mathematics

MDPI

Article

An Energy Efficient Specializing DAG Federated Learning Based on Event-Triggered Communication

Xiaofeng Xue [1], Haokun Mao [1], Qiong Li [1,*], Furong Huang [2] and Ahmed A. Abd El-Latif [3,4]

[1] Information Countermeauser Technique Institute, School of Cyberspace Science, Faculty of Computing, Harbin Institute of Technology, Harbin 150001, China
[2] School of International Studies, Harbin Institute of Technology, Harbin 150001, China
[3] EIAS Data Science Lab, College of Computer and Information Sciences, Prince Sultan University, Riyadh 11586, Saudi Arabia
[4] Department of Mathematics and Computer Science, Faculty of Science, Menoufia University, Menofia 32511, Egypt
* Correspondence: qiongli@hit.edu.cn

Abstract: Specializing Directed Acyclic Graph Federated Learning (SDAGFL) is a new federated learning framework with the advantages of decentralization, personalization, resisting a single point of failure, and poisoning attack. Instead of training a single global model, the clients in SDAGFL update their models asynchronously from the devices with similar data distribution through Directed Acyclic Graph Distributed Ledger Technology (DAG-DLT), which is designed for IoT scenarios. Because of many the features inherited from DAG-DLT, SDAGFL is suitable for IoT scenarios in many aspects. However, the training process of SDAGFL is quite energy consuming, in which each client needs to compute the confidence and rating of the nodes selected by multiple random walks by traveling the ledger with 15–25 depth to obtain the "reference model" to judge whether or not to broadcast the newly trained model. As we know, the energy consumption is an important issue for IoT scenarios, as most devices are battery-powered with strict energy restrictions. To optimize SDAGFL for IoT, an energy-efficient SDAGFL based on an event-triggered communication mechanism, i.e., ESDAGFL, is proposed in this paper. In ESDAGFL, the new model is broadcasted only in the event that the new model is significantly different from the previous one, instead of traveling the ledger to search for the "reference model". We evaluate the ESDAGFL on the FMNIST-clustered and Poets dataset. The simulation is performed on a platform with Intel®Core™ i7-10700 CPU (CA,USA). The simulation results demonstrate that ESDAGFL can reach a balance between training accuracy and specialization as good as SDAGFL. What is more, ESDAGFL can reduce the energy consumption by 42.5% and 51.7% for the FMNIST-clustered and Poets datasets, respectively.

Keywords: energy efficient; federated learning; event-triggered communication; DAG-DLT

MSC: 68T99

Citation: Xue, X.; Mao, H.; Li, Q.; Huang, F.; Abd El-Latif, A.A. An Energy Efficient Specializing DAG Federated Learning Based on Event-Triggered Communication. *Mathematics* **2022**, *10*, 4388. https://doi.org/10.3390/math10224388

Academic Editor: Daniel-Ioan Curiac

Received: 9 October 2022
Accepted: 15 November 2022
Published: 21 November 2022

1. Introduction

With the Internet of Things (IoT) development, data have become more diversified and distributed, stimulating the demand for privacy and security [1,2]. As a result, traditional centralized cloud-based machine learning is being challenged. A few machine learning techniques have been proposed to meet these challenges. A technique called Federated Learning (FL) [3] provides a promising solution that allows the clients to work together to build a global machine learning model without sharing the local data on their own devices. A typical federated learning framework consists of a server and some local clients. The clients in FL can access the same global model and train it using local data. Then, they update the trained local model to the server. Once the locally trained models are received, the server will aggregate them as a new global model and feed it back to the clients for

the next local update. FL will repeat this process several rounds until the global model converges or the number of repetitions meets the predetermined target.

This way, the client can protect data privacy as FL implements model training without collecting user data. Nevertheless, there are still some challenges in Federated Learning, as follows:

- **Device Heterogeneity:** In FL, the server must wait for all the selected clients to complete the local training before aggregating the local models. However, the computing power and network bandwidth vary from client to client [4,5]. The clients with a higher computing power and network bandwidth can complete the local training faster than the others. Thus, the server will wait a long time until the slowest client completes the local training before starting the next round. This leads to the performance bottleneck of FL [6];
- **Data Heterogeneity:** A significant difference between FL and other machine learning is that model's training in FL is completed on the clients with the data of Non-IID. The Non-IID data distribution will cause a significant decrease in the global model's accuracy. The existing reach shows that the accuracy can be reduced by 55% of the neural network trained with the highly skewed Non-IID dataset [7];
- **Single Point of Failure:** In the traditional FL, there is a single central server to aggregate the local model and publish the new global model. Therefore, the hacker can attack the central server to cause a single point of failure, leading to a performance decrease or even a failure of FL training;
- **Poisoning Attack:** The clients in FL are not all honest, and some clients may use the poisoned local data to train the model and then send the poisoned model to the server. However, the server cannot detect the poisoned model and will aggregate it into the global model. The poisoning attack will cause a decrease in the global model's accuracy [8].

Researchers have conducted much research to solve the above problems and proposed various solutions. For example, blockchain technology has been introduced into FL [9–11] to address the problem of a single point of failure and poisoning attack. Some distributed federated learning frameworks [12–14] have been proposed to adapt to the devices' heterogeneity. The problem of data heterogeneity has been widely studied, and different algorithms of framework have been proposed [15,16]. However, the existing studies focus on solving one or more problems, but not all of them. The framework proposed by Beilharz et al., called Specializing Directed Acyclic Graph Federated Learning (SDAGFL), realizes adaptive data and device heterogeneity, and is robust for poisoning in a fully decentralized federated learning environment.

SDAGFL introduced the Tangle [17] into the FL. The Tangle is a typical DAG Distributed Ledger Technology (DLT) designed by the IOTA foundation for the devices of the Internet of Things (IoT) to participate in a low-energy network. The Tangle uses a DAG data structure to store the transactions, allowing multiple transactions to be added to the ledger simultaneously, and it can achieve consensus similar to the Nakamoto DLT [18], which is also called blockchain, in a distributed system. Furthermore, the Tangle has a higher Transaction Per Second (TPS) than the blockchain because the transactions in the Tangle can be confirmed within minutes. In summary, the Tangle has the advantages of high TPS, low energy usage, and decentralization. Therefore, the Tangle is considered to be suitable for the scenario that includes many distributed devices.

SDAGFL inherits the features of the Tangle. In the SDAGFL, the participating clients use the DAG-DLT for the communication of models and an accuracy-biased random walk to obtain the models from other devices with similar data distribution to update their local model. It does not only overcome the challenges of device heterogeneity, failure of a single point, and poisoning attack, but also creates a balance between reaching a consensus on a generalized model and personalizing the model to the clients, which is different from the traditional FL framework, where all the participating clients train and reach a consensus for a global model together.

With the above advantages of the SDAGFL, it is suitable for the FL in an IoT scenario. However, some devices in the IoT are usually powered by a battery with strict energy restrictions [19]. It is necessary to reduce the energy consumption of federated learning. The benefit of reducing energy consumption is two-fold. On the one hand, it can reduce the number of charging times and prolong the device's service life. On the other hand, it is friendly to the environment. In addition, SDAGFL has some unnecessary energy consumption according to our study. Therefore, reducing the energy consumption while maintaining training performance is a problem that needs to be solved to promote the application of SDAGFL in IoT.

