encryption and blockchain network is adopted to mitigate this issue. In the IIoT environment, those solutions are acceptable but not adequate. High security requires higher computational resources and additional processing time, which does not apply to IIoT. In this work, we adopt lightweight parameter conversion with efficient smart contract design for the blockchain network to comply with IIoT requirements.

### 3.2. TSFed system model

This paper presents a three-stage optimization for secure and efficient Federated Learning (FL) in the Industrial Internet of Things (IIoT) network, known as TSFed. The architecture of the TSFed system is shown in Fig. 1. $k$, where $k \in \{1, 2, 3, \dots, K\}$, denotes the number of factories that can participate in the FL process among the total of $K$ factories in the IIoT network. Each factory has a local server that is responsible for data collection, model training, and parameter updating. On the other hand, the FL server is responsible for client selection and model aggregation. The TSFed starts with the ECSM stage, which combines multiple criteria to choose the appropriate local server for the training process. The second stage involves the training process on the selected local server. The training process is adaptively configured based on a specific hyperparameter defined for each communication round by utilizing the ADCT. In the final stage, the security of the FL process is considered by using blockchain to store the model parameters. The SECC is introduced to enhance the security of the TSFed optimization algorithm. $K/2$ local servers and the FL server are configured as validators for the blockchain network to prevent a 51% attack, which is a common issue in blockchain-related applications [38].

The main objective of the TSFed is to provide a secure and efficient FL process that is suitable for the IIoT environment. This work takes into account not only model accuracy but also training time and security as the main objectives.

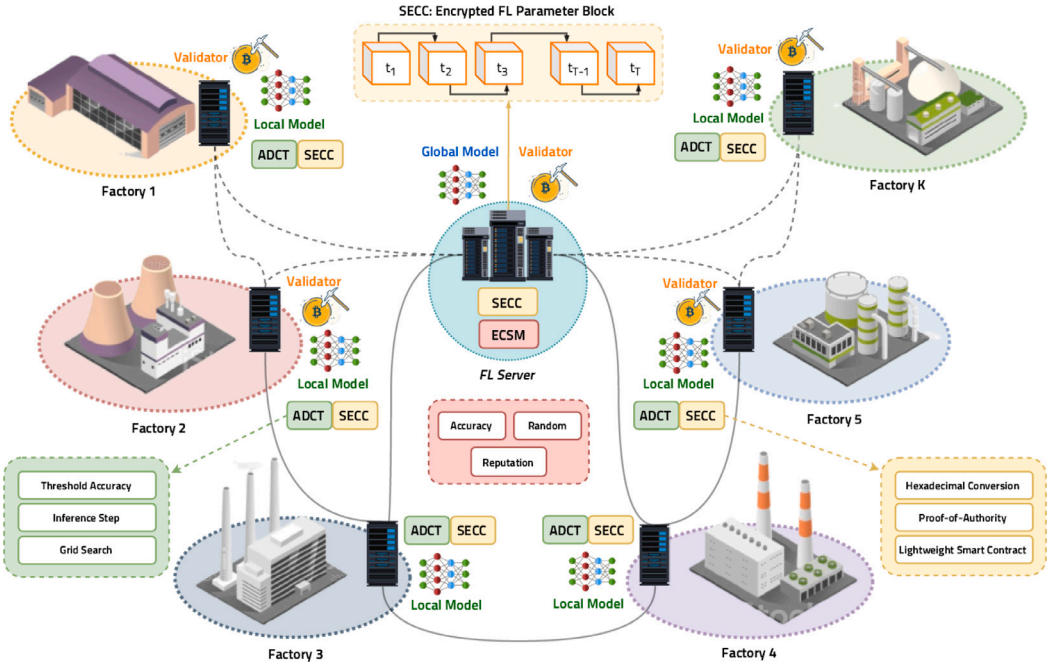



**Fig. 1.** The proposed TSFed optimization mechanism implementation on IIoT environment.

### 3.3. Stage 1: ECSM

To establish a mechanism for multi-criteria client selection, we integrated three distinct approaches with customized parameters for the $ECSM(A, R, r)$, which considers accuracy, reputation, and random client selection.

#### 3.3.1. Accuracy-based selection

The first mechanism adopted from our previous work is accuracy-based client selection (ACS) [15]. The idea is to select high-performance clients based on the centralized evaluation performed on the FL server side to improve the model performance. The ACS is a lightweight client selection mechanism that uses a sorting algorithm to sort the high-performing client and is updated at every communication round.

#### 3.3.2. Reputation-based selection

In the first stage of TSFed, accuracy and the client's reputation is also considered. If the client $k$ transmits updated parameter information to the server, the reputation is calculated based on the client's consistency in terms of evaluation accuracy. For the first round $t = 1$ of FL, the client's reputation is calculated using evaluation accuracy as per Eq. (1).

$$Rep_{(k,t)|t=1} = Acc(k_t)\,. \tag{1}$$

where $Acc(k_t)$ represent accuracy of local server $k$ at $t$ communication round. For the rest of the communication round $t > 1$, the reputation table is calculated using Eq. (2).

$$Rep_{(k,t)|t>1} = \frac{Acc(k_{t-1}) + Acc(k_t)}{2 \times \overline{Acc}}\,. \tag{2}$$

where $Acc(k_{t-1})$ refer to the previous round accuracy value from client $k$, whereas $\overline{Acc}$ denotes the average accuracy of the FL training process and calculated using Eq. (3).

$$\overline{Acc} = \sum_{k=1}^{S_k} \frac{Acc(k_t)}{S_k}\,. \tag{3}$$

where $S_k$ represents the total number of local servers participating in the FL. The average accuracy is calculated using the FL server dataset before the aggregation process.

#### 3.3.3. Random selection

TSFed also incorporates a random client selection mechanism to ensure model generalization across all participants, not just a limited number of local servers trained repeatedly.

The integration of these three mechanisms in the first stage of TSFed is achieved by adjusting the proportion of each client selection approach. The ECSM mechanism is performed after the first communication round, with a total of $S_k$ clients determined by the fraction size $\alpha \times K$. In contrast, all available clients K are utilized at once for the first round of the training process. The size of the client for ECSM is fixed, while the accuracy and reputation-based client selection range from 0.1 to 0.8. It is important to note that the ECSM step will not select the same local server twice. If local server $k$ is selected through the ACS mechanism, it will be excluded from other selection mechanisms, even if it has the highest reputation among other clients. This ensures that the specific number of local servers selected to train their local models always meets the FL requirements.

### 3.4. Stage 2: ADCT

The second stage of the proposed TSFed focuses on improving the training performance of the local servers. In this stage, we employ ADCT to adaptively adjust the training process of each local server. Two parameters are introduced into the $ADCT(\gamma, s)$, where $\gamma$ represents the accuracy threshold, and $s$ represents the step for conducting a grid search to determine the optimal hyperparameters of model training. These two parameters in the ADCT regulate the intensity of the grid search process

$$ADCT(\gamma, s) = \begin{cases} 1, & if \begin{pmatrix} \gamma \geq \overline{Acc} \\ t \bmod s = 0 \end{pmatrix} \\ 0, & if \begin{pmatrix} \gamma < \overline{Acc} \\ t \bmod s \neq 0 \end{pmatrix} \end{cases}. \tag{4}$$

The process of $ADCT(\gamma, s)$ is described in Eq. (4). The grid search will be conducted if the average accuracy $\overline{Acc}$ of all local servers is below the accuracy threshold $\gamma$ or if the current round $t$ is divisible by step $s$. In other circumstances, the grid search process will be omitted, and the training process will use the hyperparameters from the previous round. The grid search is designed to identify the optimal hyperparameters, thus improving the model's performance. This study examines two model hyperparameters: the learning rate $\eta$ and the local epoch $e_l$. Appropriate selection of these two hyperparameters can significantly enhance the performance of DL models. After the local server training process, the best-performing local parameters are saved and forwarded to the blockchain network for aggregation.

