Journal of **N**onlinear **A**nalysis and **O**ptimization
Vol. 14, Issue. 01 : 2023
ISSN : **1906-9685**

INTELLIGENT GRID-BASED FEDERATED LEARNING ON BLOCKCHAIN

#1KONTHAM SRIDHAR, Assistant Professor,
Department of Computer Science and Engineering,
#2Dr. N. PRABHAKARAN, Professor,
Department of Computer Science and Engineering
MOTHER THERESA COLLEGE OF ENGINEERING AND TECHNOLOGY, PEDDAPALLY, TS.

ABSTRACT: Prosumers' participation in demand-response programs is essential to the success of demand-side management in renewable energy infrastructure. People's concerns regarding the confidentiality of their personal energy data used for forecasting continue to act as a barrier to wider adoption of the technology. In this piece, we explore how blockchain-based distributed federated learning (FL) might be used to forecast future energy needs. This approach utilizes FL and blockchain technology to ensure the confidentiality and security of client energy data. Only the weights of locally learnt models are transmitted using the blockchain. Edge prosumer nodes are where sensitive energy data is stored. The worldwide federated technique assures that parameters cannot be modified and can be traced back to their original source by transferring and copying data over the blockchain overlay. We proposed using smart contracts to integrate local machine-learning prediction models with the blockchain, establish scaling functions for model parameters, and reduce network overhead. We use a multi-layer perceptron model and data from prosumers to evaluate centralized, local-edge, and blockchain-integrated algorithms for anticipating energy consumption 24 hours in advance. Even though centralized learning is superior at prediction, blockchain-based distributed FL that consistently protects user data is only marginally less accurate.

Keywords: energy prediction; federated learning; blockchain; smart grid management; demand response; smart contracts; machine learning

1.INTRODUCTION

The energy grid is decentralizing as intermittent renewable energy sources and energy storage proliferate at the system's periphery. Using demand-side management, an area may be better able to deal with energy fluctuations. It regulates electricity consumption via load scheduling performed by prosumers. Some examples are converters for electric vehicles, programmable thermostats, and smart home equipment. The effectiveness of a demand-response (DR) program is contingent on two factors: the number of people using the program and the expected daily energy consumption. For load-flexibility solutions to reliably deliver set points to many assets over an extended period of time, multiple energy prediction phases are required. Renewable energy sources have an inconsistent production, and the demand from smaller customers fluctuates often, making it difficult to predict future energy consumption. Currently, accurate one-step-ahead is used by the vast majority of energy prediction algorithms. When considering longer time periods, DR becomes significantly less precise. In the Internet of Things, smart energy meters produce vast amounts of data. Because of this, energy firms utilize cloud-based data storage, big data, and machine learning to forecast future energy consumption and output. Model uncertainty is reduced and forecast accuracy is improved by considering many temporal and energy scales (individuals, communities, etc.).

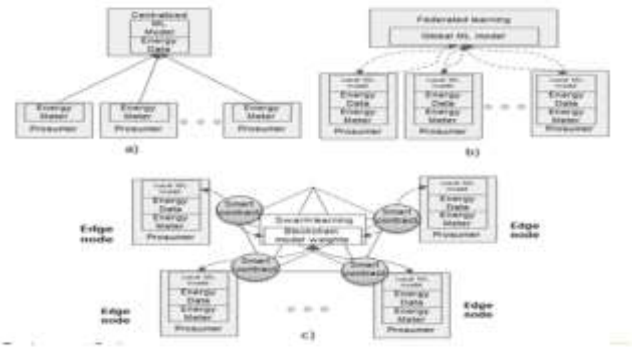

Figure1. Learning includes swarm (blockchain-based distributed FL), centralized, and federated.

Using extensive data about prosumers, deep learning models are trained to make predictions about the future. Models predict energy use and deliver messages to the running software to adjust the prediction (Figure 1a).

Combining energy characteristics with contextual variables, such as behavioral or social elements, non-energy vectors and features, makes it easier to create reliable predictions. Although the cloud-based scenario is more likely to occur, it raises privacy concerns. People don't participate in many DR projects because they're concerned about the security of their personal information. Researchers are exploring on measures to preserve privacy while tracking energy use, as this has caused several European countries to delay implementing smart meters and left consumers and families powerless to aid in DR. There have been initiatives to ensure secure two-way communication between utilities and prosumers, but centralization raises privacy concerns for users' data. Privacy-based ML is essential for energy forecasting because of the General Data Protection Regulation (GDPR) in Europe. Decentralized prediction is simplified by FL infrastructures that safeguard personal information (Figure 1b). The prosumer edge node archives information in order to preserve confidentiality and prevent leaks. Edge node data is used to update the global model, while prosumer websites are mined for local ML model training. At this time, only model parameters are being transmitted, and no personally identifiable data. When energy data is stored locally rather being transmitted across the network, latency and bandwidth costs are reduced, and prediction accuracy is enhanced.