To reduce the energy consumption of the SDAGFL, we first analyze the energy consumption in SDAGFL, and we propose an event-triggered communication-based SDAGFL called event-triggered SDAGFL (ESDAGFL) to reduce the energy consumption of SDAGFL. The main contributions of this article are summarized as follows:

1. We analyze and give the energy consumption formula of SDAGFL. Then, we give the optimization objective of energy efficient SDAGFL;
2. Based on the energy consumption optimization objective, we propose an energy-efficient SDAGFL(ESDAGFL) scheme to reduce the energy consumption of SDAGFL. We evaluate the performance of ESDAGFL on two datasets, and the evaluation results show that ESDAGFL can efficiently reduce energy consumption compared to SDAGFL.

The rest of the paper is organized as follows. In Section 2, we review and compare the art of the federated learning framework based on DAG-DLT and show the advantages of SDAGFL over other frameworks. In Section 3, we formulate the energy consumption and give the optimization objective of energy efficient SDAGFL. We propose an energy efficient optimization scheme of SDAGFL in Section 4. Numerical simulation results are presented in Section 5. Finally, we summarize the work and give possible future researches.

2. Related Work

In this section, we will introduce and compare four federated learning frameworks based on the Tangle. Then, we analyze the advantages of SDAGFL compared to other works. Finally, we introduce the SDAGFL in detail.

Recently, the Tangle, which features high TPS, low energy usage, and decentralization, has received extensive attention from researchers. There are four Tangle-based federated learning frameworks that have been proposed as far as we know. Cao et al. [20] deployed a federated learning framework with Tangle called DAG-FL. DAG-FL includes three layers: the federated learning layer, the DAG layer, and the application layer, and it implements federated learning with asynchronous training and anomaly detection features. Shuo Yuan et al. [21] proposed ChainsFL, a two-layer hierarchical federated learning framework combining the Tangle and blockchain. The ChainsFL consists of a sub-ledger, the Raft [22]-based Hyperledger Fabric [23], deployed on edge nodes, and a Tangle-based master ledger. The ChainsFL has overcome the disadvantages of massive resource consumption and limited throughput in traditional blockchain-based FL. In addition, Schmid Robert et al. discussed the basic applicability of the combination of federated learning and the Tangle in Ref. [24]. They assumed a long-standing open network for continuous learning and development. In this framework, the tip selection algorithm based on Monte Carlo Markov Chain in the Tangle is used to implement model selection. The averaged model is used to update the model. The evaluation of this framework shows high training accuracy and model-agnostic resistance against random poisoning and label-flipping attacks. Beilharz et al. [25] further optimized the framework in Ref. [24]. They proposed an implicit model specialization framework for federated learning called Specializing DAG Federated Learning (SDAGFL). In SDAGFL, the implicit clustering of clients with a similar local dataset is achieved through the accuracy-biased random walk on DAG-DLT and the Fedavg algorithm. We compare these four frameworks in Table 1 from four different aspects.

Table 1. Comparison of four Federated Learning Frameworks based on Tangle DAG-DLT.

Framework	Decentralized	Robust for Poisoning Attack	Asynchronous Training	Individualized Model
ChainsFL [21]	✓	✓	×	×
DAG-FL [20]	✓	✓	✓	×
Learning-Tangle [24]	✓	✓	✓	×
SDAGFL [25]	✓	✓	✓	✓

Table 1 shows that SDAGFL is better than the three other works, and it sufficiently uses the DAG data structure's feature, and an accuracy-biased random walk for model update realized implicit clustering of the client. SDAGFL can adapt device and data heterogeneity, resist poisoning attacks, and support creating personal models. Thus, we think it is worthy of further study. However, from the prototype implementation of SDAGFL, we notice that the training process of SDAGFL is quite energy consuming because the client needs to compute the confidence and rating of the nodes selected by multiple random walks by traveling the ledger with 15–25 depth to obtain the "reference model" to judge whether or not to broadcast the newly trained model. In this paper, we will analyze the energy consumption of SDAGFL.

In particular, if not specified, the DAG-DLT in this work refers to the Tangle, and SDAGFL refers to the framework proposed in Ref. [25].

3. Optimization Objective of Energy Efficient SDAGFL

In this section, we will analysis the energy consumption of SDAGFL and give the energy optimization objective. The main notations used in this section is summarized in Table 2.

Table 2. Notations adopted for the energy consumption analysis.

Notations	Meaning
d	The depth of the Tangle ledger
c	The average number of the site node's children
f_i	The CPU-cycle frequency of the $client_i$
C_i	The CPU chipset's effective capacitance coefficient of the $client_i$
f_i^w	Number of CPU cycles required while the $client_i$ tests the model's accuracy
f_i^a	Number of CPU cycles required while the $client_i$ averages the model
f_i^t	Number of CPU cycles required while the $client_i$ trains one sample of dataset
f_i^c	Number of CPU cycles required while the $client_i$ compares the model
f_i^r	Number of CPU cycles required while the $client_i$ performs the random walk
E_i^{tip}	Energy consumption when the $client_i$ obtains two tip nodes
E_i^{agg}	Energy consumption when the $client_i$ averages the model
E_i^{train}	Energy consumption when the $client_i$ trains the model
$E_i^{reference}$	Energy consumption when the $client_i$ obtains and compares with the reference model
ω	Model parameter
$l(x_j, y_j, \omega)$	The loss function on data point (x_j, y_j)
$f(\omega)$	The global objective optimization in each clustered community
$F_n(\omega)$	The objective optimization in client n
N	The number of clients in each community
b	Size of the local batch
B	Number of the local batch
τ	Number of training epoch

3.1. Principle and the Workflow of SDAGFL

Before analysis of the energy consumption of SDAGFL, we first introduce the basic principle of SDAGFL. The basic principle of SDAGFL is shown in Figure 1. All the clients share a public DAG ledger in SDAGFL. The nodes in SDAGFL are divided into four categories, each containing a complete model parameter.

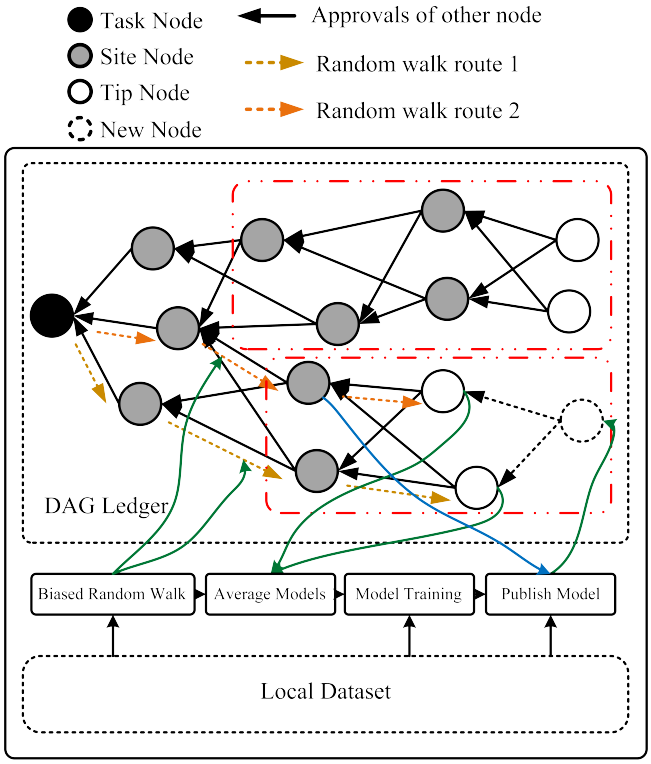

Figure 1. Basic principle of SDAGFL.