### 3.5. Stage 3: SECC

The final stage of the TSFed optimization algorithm is to ensure the security of the local model; hence, the aggregation process is conducted using all local parameters from the local server. In this aspect, we consider two different conditions: (1) Parameter conversion is performed to convert the DL model array into a byte, then transformed to the form of hexadecimal to reduce the file size. (2) Since file conversion is unreliable enough to secure the local model parameter, we adopt a blockchain network to ensure the data is secured.

Blockchain enables a decentralized digital ledger to store information securely and transparently. There are various consensus mechanism that ensures the integrity of stored data in the blockchain network, such as proof-of-work (PoW), proof-of-stake (PoS), and proof-of-authority (PoA). Although blockchain introduces some performance trade-offs, especially in systems where high throughput is crucial, the benefits of enhanced security, improved transparency, and increased data integrity often surpass these issues. For many applications, the use of blockchain provides a positive outcome, delivering a robust solution for secure data handling in a distributed environment.

Considering the application in the IIoT environment, specifically in the model training process where security is a major concern, blockchain is a potential solution. The applicability of the PoW and PoS consensus mechanisms is relatively low due to their high complexity requirement. Industrial IoT requires a fast and secure blockchain network to record and store sensitive information about FL model updates. The PoA-based consensus is the most suitable for this environment. Despite the fact that PoA is less decentralized compared to other consensus algorithms, the efficiency of the PoA for a private network (e.g., IIoT) is flawless. Known and trusted entities usually perform the FL process for private network IIoT network; hence less secure blockchain network with high efficiency like PoA is prioritized. In the proposed SECC, we design the PoA-based consensus with a $K/2$ local server followed by an FL server that acts as a validator to create and validate blocks containing local server parameters.

#### 3.5.1. Smart contract design

Providing lightweight smart contracts to interact with the local server is mandatory. In the proposed SECC, we designed three different functions under the TSFed contract detailed in Algorithm 1.

- *collect_globalparam* function is designed to retrieve global parameter that is stored previously by the FL server to the blockchain. This function is invoked without any additional parameter. The smart contract will check whether the requester's address is included in the eligible list for the global model retrieval process. Additionally, the smart contract ensures that only trusted entities can download the global parameter.

**Algorithm 1:** TSFed Smart Contract

```
Contract TSFed {
    struct Data {
        string parameter;
        uint256 id;
    }
    mapping Data[..];
    address[] clients = [..] ;
    address serverAddress;
    string global_parameter;
    Function store_param (uint256 id, string param) {
        if (msg.sender in clients) {
            Data[id].parameter = param;
        };
    };
    Function aggregate_param (uint256 id) {
        if (msg.sender == serverAddress) {
            return Data[id].parameter;
        };
    };
    Function collect_param ( ) {
        if (msg.sender in clients) {
            return global_parameter;
        };
    };
};
```

- *store_param* function is used to insert local parameters into the blockchain network. These functions require two parameters, particularly client id $cid$ and local_parameter. The store_param function is designed to accept only trustworthy addresses; hence, only specified local servers can interact with the smart contract and store the updated parameter in the blockchain network.
- *aggregate_param* function is used to retrieve updated local parameter that requires a single parameter that represents the client number. FL server will retrieve each local server parameter from the blockchain network for each communication round. The aggregate_param function will return a specific value only if the function requester comes from the FL server address. Thus, this mechanism guarantees that the local parameter information is secured.

### 3.5.2. Smart contract complexity

To provide better insight into the proposed smart contract designed for the SECC stage, we provide a detailed computational complexity (CC) of each function inserted in the TSFed contract, specifically:

- $O(1)$ for *collect_globalparam*. The global parameter collection process designed in this work only requires a simple condition check to validate that the requester is included in the trusted entities with $O(1)$ CC.
- $O(log\ n)$ for *store_param*. The parameter update mechanism is also attached with a simple condition check to verify the local server that tries to store the parameter. The CC of this function is $O(logn)$, where n represents the total number of participants for each training process.
- $O(1)$ for *aggregate_param*. The server performs parameter aggregation. The function returns specific parameters only if the requester comes from the FL server. Thus, the CC of this function is $O(1)$ just for the condition check.

Finally, the overall CC of the proposed smart contract is constant, which is proven by $O(1) + O(log\ n) + O(1) = O(log\ n)$. A larger number of participants selected for the training process increase the CC of the TSFed in a logarithmic manner. This finding indicates the efficiency of the proposed smart contract in the TSFed optimization algorithm.

## 3.6. TSFed integration

The proposed TSFed consists of three optimization stages: 1) client selection mechanism using ECSM; 2) adaptive local training using ADCT; and 3) secure communication channel using SECC. Integration between these optimizations is illustrated in Fig. 2. All available local servers were initially selected for the first communication round to perform the adaptive training using $ADCT(\gamma, s)$.

**Algorithm 2:** TSFed at the FL Server

Initialize global DL model and parameter: $\omega_g$;
Initialize other parameters: $\alpha, K, T, \eta, e$ ;
Initialize clients blockchain address;
Initialize smart contract based on Algorithm 1;
Initialize $ECSM(A, R, r)$;
**for** *each communication round* $t$ *in* $T$ **do**
  $S_k \leftarrow \alpha \times K$;
  Store parameter $\omega_g, \overline{Acc}$ to blockchain;
  **for** *each* $k \in S_k$ *in parallel* **do**
    Perform $LocalServerTraining(\omega_g, t)$;
  **end**
  **for** *each* $k \in S_k$ *in parallel* **do**
    $\omega_k \leftarrow$ Collect parameter from blockchain;
    Calculate accuracy using $Acc(k_t)$;
    Calculate reputation using Equation (1) or (2);
  **end**
  Update $\overline{Acc}$ using (3);
  Adjust accuracy and reputation table using timsort;
  $\omega_g \leftarrow$ aggregate global parameter: $\sum_{k=1}^{K} \frac{\eta k}{\eta} \omega_l{}^k_{t+1}$;
  $Acc, F_1 \leftarrow global\_model(\omega_g).evaluate()$;
**end**

**Algorithm 3:** TSFed at the Local Server

Initialize local DL model: $\omega_l$;
Initialize grid search parameters: $\eta, e$ ;
Initialize $ADCT(\gamma, s)$;
**Function** $LocalServerTraining(\omega_l, t)$;
$\omega_l \leftarrow$ Collect global param $\omega_g$ from blockchain
**if** $t$ *mod* $s = 0$ *or* $\gamma \geq \overline{Acc}$ **then**
  **for** *each* $\eta$ *and* $e$ *configuration* **do**
    $\omega_l \leftarrow$ parameter with the highest accuracy
  **end**
**end**
**else**
  $\omega_l \leftarrow$ train with saved parameter configuration;
**end**
Save the best parameter configuration;
Store updated local parameter $k$, $\omega_l$ to blockchain

It is worth noting that the $ECSM(A, R, r)$ is not employed at the first round since the accuracy and reputation table is still empty. These tables are periodically updated from the second round until the learning process is done. After the training performed by the local server, the best-performing parameter configuration is converted into hexadecimal and stored in the blockchain network via SECC. Using the same method, the FL server retrieves the parameter information of all clients, followed by the centralized evaluation, ACS and reputation table updating and parameter aggregation.