Distributed ledger technology's transparency and user-friendliness can boost the authority of DR program oversight. Data provenance allows for a complete audit trail of all modifications made to any given link in the chain. Imperceptible chain information is immune to tampering by third parties. These characteristics ought to characterize distributed machine learning. The global FL case model can be kept in a distributed ledger accessible to the public. Customers get trust and comprehension as a result. Since data once stored cannot be altered, fraud is reduced. The public blockchain records updates to the global federated model.

New approaches to data privacy and trust protection using FL models and blockchain technology are emerging (Figure 1c), but they do not yet account for the energy needs of prosumers. We employ machine learning to anticipate the energy needs of local prosumers, while the blockchain records and updates the global energy forecast model. People replicate model parameters, update the federated model, and onboard new prosumers all through the usage of public blockchains. Participation from prosumers is facilitated by the elimination of concerns over data ownership and confidentiality thanks to the combination of public blockchain integration and state change model tracking.

An anonymous prediction of future energy consumption can be made via a blockchain-based distributed FL for energy. Global model parameters that cannot be modified are recorded and broadcast via a blockchain network overlay.

Regression-based energy forecast becomes more difficult as prosumer size and blockchain transaction cost scale, therefore smart contracts update global model parameters to account for these changes.

We evaluate the MLP models used to predict energy consumption by prosumers in three distinct environments: the distributed cloud, the local edge, and the blockchain.

Distributed methods of machine learning aim to improve both these aspects. FL solutions provide only a minimal measure of confidentiality. Centralized learning models might become vulnerable to attacks because of their centralization. Blockchain, FL, and distributed optimization are used to fix the smart grid's credibility, privacy, and security flaws. The remainder of this article will focus on FL solutions for smart energy grid security and distributed optimization methods. The advantages of distributed FL methods built on the blockchain over more conventional models are then discussed.

Their FL model revolutionized the field with "one-shot parameter averaging" via simultaneous stochastic gradient descent (SGD). The proposed architecture required one last communication round to generate the main model after the slow SGD optimizer was used to educate local models. The distribution of data among nodes and the averaging of a single parameter were not taken into account by other machine learning approaches. SVM, sparse logistic regression, basis pursuit, covariance selection, and distributed convex optimization using alternating direction multipliers on the lasso are all explored in Boyd and his team's work. Convergence was found to be less common while using longer communication cycles. Distributed Approximate Newton (DANE) by Shamir et al. finds a solution with fewer communication rounds due to the similarities between computer problems. Konecn et al. developed FL, a centralized technique for training models. Instead of sending data samples to other nodes, it leverages data stored locally. Convergence and communication times were modified to strike a better balance between security and transparency of data. McMahan et al.'s research analyzed simple methods for training FL models. FedSGD averages local node gradients during the entire learning process, whereas FedAvg only does so once all clients have computed their local models. Zhu et al. claim that using non-IID training data from neighboring nodes significantly reduces accuracy. Users' privacy was compromised because many systems failed to relay information to all nodes. The FedAvg and FedSGD algorithms have been the subject of article suggestions for improvement. Connecting gadgets, seeing patterns in data, and learning to work together are all high on their list of priorities. While FedDACNE is theoretically superior, FedProx performs better in practice. Convergence to the average is enhanced by the proximal approximation in FedProx, as demonstrated by Li et al. Yang et al. investigate what happens to a wireless FL configuration when the network is linked and there is latency in an effort to determine the optimum strategy to spend energy efficiently while keeping latency restrictions. Uddin et al. demonstrate the convergence of mutual information-based FL solutions on clinical datasets. The authors refined their approach by considering the concept of a "information bottleneck" and a "Lagrangian loss function." FL is an expert in managing complex energy networks. Data privacy and security are addressed via FL solutions. Data about users is often stored in insecure centralized databases, which might compromise confidentiality. In order to employ FL approaches for smart grid activities like estimating energy use and discovering patterns among prosumers, it is necessary to install smart meters in areas where prosumers are situated. Li and his group have identified six different business sectors that could benefit from FL software. Su et al. investigate smart grids with decentralized deep learning, and Husnoo et al. employ FL to forecast prosumer energy consumption. The LSTM and FedAvg neural networks were used to complete the task. The FL smart grid demonstrated by Singh et al. is server-less and produces reliable results. As their FL technique of choice, Tak et al. opted upon LSTM. Even after several iterations of transmission, outcomes were often inaccurate without consumer clustering. In order to hasten convergence, Gholizadeh et al. (who studied FL) devised a novel method based on clustering processes. For smart grid prosumers to transmit data without incurring IID fines, Su et al. propose a secure floating-point (FL) approach. The best outcomes and most understandable dialogue emerge from training a two-layer deep reinforcement learning system. Wang et al. argue that FL should be integrated with smart meters and consumer-to-consumer social aspects. Saputra et al. made educated guesses about the energy consumption of EVs by employing FL design. By avoiding information exchange with the service provider, the charging station can cut down on operational expenses.