1. **Task Node:** The task node is a genesis node. It contains the basic training task and initial model parameter;
2. **Site Node:** The site node is the basic node type. It contains the model published by the clients;
3. **Tip Node:** The tip node is a special site node, which is not approved by other node;
4. **New Node:** The new node contains the model that will be published.

During the process of the SDAGFL, the clients will repeat the following five steps to update the model parameter:

1. Each client performs two accuracy-biased random walks in Algorithm 1 based on the local dataset until reaching the tip node. We show the random walk route using the yellow dashed line in Figure 1;
2. The client then averages the models in two tips obtained from Step 1;
3. Next, the client trains the average above model using its local dataset, and then obtains a new model;
4. In this step, the client will perform several accuracy-biased random walks to obtain the "reference model"(the site node approved by the blue line in Figure 1);
5. Finally, the client will compare the new model with the "reference model" to determine whether or not to publish the new model. If the test loss using the local data of new model is lower than the "reference model", the client will publish the new model to the public DAG ledger.

With the increase of the DAG ledger, the site node trained by the clients with independent identical distributed data will be clustered in the same group. As we can see, the site nodes in Figure 1 are clustered into two groups (red dashed box).

Algorithm 1: Random Walk of SDAGFL.

```
children ← GetChildren(Site Node n);
n = len(children);
initial accuracy[n];
for child[i] in children do
    accuracy[i] = EvaluateOnLocalData(child[i]);
end
for accuracy[i] in accuracy do
    normalized[i] = (accuracy[i] − max(accuracy)) / (max(accuracy) − min(accuracy));
    weight[i] = e^normalized × α ;
end
// WeightChoice is a weight-based child node selection function
nextnode = WeightChoice(weight) ;
RandomWalk(nextnode);
```

3.2. Energy Consumption Optimization Objective Analysis

In this section, we will give the optimization objective of energy efficient SDAGFL through analysis of the energy consumption and training target.

According to Section 3.1, the training process in SDAGFL can be divided into four parts: client obtains two tip nodes, Model Aggregation, Model Training, and client obtains and compares with the reference model. According to the analysis in the literature [19,26,27], we can analyze the energy consumption as follows.

Client obtains two tip nodes: The energy consumption of this process is decided by the depth of the ledger and the number of children nodes of the site node. We assumption that the each client i performs the two independently random walks on the ledger with the d depth and c average number of children. The energy consumption of the process in which the client i obtains two tip nodes is shown as follows.

$$E_i^{tip} = C_i \times 2 \times dcf_i^w \times f_i^2 \tag{1}$$

Client aggregates the model in tip node: In this process, the number of CPU cycles for the client i to aggregate the model is f_i^a. Thus, the energy consumption in model aggregation is

$$E_i^{aggregation} = C_i \times f_i^a \times f_i^2 \tag{2}$$

Client trains the model: The energy cost of model training depends on the size of the dataset. The number of CPU cycles for training one sample of the dataset is defined as f_i^t, and the training batch size and number of batch is defined as b and B. The training epoch is defined as τ. The energy consumption can be expressed as

$$E_i^{training} = C_i \times bB\tau f_i^t \times f_i^2 \tag{3}$$

Client obtains and compares with the reference model: In the prototype implementation of SDAGFL (https://github.com/osmhpi/federated-learning-dag, accessed on 6 September 2022), the client needs to compute the confidence and rating of each site node to obtain the reference model. The confidence is the number of the site node selected during the multiple random walks. The rating is the number of nodes that are directly and indirectly approved by every site node. Thus, the energy cost of this process contains two aspects: confidence computing and rating computing. Thus, the energy consumption of the client obtaining the "reference model" can be expressed as follows.

$$E_i^{reference} = C_i \times r \times d \times c \times (f_i^w + f_i^c + f_i^r) \times f_i^2 \tag{4}$$

where r is the number of random walk, which is 5 in SDAGFL.

Through the above analysis, we can get the energy consumption by the client i for one time model update as follows:

$$\begin{aligned} E_i^{total} &= E_i^{tip} + E_i^{aggregation} + E_i^{training} + E_i^{reference} \\ &= C_i \times \{2dcf_i^w + f_i^a + bB\tau f_i^t + rdc(f_i^w + f_i^c + f_i^r)\} \times f_i^2 \\ &= C_i \times \{(2+r)dcf_i^w + f_i^a + bB\tau f_i^t + rdc\{f_i^c + f_i^r)\} \times f_i^2 \end{aligned} \tag{5}$$

According to Equation (5), energy can be divided into four parts: energy consumption of $(2 + r)$ times random walk, energy consumption of model aggregation, energy consumption of model training, and energy consumption of computing reference and rating. Although reducing the walk depth d to a fixed depth of 15 to 25 transactions from the tip nodes is beneficial to reduce the energy consumption, this process must be repeatedly executed multiple times, resulting in massive energy consumption. The energy consumption of model aggregation is decided by the model's size, which is a fixed parameter in a specific federated learning task. In addition, the energy consumption of model training is connected to the training batch. To be fair and simplify the problem, we let each device execute the same batch. So, model aggregation and training energy consumption can be regarded as fixed energy. Therefore, the energy consumption should be optimized, which can be expressed in Equation (6) as follows:

$$\mathbb{E} = C_i \times \{(2+r)dcf_i^w + rdc\{f_i^c + f_i^r)\} \times f_i^2 \tag{6}$$

In addition, like general federated aggregation algorithms such as FedAvg, we use the same suggested supervised objective function. The clients with the same data distribution will be clustered in multiple implicit communities. For every cluster, we assume that the number of clients in each community is N, that the nth client has k data points $|k| = m_n$, and that the size of the whole dataset is m. Therefore, the optimization objective of federated training is defined as:

$$\min_{\omega \in \mathbb{R}^d} f(\omega) \tag{7}$$

where

$$f(\omega) = \sum_{n=0}^{N} \frac{m_n}{m} \times F_n(\omega) \tag{8}$$

and

$$F_n(\omega) = \frac{1}{m_n} \sum_{j}^{m_n} l_j(x_j, y_j, \omega) \tag{9}$$

Thus, the energy consumption optimization objective of SDAGFL is formulated as follows:

$$Minimize[\mathbb{E}, f(\omega)], \quad where \quad \omega \in \mathbb{R}^d \tag{10}$$

In other words, the optimization objective can be expressed as minimizing the energy consumption of the client while minimizing the federated learning training loss function.

4. Energy Efficient Optimization Scheme of SDAGFL

Event-triggered communication mechanism is proposed for the network control systems to reduce the frequency of communication [28,29]. In an event-triggered mechanism-based system, the device will perform the predetermined operation only when it detects an event that meets the trigger conditions.

The event-triggered communication mechanism has recently been introduced into the field of parallel machine learning. In a event-triggered based machine learning scenario, the client computes the model's gradient and broadcasts the gradients that meet the trigger threshold to the neighbors [30–34].