The implementation of TSFed requires initializing several hyperparameters configured on the FL server side, such as the client portion for ECSM, accuracy threshold and step for ADCT. Those parameters must be determined before the FL process is initiated. In addition, since the smart contract we designed requires participant addresses to be configured, all trustworthy local server details must be determined.

Furthermore, the TSFed mechanism is installed separately in FL and local server as follows:

1. *FL Server:* In the FL server, initialization of $ECSM(A, R, r)$ portion is required as shown in Algorithm 2. Then, SECC is also configured on the server side by deploying the smart contract with a predetermined address of trustworthy entities.
2. *Local Server:* The optimization for the client side introduced in this paper focused on the adaptive learning process based on average accuracy $\overline{Acc}$ that is performed using $ADCT(\gamma, s)$. The accuracy threshold $\gamma$ and step $s$ must be configured carefully

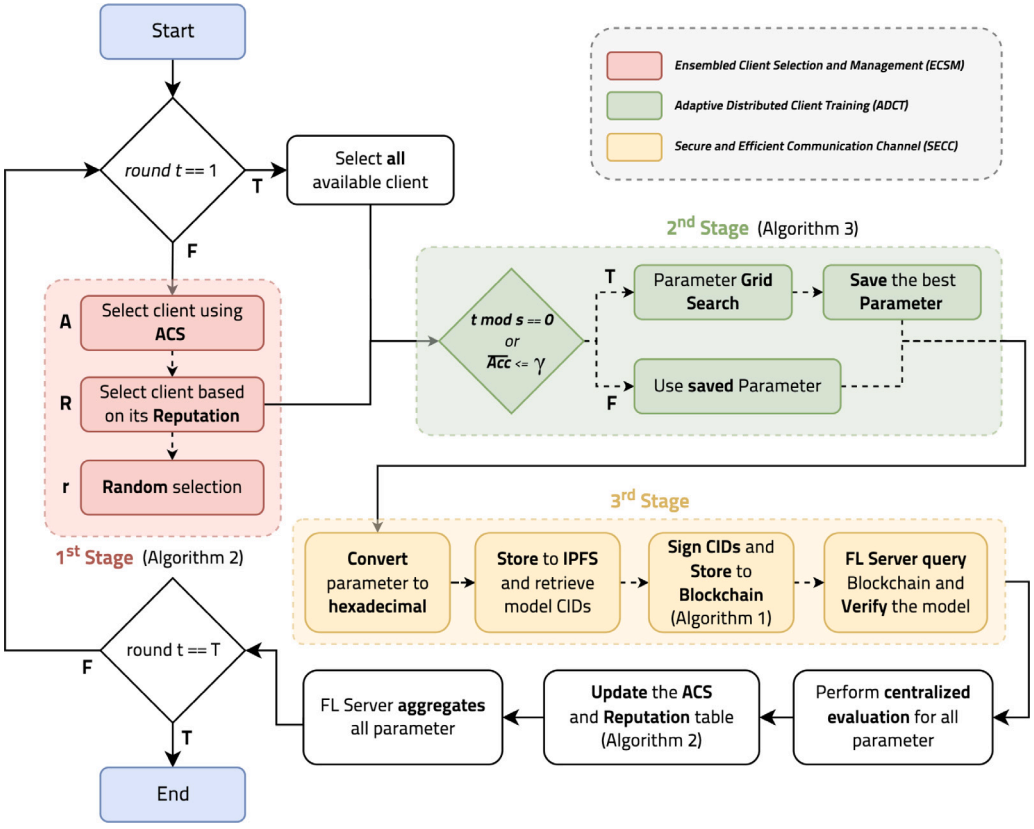



**Fig. 2.** Workflow of the TSFed that consist of three stage optimization: (1) client selection; (2) local training; and (3) security.

to avoid a long training process. Finally, the hyperparameter option for the grid search (e.g., learning rate and epoch) also needs to be defined as shown in Algorithm 3.

## 4. Evaluation and discussion

This section presents the evaluation technique for the proposed TSFed for IIoT networks. First, we present the simulation settings used in this work to compare state-of-the-art mechanisms with the proposed TSFed. Second, we show several performance comparisons of each optimization stage along with TSFed integration performance. Finally, the complexity analysis is also provided to indicate the efficiency of the proposed TSFed for industrial application.

### *4.1. Simulation setup*

Three well-known datasets for FL problems, i.e., MNIST,[1] FMNIST,[2] and CIFAR10[3] are used for the evaluation process with single multilayer perceptron (MLP) layer, dropout layer and one Dense layer. Table 2 summarizes the characteristics of each dataset, followed by the model configuration details. Additionally, Fig. 3 illustrates the deep learning model architecture used in this work. Initially, each type of dataset was converted from images into numerical form with a specific size. For instance, images in the MNIST dataset were converted into 28x28x1 pixel numerical values. To mimic real-world use case scenarios, no other preprocessing methods were applied.

For the comparative analysis, we investigate FedAvg, which is widely adopted for FL applications. Furthermore, we have also demonstrated the performance of the proposed TSFed compared with state-of-the-art works: FedYogi [39] and FedOpt [40], using numerous performance metrics to show the advancement of the proposed optimization mechanism. Initially, the accuracy of the centralized model is assessed on the FL server after the parameters have been aggregated using Eq. (5). This is accomplished to

[1] http://yann.lecun.com/exdb/mnist/
[2] https://github.com/zalandoresearch/fashion-mnist
[3] https://www.cs.toronto.edu/kriz/cifar.html

**Table 2**
Dataset characteristic and DL model configurations.

| Terms | MNIST | FMNIST | CIFAR10 |
|---|---|---|---|
| Dataset types | Images | Images | Images |
| Task complexity | Low | Medium | High |
| Total clients | 60 | 60 | 20 |
| Total rounds | 50 | 50 | 50 |
| Total dataset samples | 60,000 | 60,000 | 60,000 |
| Total training samples | 50,000 | 50,000 | 50,000 |
| Total testing samples | 10,000 | 10,000 | 10,000 |
| Total of classes | 10 | 10 | 10 |
| Sample size | 28 × 28 × 1 | 28 × 28 × 1 | 32 × 32 × 3 |
| MLP layer | 128 | 128 | 128 |
| Dropout layer | 0.2 | 0.2 | 0.2 |
| Optimizer | SGD | SGD | SGD |
| Loss Function | SCC | SCC | SCC |
| Grid Search Step | 5,10,15 | | |
| Local Learning Rate | 0.00001, 0.0001, 0.001, 0.01, 0.1 | | |
| Local Epochs | 1, 2, 3, 4, 5 | | |