Inspired by above researches, we utilize an event-triggered communication based on the parameter change to solve the optimized objective in Equation (10). The workflow of

ESDAGFL is shown in Figure 2. Compared to the baseline, the event-triggered communication mechanism only needs to judge the change of the model's norm, and does not need to find a reference model. Therefore, ESDAGFL can reduce the energy consumption of the client and the simulation results in Section 5 also demonstrate the advantages of the proposed scheme.

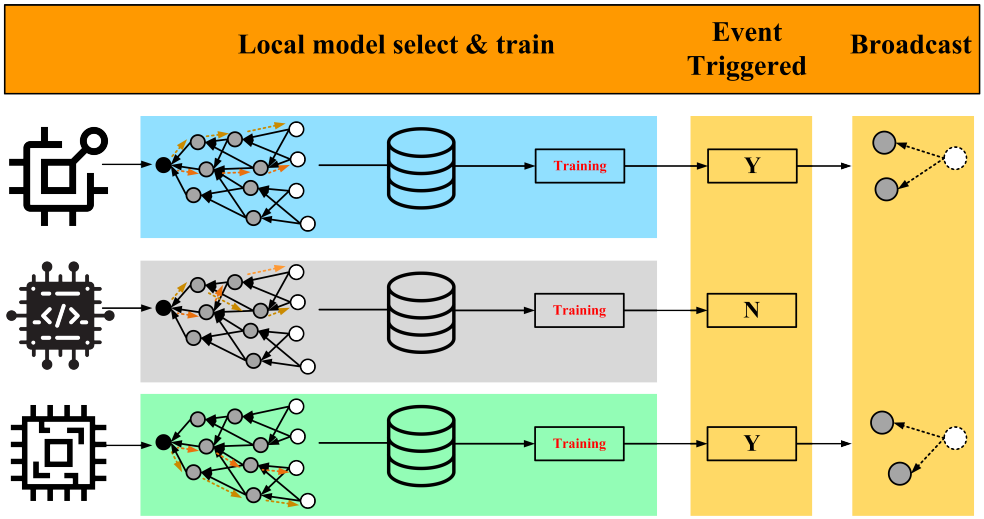

Figure 2. Workflow of the Energy Efficient SDAGFL Scheme.

The clients participating in the federated learning have the following features:

- **Data are private to the clients:** In ESDAGFL, the dataset is stored in the spatially distributed clients, and it is private to all clients. There is no data exchange among the clients;
- **Data are Non-IID for all clients:** Data is non-independent and identically distributed (Non-IID) among all clients participating in the ESDAGFL;
- **Data are IID for clients in the same cluster:** The site node published by the clients with a similar data distribution feature could be clustered into a "implicit community". So, we can regard that the data distribution among the clients clustered in the same community is an independently identical distribution;
- **Honest and Malicious client's behavior:** The honest clients always comply with the rules of the SDAGFL and use their complete local datasets to participate in the training. Once the trained model meets the broadcast condition, the client will broadcast it in time. On the contrary, the malicious clients do not comply with any rules;
- **Synchronicity:** Although the ledger is not synchronized in the vast majority of the reality cases, we think that the performance of the ESDAGFL would not be affected whether the ledger is synchronous or not. The reason is that whether the client publishes the new model or not only depends on the average model and the new trained model. Thus, we define that the ledger on each client is synchronized in real-time to analyze the performance of ESDAGFL for convenience.

In our ESDAGFL framework, each client participates in the following federated learning executed steps, as shown in Algorithm 2.

Firstly, each clients performs two random walks and obtains the aggregated model $\omega_{avg} = Avg(\omega_{tip1}, \omega_{tip2})$ with two tip nodes. (Lines 1 and 2 in Algorithm 2).

Then, the client n uses a Stochastic Gradient Descent (SGD) to compute the local gradient and perform τ local model parameter updates using the local dataset to train the new model ω_{new}. This process can be expressed as Equation (11). (Lines 3–9 in Algorithm 2).

$$\omega_{new} = \omega_{avg} - \eta \sum^{\tau} \sum^{B} \nabla l(w_{avg}) \tag{11}$$

Next, the client computes the model parameter change rate between the trained new model and the input averaged model by Equation (12). (Lines 10 and 11 in Algorithm 2).

$$\Delta = \frac{||\Delta\omega||_2}{||\omega_{avg}||_2} \tag{12}$$

where,

$$||\Delta\omega|| = ||\omega_{new} - \omega_{avg}|| \tag{13}$$

Finally, the client broadcasts the trained new model ω_{new} if the model parameter change rate is equal to or greater than the trigger threshold. (Lines 12–14 in Algorithm 2)

$$\Delta \geq trigger_{threshold} \tag{14}$$

Algorithm 2: Energy efficient SDAGFL scheme.

Input: learning rate η , local batch is B, τ is the number of local epochs
Output: New Model:ω_{new}

1 (ω_1,ω_2)←Random Walk of SDAGFL of client;
2 $\omega_{avg} = (\omega_1 + \omega_2)/2$;
3 $\omega_{i,j} \leftarrow \omega_{avg}$;
4 **for** *local epoch* $i = 0, 1, 2, ..., \tau$ **do**
5 **for** *batch* $j = 0, 1, 2, ..., B$ **do**
6 $\omega_{i,j+1} = \omega_{i,j} - \eta\nabla l(\omega_{i,j})$;
7 **end**
8 **end**
9 $\omega_{new} \leftarrow \omega_{\tau,B}$;
10 $\Delta\omega = \omega_{new} - \omega_{avg}$;
11 Compute $\Delta = \frac{||\Delta\omega||_2}{||\omega_{avg}||_2}$;
12 **if** $\Delta \geq threshold_{tirgger}$ **then**
13 return ω_{new} ;
14 **end**

5. Experiment and Results

In this section, we evaluate ESDAGFL in the framework (https://github.com/osmhpi/federated-learning-dag, accessed on 6 September 2022) proposed by Beilharz et al. [25] with PyTorch. The parameters of the evaluation platform used in the experiment are shown in Table 3.

Table 3. Parameters of the experiment platform.

Item	Parameter
OS	Ubuntu 22.04
CPU	Intel®Core™ i7-10700 @2.9 GHz × 16
RAM	32 GB

5.1. Experiment Setting

This section introduces the experimental setting of ESDAGFL. The datasets and models used in the experiment are described in Section 5.1.1, and the fixed training hyperparameters setting is shown in Section 5.1.2.

5.1.1. Datasets and Models

We evaluated ESDAGFL on two training tasks. We evaluated a handwriting recognition task on the FMNIST-clustered dataset and a next character prediction task on the Poets dataset. The training and test datasets have a ratio of 9:1 for each client.

- **Handwriting Recognition Task:**
  - **Dataset:** The FMNIST-clustered dataset is a synthetically clustered version of Federated extended MNIST (FMNIST) built by LEAF [35]. In this dataset, the 28×28 pixel handwriting dataset is divided into 3 disjoint classes: [0,1,2,3], [4,5,6], [7,8,9], and the number of clients is the same in every class;
  - **Model:** For the handwriting recognition task, a Convolutional Neural Net (CNN) is used. It contains two convolution layers and two fully connected layers. The kernel size of the convolution layer is 5 with the RELU activation function. Each convolution layer is followed by a max pooling layer with pool size and stride length of 2. The two fully connected layers contain 2048 and 10 neurons, respectively.
- **Next Character Prediction Task:**
  - **Dataset:** The Poets dataset is used to evaluate the performance of ESDAGFL for the next character prediction task. The Poets dataset consists of an English dataset from William Shakespeare's works and a German dataset from Goethe's plays. The English and German datasets have an equal number of samples and are put into different clusters.
  - **Model:** For the next character prediction task, the model first maps each character to an embedding of dimension 8, calculated from the 80 character sequence. Then the model passes each character through a Long Short-Term Memory (LSTM) consisting of 2 layers with 256 units each. Finally, there is a dense layer for prediction.

5.1.2. Training Hyperparameters Setting

The training hyperparameters setting is shown in Table 4.

Table 4. Training Hyperparameters Setting of the Experiment.

Parameters	FMNIST-Clustered	Poets
Local epochs	1	1
Local batches	10	200
Batch size	10	10
Learning rate	0.05	0.8
Clients per round	10	10

5.2. Experimental Results

Different from the traditional FL, ESDAGFL is a fully asynchronous federated learning framework without any central server. It realizes the training target through each client run of the training process continuously and independently as long as its resources permit. Therefore, the concept of the rounds is introduced to enable a better demonstration of the experimental results.

In the experiment, the $threshold_{trigger}$ is set to 0.008 for the FMNIST-clustered dataset and 0.12 for the Poets datasets. The evaluation of the training accuracy and loss was done every five rounds using the test dataset with 5% of all clients randomly selected. We regard the work in Ref. [25] as the baseline, and we evaluated our ESDAGFL on three metrics: training accuracy and loss, implicit specialization, and training time cost.

5.2.1. Training Accuracy and Loss Evaluation

Training accuracy and loss are two standard metrics used to evaluate the model performance. The accuracy is the percentage of correct predictions, and the loss is the cross-entropy loss.

Firstly, we evaluated the average accuracy and loss on the test dataset for the handwriting recognition task. The evaluation results are shown as Figures 3 and 4.

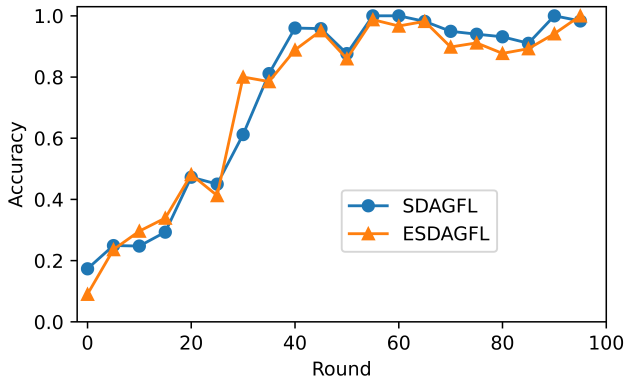

Figure 3. The average training accuracy of ESDAGFL and SDAGFL evaluated on the FMNIST-clustered dataset.

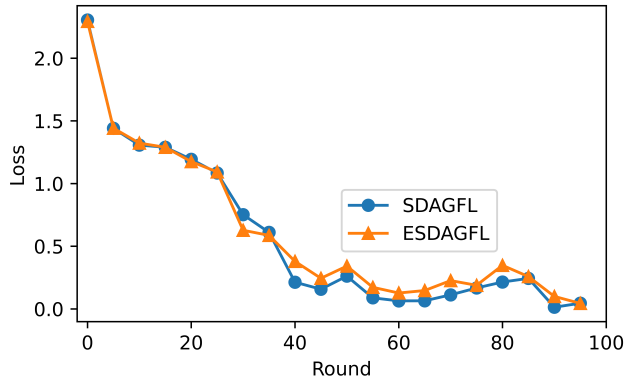

Figure 4. The average training loss of ESDAGFL and SDAGFL evaluated on the FMNIST-clustered dataset.

The evaluation result shows that the ESDAGFL achieves similar accuracy and approximate loss as the baseline. The experimental results demonstrate the applicability of ESDAGFL to the written recognition task.

Then, we evaluated the next character prediction task on the Poets dataset to further illustrate the applicability of ESDAGFL for different models. Figures 5 and 6 show ESDAGFL's average accuracy and loss in the Poets dataset. The experiment results show that ESDAGFL equally applies to the Poets dataset.

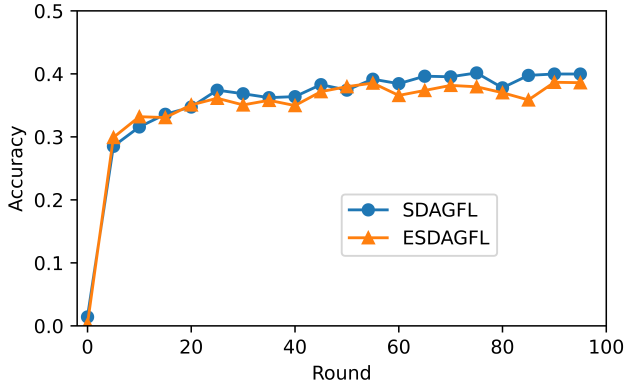

Figure 5. The average training accuracy of ESDAGFL and SDAGFL evaluated on the Poets dataset.

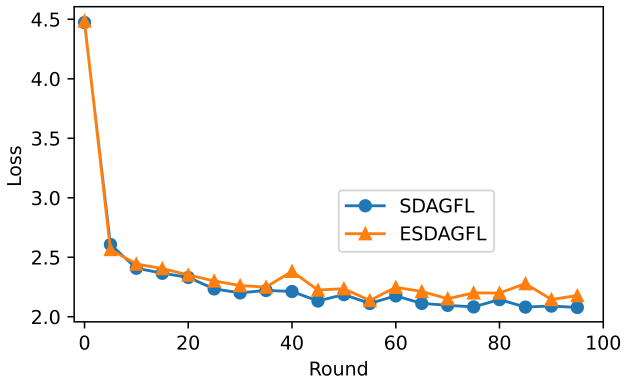

Figure 6. The average training loss of ESDAGFL and SDAGFL evaluated on the Poets dataset.

Convergence analysis of ESDAGFL: As the new models are only published if they are changed when the honest clients' training for the model is positive, the federated training will converge to an expected direction. Figures 3 and 5 show the convergence rate of ESDAGFL on the FMNIST-clustered and Poets datasets. Multiple experiment results show that the training process converges to nearly the same result. It confirmed that ESDAGFL could converge to a reasonable training accuracy.

5.2.2. Implicit Specialization Evaluation

In this part, we evaluated the implicit specialization of the ESDAGFL. There are three metrics to evaluate the implicit specialization of the ESDAGFL. The first is the modularity $m \in [-\frac{1}{2}, 1]$, which is a metric of the community segmentation in the community discovery algorithm. The more closely connected the nodes within a community and the more sparsely the connections between the communities are, the greater the modularity is. The second metric is the number of modules, which indicates the partitioning of all clients. The number of modules should be an appropriate value. The final metric is approval pureness, which measures the probability that a client approves the nodes published by other clients in the same cluster.