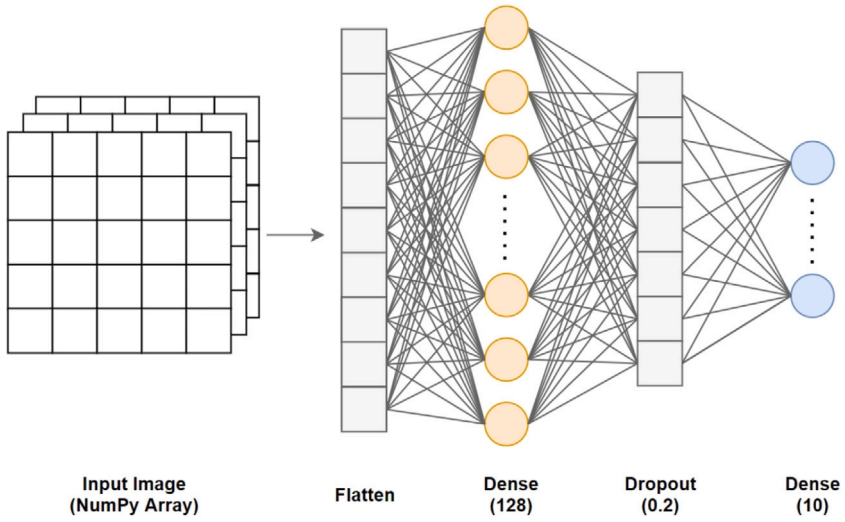



**Fig. 3.** The lightweight global model used for evaluation purposes.

confirm that the FL process can produce a generalized model. Additionally, the F1-score is considered a measure of the model's robustness. In order to calculate the F1-score, precision and recall must be computed using Eqs. (6) and (7), respectively. Finally, the F1-score is obtained using Eq. (8), which is based on the calculated precision and recall values.

$$Accuracy\,(\%) = \frac{T_p + T_n}{T_p + T_n + F_n + F_p} \times 100\,. \tag{5}$$

$$Precision\,(\%) = \frac{T_p}{T_p + F_p} \times 100\,. \tag{6}$$

$$Recall\,(\%) = \frac{T_p}{T_p + F_n} \times 100\,. \tag{7}$$

$$F1 - score\,(\%) = \frac{2 \times Precision \times Recall}{Precision + Recall}\,. \tag{8}$$

where $T_p$ represents true positive, $T_n$ refers to true negative, followed by $F_p$ and $F_n$, which donates false positive and false negative, respectively. In terms of the effectiveness of blockchain implementation, transaction time and gas are considered. The measures are made by investigating the processes from the initial transaction request created by the local or FL server until the transaction is stored in the blockchain.

The implementation of TSFed presented in this paper was carried out using the Flower framework [41]. Flower was constructed on the Python programming language and can be adapted according to the architecture proposed in this study. With regards to the availability of modules and aggregation strategies, Flower offers FedAvg as its primary aggregation technique, along with FedBN [42]

**Table 3**
Hardware and software specification for simulation work.

| Component name | Description |
|---|---|
| Processors | 27x Intel(R) Core(TM) i9-10940X CPU |
| Graphic Cards | 3x NVIDIA GeForce RTX 3090 |
| RAM | 125.5 Gigabytes |
| OS | Ubuntu 20.04.4 LTS |
| Environment & Libraries | Virtualenv with Python 3.7.12. Flower 1.0, TensorFlow 2.9.1, Web3 1.8.1, Solidity 0.8.7+commit.e28d00a7 |

**Table 4**
Performance Comparison among numerous variations of client selection and local training optimization of TSFed.

| Optimization | Variants | Accuracy (%) | | | Round (t) to Accuracy (%) | | |
|---|---|---|---|---|---|---|---|
| | | MNIST | FMNIST | CIFAR10 | MNIST (80%) | FMNIST (60%) | CIFAR10 (35%) |
| Client Selection | ECSM (0.1, 0.8, 0.1) | 81.68 | 70.66 | 38.71 | 43 | 14 | 22 |
| | ECSM (0.2, 0.7, 0.1) | **83.64** | **79.75** | **39.11** | **28** | **12** | **21** |
| | ECSM (0.3, 0.6, 0.1) | 85.64 | 72.44 | 38.90 | 31 | 20 | 23 |
| | ECSM (0.4, 0.5, 0.1) | 85.19 | 75.42 | 38.45 | 33 | 11 | 25 |
| | ECSM (0.5, 0.4, 0.1) | 82.31 | 75.95 | 39.28 | 35 | 13 | 20 |
| | ECSM (0.6, 0.3, 0.1) | 83.02 | 75.42 | 37.62 | 33 | 17 | 25 |
| | ECSM (0.7, 0.2, 0.1) | 80.64 | 73.84 | 39.02 | 45 | 14 | 23 |
| | ECSM (0.8, 0.1, 0.1) | 84.75 | 73.04 | 38.48 | 33 | 18 | 25 |
| Local Training | ADCT (75, 5) | **78.99** | **66.13** | **38.41** | **40** | **15** | **20** |
| | ADCT (75, 10) | 78.55 | 53.37 | 32.78 | 49 | 35 | – |

with batch normalization and adaptive federated optimization methods such as FedOpt and FedYogi. Moreover, Flower is compatible with various AI-based frameworks, including TensorFlow (utilized in this work), PyTorch, and scikit-learn.

In addition to the use of the Flower framework for the SECC implementation, several blockchain tools were employed, including the Web3 libraries, to facilitate interaction with the blockchain network. Since the Web3 libraries are available in Python, seamless synchronization with the Flower framework was possible. The Ganache library was utilized for account creation in local deployment. The Remix integrated development environment (IDE) was employed for developing, deploying, and testing the smart contract for the proposed architecture. A comprehensive list of the hardware and software used in this study is presented in Table 3.

### 4.2. ECSM performance analysis

The performance evaluation was initially performed based on the ensembled client selection mechanism introduced in this paper. Three different methods were used to select FL participants in order to obtain the best allocation option for the highest performance. Notably, the allocation for random client selection was always set to 10% of all participants. Meanwhile, the accuracy and reputation-based selection ranged from 10%-80%. In this work, we utilized 75% of all participants for the training and evaluation process.

Table 4 shows the performance evaluation of the ECSM mechanism with eight different variations on three different datasets. Based on the presented findings, a larger number of accuracy-based client selections does not guarantee the best performance generated by the FL process. Regarding accuracy and round to achieve certain accuracy, ECSM, with 20% accuracy-based client selection, 70% reputation-based client selection, and 10% random client selection, successfully provides the highest performance compared to other possible combinations. The findings also show that the proposed ECSM optimization suits low and high-task complexity.

### 4.3. ADCT performance analysis

The second evaluation was conducted based on the adaptive distributed training of each FL participant. As the proposed ADCT requires two parameters, we design two variations of ADCT parameters: $ADCT(75, 5)$ and $ADCT(75, 5)$. Those two configurations are selected to avoid a high-complexity process. The performance evaluation results of the ADCT are presented in Table 4. Similarly, the performance was investigated under three different datasets. In low task complexity, both ADCT configurations work perfectly. However, for the medium and high-complexity tasks, the $ADCT(75, 5)$ is more reliable compared to $ADCT(75, 10)$. The repetitive checking on the average accuracy is proven to be more effective, as is demonstrated by the findings, especially for medium-to-high complexity jobs. This finding is also supported by the outcomes obtained using the CIFAR10 dataset with $ADCT(75, 10)$ that could not achieve 35% of accuracy under 50 communication rounds.

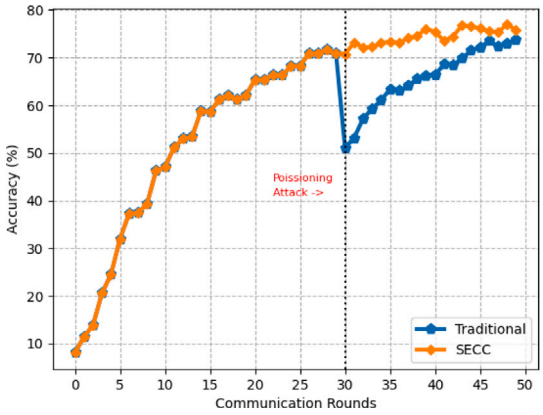



**Fig. 4.** Performance comparison of proposed SECC with traditional FL affected by single poisoning attack.

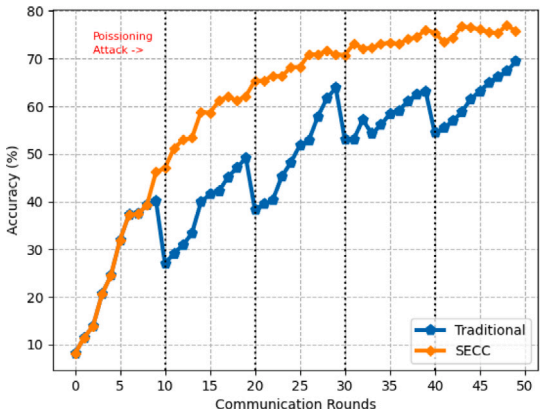



**Fig. 5.** Performance comparison of proposed SECC with traditional FL affected by multiple poisoning attack.

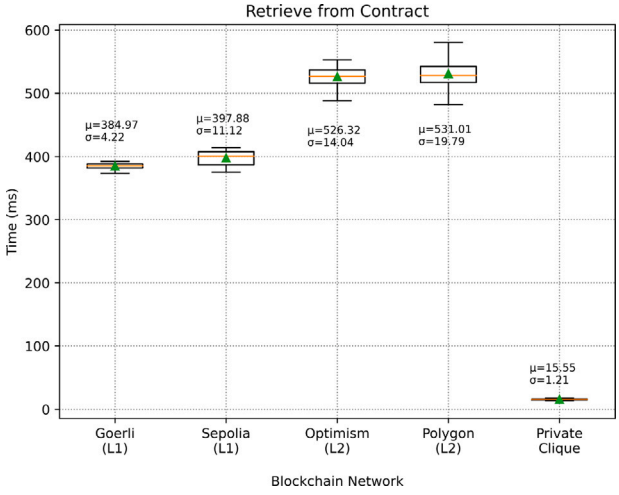



**Fig. 6.** Smart contract transaction time to retrieve information on various blockchain networks (public and private).

### *4.4. SECC performance analysis*

The last optimization proposed in TSFed focuses on maintaining parameter security and mitigating possible attacks in the FL case, specifically poisoning attacks. The proposed SECC is equipped with blockchain technology, using a lightweight smart contract

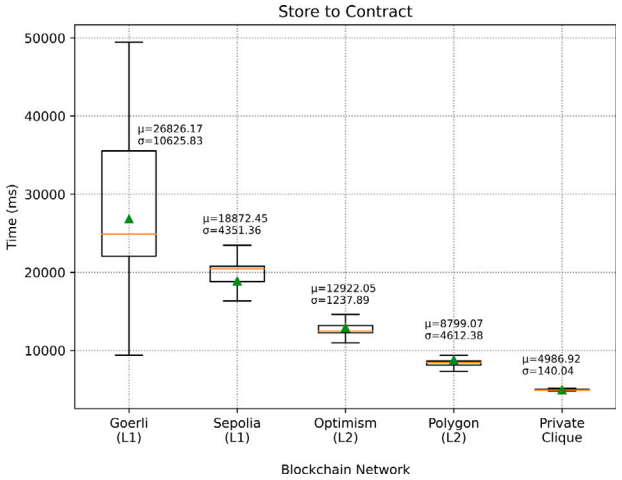



**Fig. 7.** Smart contract transaction time to store information on various blockchain networks (public and private).

deployed under the PoA network. The evaluation was performed by forwarding randomly generated parameters to the FL server. Fig. 4 shows the performance drop after the poisoning attack occurs in round 30 for traditional FL security. In contrast, the proposed SECC can maintain its performance since the poisonous attack could not store the data in the FL, which secures the aggregation process. Additionally, multiple attack scenarios occurring every ten communication rounds are also considered. The findings are illustrated in Fig. 5, where SECC is superior in maintaining model accuracy compared to traditional FL strategy. These results demonstrate that the proposed SECC can secure the FL process from poisoning attacks.

Not only the security concerns but also the efficiency of the blockchain network are assessed in this work. Fig. 6 details several blockchain networks in both public and private configurations for retrieving or reading information from the blockchain network. Meanwhile, Fig. 7 depicts the time required to store information to the blockchain network with both public and private configurations. The results show that the private PoA-based network provides the best performance compared to its counterparts. Therefore, the proposed SECC is able to maintain low processing times as it utilizes the Clique or PoA-based private blockchain network.

### 4.5. Comparative analysis

Based on previous evaluation results, the best-performing parameters were obtained for ECSM and ADCT. These parameters were combined and compared with several baseline strategies in FL.

Table 5 presents a comparison of precision and recall metrics for the proposed TSFed strategy against three baseline Federated Learning (FL) strategies: FedAvg, FedOpt, and FedYogi. The metrics are evaluated across three different datasets: MNIST, FMNIST, and CIFAR10. For the MNIST dataset, TSFed shows the highest Precision at 83.22% and Recall at 83.28%, outperforming all other strategies. In the FMNIST dataset, TSFed also leads with a precision of 73.14% and a recall of 73.10%. Similarly, for the CIFAR10 dataset, TSFed achieves the highest precision and recall among the strategies, with values at 40.92% and 42.66% respectively. This indicates that TSFed consistently provides better performance in terms of both precision and recall across all three datasets compared to the baseline methods.

To better understand TSFed's performance, three datasets were used to conduct extensive evaluations with three different strategies, as detailed in Table 6. The results were obtained from five different tests. The proposed TSFed achieved $83.75 \pm 1.13\%$ accuracy with the MNIST dataset. In contrast, the baseline strategies FedOpt produced $80.59 \pm 1.75\%$, followed by FedAvg and FedYogi with $75.82 \pm 1.42\%$ and $73.26 \pm 1.93\%$, respectively. Similarly, based on the F1-Score, the proposed TSFed mechanism provided 73.12%, while FedOpt gave $80.32 \pm 1.65\%$, and FedAvg and FedYogi provided $75.21 \pm 1.23\%$ and $72.90 \pm 2.46\%$, respectively. Additionally, the required rounds to achieve specific accuracy for different datasets are also presented in Table 6. The proposed TSFed is superior to other strategies, as the convergence time provided by the three-stage optimization is faster compared to state-of-the-art strategies. Furthermore, the performance of each strategy is depicted in Figs. 8, 9, and Fig. 10 for the MNIST, FMNIST, and CIFAR10 datasets, respectively. It can be observed that the performance of TSFed outperforms the other approaches for all task complexities.