Firstly, we evaluated the modularity and number of modules on the FMNIST-clustered dataset. The FMNIST-clustered dataset was chosen because it has more classes and is easier to visualize.

Figures 7 and 8 show the performance of ESDAGFL in the implicit specialization. Figure 7 shows that we achieved similar modularity as the baseline on the FMNIST-clustered dataset. As shown in Figure 8, with the increase of time, we achieve the same number of modules as the baseline and almost all of the clients are clustered into the community relative to their label.

To quantify the robustness of the DAG specialized in these experiments, we test the approval pureness to quantify the "specialized" ESDAGFL on the FMNIST-clustered and Poets datasets, and the result is shown in Table 5. The base pureness refers to the approval pureness if the approvals were to be randomly spread over all clusters. Since there are three clusters in the FMNIST-clustered dataset and two clusters of Goethe and Shakespeare in Poets, their basic pureness is 0.33 and 0.5, respectively. ESDAGFL and the baseline achieves 100% pureness in the FMNIST-clustered data set, and all model approvals are from the same cluster. Although the approval pureness of ESDAGFL on the Poets dataset is lower than the FMNIST-clustered dataset, the pureness is still high. The approval pureness on the Poets dataset show the balance between specialization and generalization.

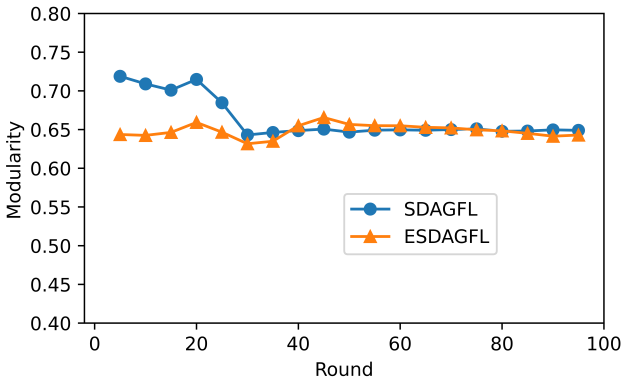

Figure 7. The modularity variation in ESDAGFL and SDAGFL evaluated on the FMNIST-clustered dataset.

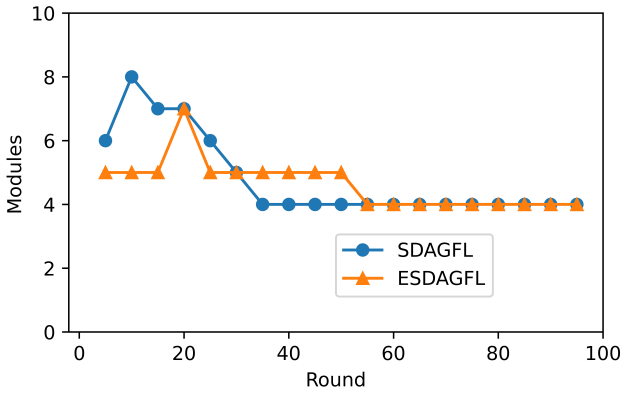

Figure 8. The modules variation in ESDAGFL and SDAGFL evaluated on the FMNIST-clustered dataset.

Table 5. Comparison of the approval pureness after 100 rounds of training between ESDAGFL and baseline.

Dataset	Base Pureness	Pureness of SDAGFL (Baseline)	Pureness of ESDAGFL (Ours)
FMNIST-clustered	0.33	1	1
Poets	0.5	0.98	0.96

The above experiments show that ESDAGFL can achieve the same implicit specialization as the Baseline (SDAGFL) for the different datasets. However, in some metrics of the implicit specialization, the performance is not as good as the baseline, which has a negligible effect for the implicit specialization.

5.2.3. Energy Consumption Evaluation

The experiments in Sections 5.2.1 and 5.2.2 show that ESDAGFL realizes the balance between the training performance and specialization on the FMNIST-clustered and Poets dataset. In this part, we will evaluate ESDAGFL's energy consumption using "pyJoules" (https://github.com/powerapi-ng/pyJoules, accessed on 6 September 2022) [36,37].

The "pyJoules" is a software-defined power meter that can measure the power consumption of the program running on a host machine. It uses the Intel "Running Average Power Limit" (RAPL) technology to monitor the energy consumption of the Intel CPU socket package, RAM, and Intel integrated GPU.

The devices that "pyJoules" can monitor match the RAPL domain, as shown in Figure 9.

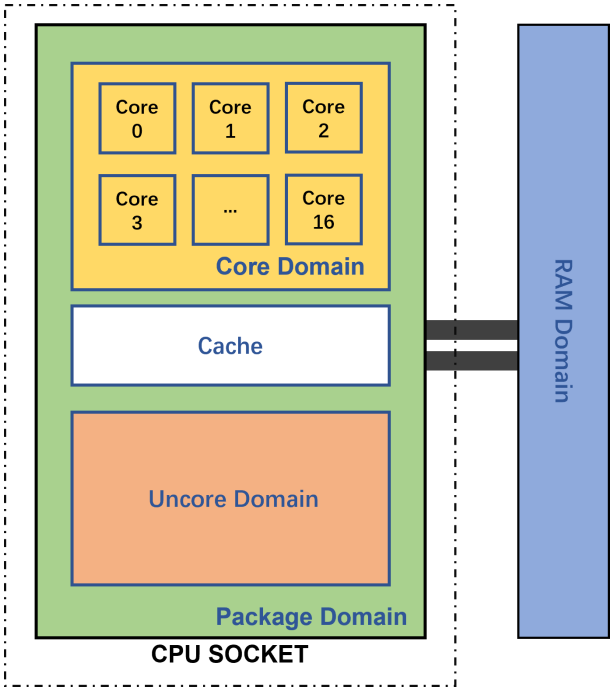

Figure 9. RAPL domain match part of our CPU socket. The energy consumption of the core measured by pyJoules is the sum of the CPU core energy consumption, the energy consumption of the packages is the wall CPU energy consumption, and the energy consumption of the uncore is the energy consumption of the integrated GPU.

In our simulation experiments, we took the statistics of the core's energy consumption per round of training multiple times and removed the bad values to get the average results.

Figures 10 and 11 show the core's energy consumption per round with 10 clients and when the client performs the random walk from the depth of 15 to 25. The energy consumption of ESDAGFL is lower than the baseline. The total CPU core's energy consumption of ESDAGFL is 61.29 KJ, and the baseline's core's energy consumption is 106.74 KJ. Our ESDAGFL scheme can reduce the 42.5% energy consumption compared with the baseline for the handwriting recognition task. As for the Poets dataset, the total CPU core's energy consumption of ESDAGFL is 2390.41 KJ, and the baseline's core's energy consumption is 4956.63 KJ. Our ESDAGFL scheme can reduce the 51.7% energy consumption for the next character prediction task.

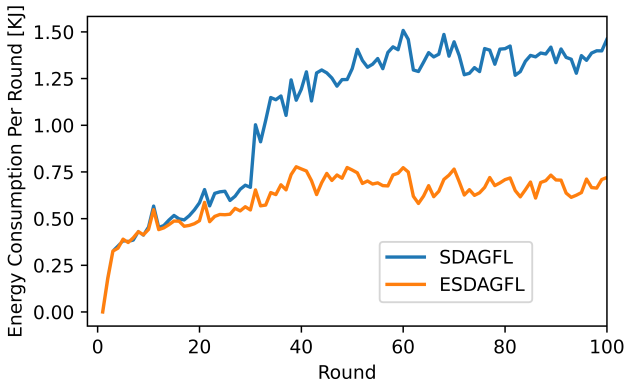

Figure 10. Energy consumption in every round evaluated on the FMNIST-clustered dataset.