### 4.6. TSFed complexity analysis

The CC of the SECC in the proposed TSFed is presented in Section 3. Inserting an additional mechanism into the traditional FL process inevitably increases the training time. The overall training time required to finish 50 communication rounds for FedAvg is 21.09 min, slightly higher than FedOpt with 21.06 min, whereas FedYogi requires 21.91 min. This time grows as additional mechanisms are added to the strategies. Based on Fig. 11, the proposed TSFed needs 23.24 min to perform 50 communication

**Table 5**
Precision and Recall of the proposed TSFed with baseline FL strategy on three different datasets.

| Strategy | Precision (%) | | | Recall (%) | | |
|---|---|---|---|---|---|---|
| | MNIST | FMNIST | CIFAR10 | MNIST | FMNIST | CIFAR10 |
| FedAvg [10] | 76.12 | 62.46 | 35.23 | 74.32 | 65.79 | 36.57 |
| FedOpt [40] | 79.62 | 65.42 | 39.21 | 81.03 | 66.45 | 36.04 |
| FedYogi [39] | 72.76 | 62.51 | 35.92 | 73.04 | 65.19 | 36.50 |
| TSFed | **83.22** | **73.14** | **40.92** | **83.28** | **73.10** | **42.66** |

**Table 6**
Comparative Performance Evaluation of the proposed TSFed with baseline FL strategy on three different datasets.

| Strategy | Accuracy (%) | | | F1-Score(%) | | | Round (t) to Accuracy (%) | | |
|---|---|---|---|---|---|---|---|---|---|
| | MNIST | FMNIST | CIFAR10 | MNIST | FMNIST | CIFAR10 | MNIST (80%) | FMNIST (60%) | CIFAR10 (35%) |
| FedAvg [10] | 75.82 ± 1.42 | 64.64 ± 2.31 | 36.37 ± 2.40 | 75.21 ± 1.23 | 64.08 ± 2.10 | 35.89 ± 2.03 | – | 25 | 35 |
| FedOpt [40] | 80.59 ± 1.75 | 66.71 ± 1.73 | 38.13 ± 1.84 | 80.32 ± 1.65 | 65.93 ± 1.69 | 37.56 ± 1.83 | 47 | 19 | 25 |
| FedYogi [39] | 73.26 ± 1.93 | 64.71 ± 2.01 | 36.80 ± 2.31 | 72.90 ± 2.46 | 63.82 ± 2.76 | 36.21 ± 2.93 | – | 26 | 32 |
| TSFed | **83.75 ± 1.13** | **73.84 ± 1.20** | **42.10 ± 1.28** | **83.25 ± 1.19** | **73.12 ± 1.26** | **41.77 ± 1.34** | **33** | **14** | **19** |

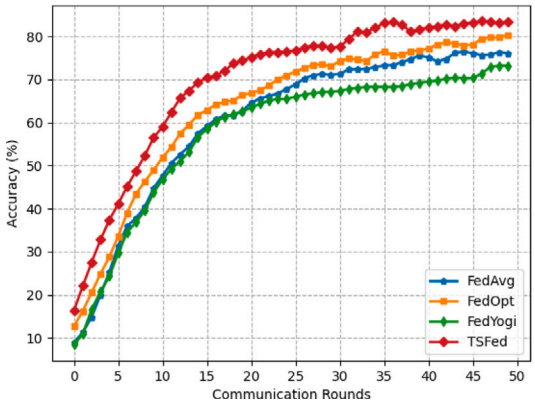



**Fig. 8.** TSFed performance on tested with MNIST dataset.

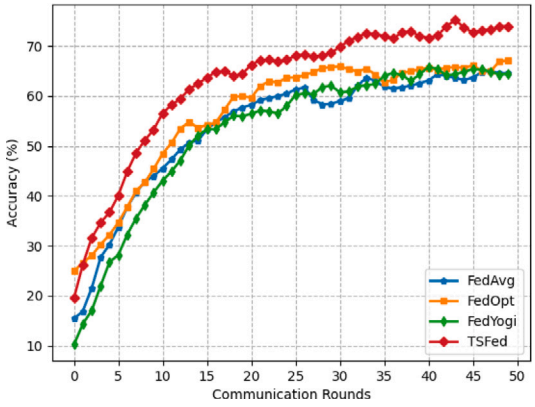



**Fig. 9.** TSFed performance on tested with FMNIST dataset.

rounds for the MNIST dataset. Considering the model's performance at the end of the training process, the proposed TSFed, which requires the longest training time, achieves the highest results, as detailed in Table 6. Moreover, the total time to achieve 80% accuracy using the MNIST dataset was calculated to show the efficiency of the proposed TSFed and illustrated in Fig. 11 (orange bar). FedAvg and FedYogi fail to achieve 80% accuracy in the MNIST dataset, which shows the poor performance of state-of-the-art strategies. In addition, FedOpt requires 19.79 min to achieve 80% performance investigated with the MNIST dataset, whereas the proposed TSFed only takes 15.33 min. Despite the fact that the total training time for TSFed is slightly longer than that of other

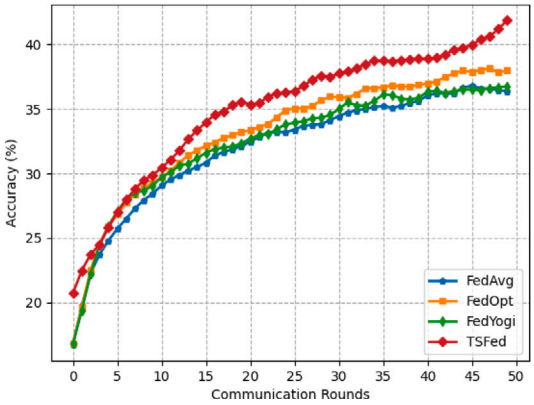



**Fig. 10.** TSFed performance on tested with CIFAR10 dataset.

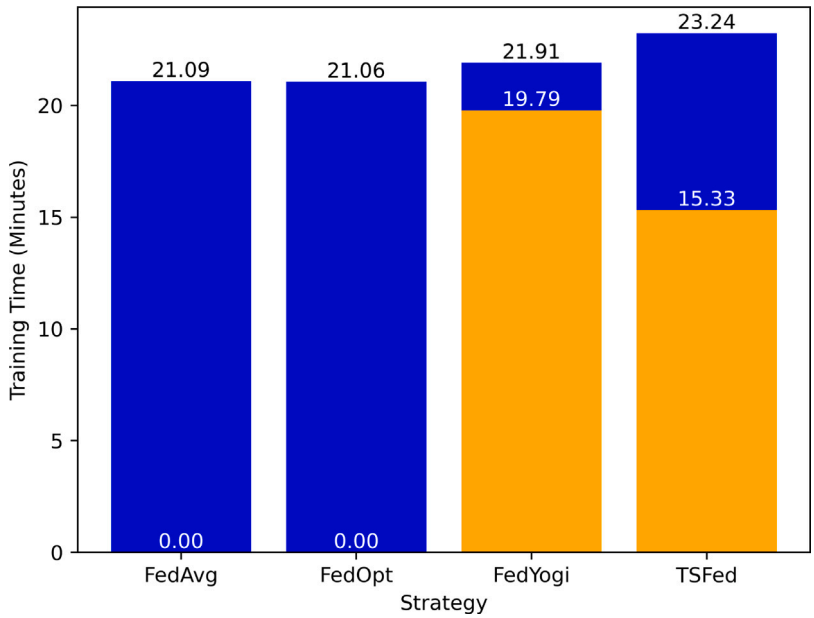



**Fig. 11.** Time evaluation for different FL strategies and proposed TSFed.

state-of-the-art methods, the performance trade-off in terms of efficiency to achieve 80% accuracy is better with TSFed. Based on these findings, TSFed is shown to be 29.09% more efficient than FedOpt.

### *4.7. Overall performance analysis*

Based on various performance metrics used to assess the proposed TSFed mechanism alongside state-of-the-art methods, the advancement of the proposed technique is consistent. These results are achievable due to the multi-stage optimization process in the proposed method. Initially, the ECSM filters which participants are eligible for the training process. Second, the selected clients conduct individual optimizations using ADCT to achieve high accuracy performance. Lastly, to secure the model parameter data transfer, a blockchain network under the SECC mechanism is applied. This three-stage mechanism is effective and efficient in improving the FL system.