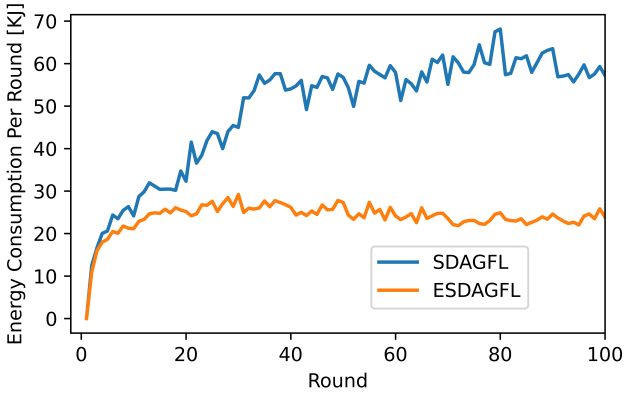

Figure 11. Energy consumption in every round evaluated on the Poets dataset.

We evaluate the energy consumption of ESDAGFL and baseline on the FMNIST-clustered dataset with a different client for each round. The result is shown in Table 6.

Table 6. Energy consumption [KJ] with a different number of clients per round evaluated on the FMNIST-clustered dataset.

Scheme	15 Clients per Round	20 Clients per Round	25 Clients per Round
SDAGFL (Baseline)	216.37	362.82	601.01
ESDAGFL (Ours)	108.35	166.21	220.42
Energy consumption reduction	47.6%	54.1%	63.3%

5.2.4. Results Analysis

We evaluate the performance of our approach on the FMNIST-clustered and Poets datasets. The experimental results show that our scheme can realize the same implicit specialization and generalization as the SDAGFL. Furthermore, compared with the SDAGFL, our scheme can reduce the energy consumption by 42.5% and 51.7% for the handwriting recognition task on the FEMNIST dataset and the character prediction task on the Poets dataset, respectively, on the condition that 10 clients participate in each round. Besides, the reduction of energy consumption increases with the number of clients.

6. Conclusions

Energy consumption is an important issue for federated learning in IoT scenarios, as many devices are battery powered. To reduce the energy consumption of the SDAGFL and make it a better federated learning scheme for IoT, the energy optimization objective of SDAGFL is constructed based on the thorough analysis of energy consumption of SDAGFL. To realize the optimization objective, we design an event-triggered communication-based SDAGFL scheme, named ESDAGFL. In ESDAGFL, the client broadcasts the trained new model only in the specific event in which the new model is significantly different form the previous one. It is necessary to search for a "reference model" by traveling the whole ledger to judge whether a new model is broadcasted or not in SDAGFL. Our simulations are performed on a platform with Intel®Core™ i7-10700 CPU. The simulation results show, that compared with SDAGFL, our scheme can reduce the energy consumption by 42.5% and 51.7% on the FMNIST-clustered dataset and the Poets dataset, respectively. Besides, the energy reduction increases along with the increase of the number of clients per round. Meanwhile, our scheme can reach an equal balance between the model's performance and specialization as SDAGFL. In the future, we plan to deploy the ESDAGFL in a practical scenario and perform an overall test. In addition, we will not only consider the energy consumption caused by computation, but also the communication.

Author Contributions: Funding acquisition, Q.L.; Investigation, Q.L. Methodology, X.X.; Resources, H.M. and Q.L.; Validation, X.X.; Writing—original draft, X.X. and H.M.; Writing—review and editing, X.X., Q.L., F.H. and A.A.A.E.-L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China (grant number 62071151).

Data Availability Statement: Not applicable.

Acknowledgments: Thanks to authors of article "Implicit Model Specialization through DAG-based Decentralized Federated Learning" for the open source simulation code in https://github.com/osmhpi/federated-learning-dag (accessed on 6 September 2022).

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

CNN	Convolutional Neural Network
DAG	Directed Acyclic Graph
DLT	Distributed Ledger Technology
ESDAGFL	Event-Triggered Specialized DAG Federated Learning
FL	Federated Learning
FMNIST	Federated extended MNIST
IoT	Internet of Things
RAPL	Running Average Power Limit
LSTM	Long Short-Term Memory
SDAGFL	Specialized DAG Federated Learning
SGD	Stochastic Gradient Descent
TPS	Transactions Per Second

References

1. Albrecht, J.P. How the GDPR Will Change the World. *Eur. Data Prot. Law Rev.* **2016**, *2*, 287–289. [CrossRef]
2. Yi, S. Personal Information Protection: China's Path Choice. *US-China Law Rev.* **2021**, *18*, 227. [CrossRef]
3. Konečný, J.; McMahan, H.B.; Ramage, D.; Richtárik, P. Federated Optimization: Distributed Machine Learning for On-Device Intelligence. *arXiv* **2016**, arXiv:1610.02527.
4. Niknam, S.; Dhillon, H.S.; Reed, J.H. Federated Learning for Wireless Communications: Motivation, Opportunities, and Challenges. *IEEE Commun. Mag.* **2020**, *58*, 46–51. [CrossRef]
5. Chen, M.; Gündüz, D.; Huang, K.; Saad, W.; Bennis, M.; Feljan, A.V.; Poor, H.V. Distributed Learning in Wireless Networks: Recent Progress and Future Challenges. *IEEE J. Sel. Areas Commun.* **2021**, *39*, 3579–3605. [CrossRef]
6. Xu, C.; Qu, Y.; Xiang, Y.; Gao, L. Asynchronous Federated Learning on Heterogeneous Devices: A Survey. *arXiv* **2022**, arXiv:2109.04269.
7. Zhao, Y.; Li, M.; Lai, L.; Suda, N.; Civin, D.; Chandra, V. Federated Learning with Non-IID Data. *arXiv* **2018**, arXiv:1806.00582.
8. Cao, D.; Chang, S.; Lin, Z.; Liu, G.; Sun, D. Understanding Distributed Poisoning Attack in Federated Learning. In Proceedings of the 2019 IEEE 25th International Conference on Parallel and Distributed Systems ICPADS), Tianjin, China, 4–6 December 2019; pp. 233–239. [CrossRef]
9. Kim, H.; Park, J.; Bennis, M.; Kim, S.L. Blockchained On-Device Federated Learning. *IEEE Commun. Lett.* **2019**, *24*, 1279–1283. [CrossRef]
10. Wang, P.; Zhao, Y.; Obaidat, M.S.; Wei, Z.; Qi, H.; Lin, C.; Xiao, Y.; Zhang, Q. Blockchain-Enhanced Federated Learning Market with Social Internet of Things. *IEEE J. Sel. Areas Commun.* **2022**, 3213314. [CrossRef]
11. Miri Rostami, S.; Samet, S.; Kobti, Z. A Study of Blockchain-Based Federated Learning. In *Federated and Transfer Learning*; Razavi-Far, R., Wang, B., Taylor, M.E., Yang, Q., Eds.; Adaptation, Learning, and Optimization; Springer: Cham, Switzerland, 2023; pp. 139–165. [CrossRef]
12. Chen, S.; Wang, X.; Zhou, P.; Wu, W.; Lin, W.; Wang, Z. Heterogeneous Semi-Asynchronous Federated Learning in Internet of Things: A Multi-Armed Bandit Approach. *IEEE Trans. Emerg. Top. Comput. Intell.* **2022**, *6*, 1113–1124. [CrossRef]
13. Xu, X.; Duan, S.; Zhang, J.; Luo, Y.; Zhang, D. Optimizing Federated Learning on Device Heterogeneity with A Sampling Strategy. In Proceedings of the 2021 IEEE/ACM 29th International Symposium on Quality of Service (IWQOS), Tokyo, Japan, 25–28 June 2021; pp. 1–10. [CrossRef]

14. Cao, J.; Lian, Z.; Liu, W.; Zhu, Z.; Ji, C. HADFL: Heterogeneity-aware Decentralized Federated Learning Framework. In Proceedings of the 2021 58th ACM/IEEE Design Automation Conference (DAC), San Francisco, CA, USA, 5–9 December 2021; pp. 1–6. [CrossRef]
15. Li, G.; Hu, Y.; Zhang, M.; Liu, J.; Yin, Q.; Peng, Y.; Dou, D. FedHiSyn: A Hierarchical Synchronous Federated Learning Framework for Resource and Data Heterogeneity. *arXiv* **2022**, arXiv:2206.10546.
16. Huang, W.; Ye, M.; Du, B. Learn from Others and Be Yourself in Heterogeneous Federated Learning. In Proceedings of the 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 18–24 June 2022; pp. 10133–10143. [CrossRef]
17. Popov, S. The Tangle. White Paper. 2018. Version 1.4.3. Available online: http://www.descryptions.com/Iota.pdf (accessed on 6 September 2022).
18. Wright, D.C.S. Bitcoin: A Peer-to-Peer Electronic Cash System. 2008. Available online: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3440802 (accessed on 6 September 2022).
19. dos Anjos, J.C.S.; Gross, J.L.G.; Matteussi, K.J.; González, G.V.; Leithardt, V.R.Q.; Geyer, C.F.R. An Algorithm to Minimize Energy Consumption and Elapsed Time for IoT Workloads in a Hybrid Architecture. *Sensors* **2021**, *21*, 2914. [CrossRef]
20. Cao, M.; Zhang, L.; Cao, B. Toward On-Device Federated Learning: A Direct Acyclic Graph-Based Blockchain Approach. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, 1–15. [CrossRef]
21. Yuan, S.; Cao, B.; Peng, M.; Sun, Y. ChainsFL: Blockchain-driven Federated Learning from Design to Realization. In Proceedings of the 2021 IEEE Wireless Communications and Networking Conference (WCNC), Nanjing, China, 29 March–1 April 2021; pp. 1–6. [CrossRef]
22. Ongaro, D.; Ousterhout, J. In Search of an Understandable Consensus Algorithm. In Proceedings of the 2014 USENIX Annual Technical Conference (USENIX ATC 14), Philadelphia, PA, USA, 19–20 June 2014; pp. 305–319.
23. Androulaki, E.; Barger, A.; Bortnikov, V.; Cachin, C.; Christidis, K.; De Caro, A.; Enyeart, D.; Ferris, C.; Laventman, G.; Manevich, Y.; et al. Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains. In Proceedings of the 13th EuroSys Conference, EuroSys'18, Porto, Portugal, 23–26 April 2018; Association for Computing Machinery: New York, NY, USA, 2018; pp. 1–15. [CrossRef]
24. Schmid, R.; Pfitzner, B.; Beilharz, J.; Arnrich, B.; Polze, A. Tangle Ledger for Decentralized Learning. In Proceedings of the 2020 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), New Orleans, LA, USA, 8–22 May 2020; pp. 852–859. [CrossRef]
25. Beilharz, J.; Pfitzner, B.; Schmid, R.; Geppert, P.; Arnrich, B.; Polze, A. Implicit Model Specialization through Dag-Based Decentralized Federated Learning. In Proceedings of the 22nd International Middleware Conference, Middleware'21, Québec City, QC, Canada, 6–10 December 2021; Association for Computing Machinery: New York, NY, USA, 2021; pp. 310–322. [CrossRef]
26. Tran, N.H.; Bao, W.; Zomaya, A.; Nguyen, M.N.H.; Hong, C.S. Federated Learning over Wireless Networks: Optimization Model Design and Analysis. In Proceedings of the IEEE INFOCOM 2019—IEEE Conference on Computer Communications, Paris, France, 29 April–2 May 2019; pp. 1387–1395. [CrossRef]
27. Lu, Y.; Huang, X.; Zhang, K.; Maharjan, S.; Zhang, Y. Communication-Efficient Federated Learning and Permissioned Blockchain for Digital Twin Edge Networks. *IEEE Internet Things J.* **2021**, *8*, 2276–2288. [CrossRef]
28. Dimarogonas, D.V.; Frazzoli, E.; Johansson, K.H. Distributed Event-Triggered Control for Multi-Agent Systems. *IEEE Trans. Autom. Control* **2012**, *57*, 1291–1297. [CrossRef]
29. Lemmon, M. Event-Triggered Feedback in Control, Estimation, and Optimization. In *Networked Control Systems*; Bemporad, A., Heemels, M., Johansson, M., Eds.; Lecture Notes in Control and Information Sciences; Springer: London, UK, 2010; pp. 293–358. [CrossRef]
30. Ghosh, S.; Aquino, B.; Gupta, V. EventGraD: Event-triggered Communication in Parallel Machine Learning. *Neurocomputing* **2022**, *483*, 474–487. [CrossRef]
31. Nguyen, N.; Han, S. AET-SGD: Asynchronous Event-triggered Stochastic Gradient Descent. *arXiv* **2021**, arXiv:2112.13935.
32. George, J.; Gurram, P. Distributed Stochastic Gradient Descent with Event-Triggered Communication. *Proc. AAAI Conf. Artif. Intell.* **2020**, *34*, 7169–7178. [CrossRef]
33. George, J.; Gurram, P. Distributed Deep Learning with Event-Triggered Communication. *arXiv* **2019**, arXiv:1909.05020.
34. Kajiyama, Y.; Hayashi, N.; Takai, S. Distributed Subgradient Method With Edge-Based Event-Triggered Communication. *IEEE Trans. Autom. Control* **2018**, *63*, 2248–2255. [CrossRef]
35. Caldas, S.; Duddu, S.M.K.; Wu, P.; Li, T.; Konečný, J.; McMahan, H.B.; Smith, V.; Talwalkar, A. LEAF: A Benchmark for Federated Settings. *arXiv* **2019**, arXiv:1812.01097.
36. Bourdon, A.; Noureddine, A.; Rouvoy, R.; Seinturier, L. PowerAPI: A Software Library to Monitor the Energy Consumed at the Process-Level. *ERCIM News* **2013**, *92*, 43–44.
37. Georgiou, S.; Rizou, S.; Spinellis, D. Software Development Lifecycle for Energy Efficiency: Techniques and Tools. *ACM Comput. Surv.* **2019**, *52*, 1–33. [CrossRef]