## 5. Conclusion and future work

### *5.1. Conclusion*

This work presents a three-stage optimization mechanism that enables a secure and efficient FL process for industrial IoT networks. The TSFed optimization consists of ECSM for client selection that considers multiple factors (e.g., accuracy, reputation, and randomness). ADCT for adaptive client training requires two parameters: average accuracy and grid search step. Lastly, SECC

is introduced to secure the updated parameter from malicious attacks. Based on the extensive performance evaluation conducted on three different datasets with different task complexities, the performance of TSFed outperforms several baseline models for FL, namely FedAvg, FedYogi, and FedOpt, in terms of accuracy and F1-score. Furthermore, we conducted an investigation of poison attacks to evaluate the robustness of the Secure and Efficient Communication Channel (SECC). The findings confirm that the TSFed mechanism successfully maintains the security of parameters, even in the event of such attacks, demonstrating its robustness. Additionally, we measured the efficiency of TSFed by calculating the time required to achieve 80% accuracy on the MNIST datasets. Our results show that TSFed achieves this benchmark with a 29.09% increase in efficiency compared to state-of-the-art strategies, underscoring its enhanced performance in both security and operational speed.

### 5.2. Future works

We plan to explore the large-scale implementation of our proposed TSFed mechanism using a distributed edge computing approach. This exploration will focus on several key aspects, such as:

- Scalability: We aim to investigate how the TSFed can be efficiently scaled across multiple edge devices in diverse geographic locations, examining the impact on performance and reliability.
- Latency: Given the critical importance of real-time data processing in IIoT environments, we will assess the latency challenges associated with edge computing and develop strategies to minimize response times while maintaining high accuracy.
- Data Privacy and Security: Extending our research to distributed edge computing raises additional concerns regarding data privacy and security. We will explore enhanced cryptographic measures and robust authentication protocols to safeguard data across decentralized nodes.
- Resource Optimization: We intend to optimize resource allocation, focusing on energy efficiency and computational resource management, to ensure the sustainable deployment of FL in resource-constrained edge environments.

By addressing these areas, we hope to advance the practical application of FL in IIoT, ultimately leading to more robust, efficient, and secure industrial systems.

## CRediT authorship contribution statement

**Made Adi Paramartha Putra:** Writing – original draft, Visualization, Validation, Software, Methodology, Data curation. **Nyoman Bogi Aditya Karna:** Writing – review & editing, Resources, Investigation. **Ahmad Zainudin:** Visualization, Software. **Dong-Seong Kim:** Writing – review & editing, Validation, Supervision. **Jae-Min Lee:** Writing – review & editing, Supervision, Methodology, Investigation, Conceptualization.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Data availability

Data will be made available on request.

## Acknowledgments

This work was partly supported by Innovative Human Resource Development for Local Intellectualization program through the Institute of Information & Communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT) (IITP-2024-RS-2020-II201612, 33%) and by Priority Research Centers Program through the NRF funded by the MEST(2018R1A6A1A03024003, 33%) and by the MSIT, Korea, under the ICAN support program(IITP-2024-00156394, 34%) supervised by the IITP.

## References

[1] X. Liu, C. Qian, W.G. Hatcher, H. Xu, W. Liao, W. Yu, Secure internet of things (IoT)-based smart-world critical infrastructures: Survey, case study and research opportunities, IEEE Access 7 (2019) 79523–79544, http://dx.doi.org/10.1109/ACCESS.2019.2920763.

[2] W. Liao, C. Luo, S. Salinas, P. Li, Efficient secure outsourcing of large-scale convex separable programming for big data, IEEE Trans. Big Data 5 (3) (2019) 368–378, http://dx.doi.org/10.1109/TBDATA.2017.2787198.

[3] P. Li, Z. Chen, L.T. Yang, Q. Zhang, M.J. Deen, Deep convolutional computation model for feature learning on big data in internet of things, IEEE Trans. Ind. Inform. 14 (2) (2018) 790–798, http://dx.doi.org/10.1109/TII.2017.2739340.

[4] State of IOT 2022: Number of connected IOT devices growing 18% to 14.4 billion globally, 2022, URL https://iot-analytics.com/number-connected-iot-devices/.

[5] W. Sun, J. Liu, Y. Yue, AI-enhanced offloading in edge computing: When machine learning meets industrial IoT, IEEE Netw. 33 (5) (2019) 68–74, http://dx.doi.org/10.1109/MNET.001.1800510.

[6] F. Foukalas, A. Tziouvaras, Edge artificial intelligence for industrial internet of things applications: An industrial edge intelligence solution, IEEE Ind. Electron. Mag. 15 (2) (2021) 28–36, http://dx.doi.org/10.1109/MIE.2020.3026837.

[7] S.S. Arumugam, R. Badrinath, A.H. Herranz, J. Höller, C.R.B. Azevedo, B. Xiao, V. Tudor, Accelerating industrial IoT application deployment through reusable AI components, in: 2019 Global IoT Summit (GIoTS), 2019, pp. 1–4, http://dx.doi.org/10.1109/GIOTS.2019.8766398.

[8] P. Nirmala, S. Ramesh, M. Tamilselvi, G. Ramkumar, 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, 2022, pp. 1–6, http://dx.doi.org/10.1109/ACCAI53970.2022.9752651.

[9] A.M. Elbir, S. Coleri, A.K. Papazafeiropoulos, P. Kourtessis, S. Chatzinotas, A hybrid architecture for federated and centralized learning, IEEE Trans. Cognit. Commun. Netw. 8 (3) (2022) 1529–1542, http://dx.doi.org/10.1109/TCCN.2022.3181032.

[10] B. McMahan, E. Moore, D. Ramage, S. Hampson, B.A. y Arcas, Communication-efficient learning of deep networks from decentralized data, in: Artificial Intelligence and Statistics, PMLR, 2017, pp. 1273–1282.

[11] F. Sattler, S. Wiedemann, K.-R. Müller, W. Samek, Robust and communication-efficient federated learning from non-i.i.d. data, IEEE Trans. Neural Netw. Learn. Syst. 31 (9) (2020) 3400–3413, http://dx.doi.org/10.1109/TNNLS.2019.2944481.

[12] D.C. Nguyen, M. Ding, P.N. Pathirana, A. Seneviratne, J. Li, H. Vincent Poor, Federated learning for internet of things: A comprehensive survey, IEEE Commun. Surv. Tutor. 23 (3) (2021) 1622–1658, http://dx.doi.org/10.1109/COMST.2021.3075439.

[13] P. Boobalan, S.P. Ramu, Q.-V. Pham, K. Dev, S. Pandya, P.K.R. Maddikunta, T.R. Gadekallu, T. Huynh-The, Fusion of federated learning and industrial internet of things: A survey, Comput. Netw. 212 (2022) 109048, http://dx.doi.org/10.1016/j.comnet.2022.109048.

[14] T. Nishio, R. Yonetani, Client selection for federated learning with heterogeneous resources in mobile edge, in: ICC 2019 - 2019 IEEE International Conference on Communications, ICC, 2019, pp. 1–7, http://dx.doi.org/10.1109/ICC.2019.8761315.

[15] M.A.P. Putra, A.R. Putri, A. Zainudin, D.-S. Kim, J.-M. Lee, ACS: Accuracy-based client selection mechanism for federated industrial IoT, Int. Things 21 (2023) 100657, http://dx.doi.org/10.1016/j.iot.2022.100657.

[16] I. Mohammed, S. Tabatabai, A. Al-Fuqaha, F.E. Bouanani, J. Qadir, B. Qolomany, M. Guizani, Budgeted online selection of candidate IoT clients to participate in federated learning, IEEE Internet Things J. 8 (7) (2021) 5938–5952, http://dx.doi.org/10.1109/JIOT.2020.3036157.

[17] M. Cao, Y. Zhang, Z. Ma, M. Zhao, C2S: Class-aware client selection for effective aggregation in federated learning, High-Confidence Comput. 2 (3) (2022) 100068, http://dx.doi.org/10.1016/j.hcc.2022.100068.

[18] P. Tian, W. Liao, W. Yu, E. Blasch, WSCC: A weight-similarity-based client clustering approach for non-IID federated learning, IEEE Internet Things J. 9 (20) (2022) 20243–20256, http://dx.doi.org/10.1109/JIOT.2022.3175149.

[19] N. Bouacida, J. Hou, H. Zang, X. Liu, Adaptive federated dropout: Improving communication efficiency and generalization for federated learning, in: IEEE INFOCOM 2021 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 2021, pp. 1–6, http://dx.doi.org/10.1109/INFOCOMWKSHPS51825.2021.9484526.

[20] A. Omar, T. Abd El-Hafeez, Optimizing epileptic seizure recognition performance with feature scaling and dropout layers, Neural Comput. Appl. 36 (6) (2023) 2835–2852, http://dx.doi.org/10.1007/s00521-023-09204-6.

[21] F. Varno, M. Saghayi, L. Rafiee Sevyeri, S. Gupta, S. Matwin, M. Havaei, AdaBest: Minimizing client drift in federated learning via adaptive bias estimation, in: Computer Vision – ECCV 2022, Springer Nature Switzerland, Cham, 2022, pp. 710–726.

[22] J. Park, D. Yoon, S. Yeo, S. Oh, AMBLE: Adjusting mini-batch and local epoch for federated learning with heterogeneous devices, J. Parallel Distrib. Comput. 170 (2022) 13–23, http://dx.doi.org/10.1016/j.jpdc.2022.07.009.

[23] B. Jiang, J. Li, H. Wang, H. Song, Privacy-preserving federated learning for industrial edge computing via hybrid differential privacy and adaptive compression, IEEE Trans. Ind. Inform. 19 (2) (2023) 1136–1144, http://dx.doi.org/10.1109/TII.2021.3131175.

[24] A.P. Kalapaaking, I. Khalil, M.S. Rahman, M. Atiquzzaman, X. Yi, M. Almashor, Blockchain-based federated learning with secure aggregation in trusted execution environment for internet-of-things, IEEE Trans. Ind. Inform. 19 (2) (2023) 1703–1714, http://dx.doi.org/10.1109/TII.2022.3170348.

[25] S. Abdulrahman, H. Tout, A. Mourad, C. Talhi, FedMCCS: Multicriteria client selection model for optimal IoT federated learning, IEEE Internet Things J. 8 (6) (2021) 4723–4735, http://dx.doi.org/10.1109/JIOT.2020.3028742.

[26] L. Yu, R. Albelaihi, X. Sun, N. Ansari, M. Devetsikiotis, Jointly optimizing client selection and resource management in wireless federated learning for internet of things, IEEE Internet Things J. 9 (6) (2022) 4385–4395, http://dx.doi.org/10.1109/JIOT.2021.3103715.

[27] F. Shi, C. Hu, W. Lin, L. Fan, T. Huang, W. Wu, VFedCS: Optimizing client selection for volatile federated learning, IEEE Internet Things J. 9 (24) (2022) 24995–25010, http://dx.doi.org/10.1109/JIOT.2022.3195073.

[28] Y. Shi, Z. Liu, Z. Shi, H. Yu, Fairness-aware client selection for federated learning, in: 2023 IEEE International Conference on Multimedia and Expo, ICME, IEEE, 2023, pp. 324–329.

[29] J. Zhang, J. Wang, Y. Li, F. Xin, F. Dong, J. Luo, Z. Wu, Addressing heterogeneity in federated learning with client selection via submodular optimization, ACM Trans. Sen. Netw. 20 (2) (2024) http://dx.doi.org/10.1145/3638052.

[30] H. Zhang, J. Liu, J. Jia, Y. Zhou, H. Dai, D. Dou, FedDUAP: Federated learning with dynamic update and adaptive pruning using shared data on the server, 2022, arXiv preprint arXiv:2204.11536.

[31] Y. Miao, Z. Liu, H. Li, K.-K.R. Choo, R.H. Deng, Privacy-preserving Byzantine-robust federated learning via blockchain systems, IEEE Trans. Inf. Forensics Secur. 17 (2022) 2848–2861, http://dx.doi.org/10.1109/TIFS.2022.3196274.

[32] Z. Yang, M. Zhou, H. Yu, R.O. Sinnott, H. Liu, Efficient and secure federated learning with verifiable weighted average aggregation, IEEE Trans. Netw. Sci. Eng. 10 (1) (2023) 205–222, http://dx.doi.org/10.1109/TNSE.2022.3206243.

[33] L. Li, M. Duan, D. Liu, Y. Zhang, A. Ren, X. Chen, Y. Tan, C. Wang, FedSAE: A novel self-adaptive federated learning framework in heterogeneous systems, in: 2021 International Joint Conference on Neural Networks, IJCNN, 2021, pp. 1–10, http://dx.doi.org/10.1109/IJCNN52387.2021.9533876.

[34] Q. Zhang, P. Palacharla, M. Sekiya, J. Suga, T. Katagiri, Blockchain-based secure aggregation for federated learning with a traffic prediction use case, in: 2021 IEEE 7th International Conference on Network Softwarization (NetSoft), 2021, pp. 372–374, http://dx.doi.org/10.1109/NetSoft51509.2021.9492652.

[35] Y.E. Oktian, B. Stanley, S.-G. Lee, Building trusted federated learning on blockchain, Symmetry 14 (7) (2022) http://dx.doi.org/10.3390/sym14071407.

[36] M.R. Behera, S. Upadhyay, S. Shetty, Federated learning using smart contracts on blockchains, based on reward driven approach, 2021, arXiv preprint arXiv:2107.10243.

[37] J. Zhang, J. Chen, D. Wu, B. Chen, S. Yu, Poisoning attack in federated learning using generative adversarial nets, in: 2019 18th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/13th IEEE International Conference on Big Data Science and Engineering (TrustCom/BigDataSE), 2019, pp. 374–380, http://dx.doi.org/10.1109/TrustCom/BigDataSE.2019.00057.

[38] F.A. Aponte-Novoa, A.L.S. Orozco, R. Villanueva-Polanco, P. Wightman, The 51% attack on blockchains: A mining behavior study, IEEE Access 9 (2021) 140549–140564, http://dx.doi.org/10.1109/ACCESS.2021.3119291.

[39] S. Reddi, Z. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Konečnỳ, S. Kumar, H.B. McMahan, Adaptive federated optimization, 2020, arXiv preprint arXiv:2003.00295.

[40] M. Asad, A. Moustafa, T. Ito, FedOpt: Towards communication efficiency and privacy preservation in federated learning, Appl. Sci. 10 (8) (2020) http://dx.doi.org/10.3390/app10082864, URL https://www.mdpi.com/2076-3417/10/8/2864.

[41] D.J. Beutel, T. Topal, A. Mathur, X. Qiu, T. Parcollet, P.P. de Gusmão, N.D. Lane, Flower: A friendly federated learning research framework, 2020, arXiv preprint arXiv:2007.14390.

[42] X. Li, M. Jiang, X. Zhang, M. Kamp, Q. Dou, Fedbn: Federated learning on non-iid features via local batch normalization, 2021, arXiv preprint arXiv:2102.07623.