Improving the precision of predictions. A FL framework for trend-learning-based power data security in energy networks is described by Liu et al. Vertical and horizontal FL are utilized in this technique.

Because intrusions put the learning model and data transfer at risk, there needs to be a balance between security and decentralization. Usynin and his colleagues face a model inversion danger when an attacker dismantles the federated model and exchanges training data. Gradient-based techniques allow adversaries to view visual data, but they also provide safeguards. Data breaches can be avoided with Song et al.'s successful privacy-preserving data aggregation method since it connects weights without disclosing models. Even if many users abandon the search due to communication difficulties, a suitable model can be located. Data security was ensured by the use of poisoned FL designs by Ganjoo et al. and FL encryption methods by Liu et al. Ma et al. propose a method to counter FL-based Byzantine attacks. Gradient aggregation using a two-party calculation protocol is fast, secure, and private. Multi-party computation and Paillier's additive homomorphism are used by some to provide privacy and security. In the social Internet of Vehicles, Zhao et al. established key exchanges for the purposes of identity authentication and data encryption. Model extraction, poisoning assaults, and member incentives to comment on their own resources and local results are just some of the privacy and security challenges that Hou et al. discuss in their conclusion on FL. The current FL directions are listed in Table 1.

2.RELATEDWORK

Distributed ML techniques aim to improve learning and privacy. FL solutions only provide limited privacy and security. Centralised learning models can be attacked and become a single point of failure. Blockchain, FL, and distributed optimization address smart energy grid security, trust, and privacy leakage. The rest of this section discusses distributed optimization algorithms and smart energy grid security in FL solutions. Next, blockchain-based distributed FL methods and their advantages over classic FL models are examined.

Zinkevich et al.'s breakthrough FL model was their "one-shot parameter averaging" decentralized ML model based on parallel stochastic gradient descent (SGD). The suggested architecture used the SGD optimizer to train local models and a single final communication round to build the central model, which is not ideal. ML systems other than SVM ignored data distribution between nodes and one-shot parameter averaging. Distributed convex optimization is tested by Boyd et al. using alternating direction multipliers on the lasso, sparse logistic regression, basis pursuit, covariance selection, and SVM. The results demonstrated that communication rounds can hinder convergence. Distributed Approximate Newton (DANE) by Shamir et al. converges in less communication rounds by considering computer problem similarities. Konecn et al.'s centralized model training method FL uses local datasets at local nodes without sending data samples. To balance accuracy and data privacy, communication rounds and convergence time were optimized. McMahan et al. examined fundamental FL model training methods. Federated Stochastic Gradient Descent (FedSGD) averages local node gradients at each learning phase step, while FedAvg averages weights after all clients compute their local models. Zhu et al. say local node non-IID training data can significantly reduce accuracy. Several systems did not distribute data among nodes, compromising privacy. Articles suggest FedAvg and FedSGD algorithm improvements. Device communication, data heterogeneity correlation, and learning convergence are their concerns. FedDACNE is more exact theoretically, but FedProx outperforms it in practice. Li et al. showed that FedProx's proximal approximation component increases FedAvg convergence. Yang et al. examine how network connectivity and latency affect a wireless FL configuration in an energy efficiency optimization problem with latency limitations. Uddin et al. show mutual information-based FL solution convergence on clinical datasets. The authors improved their method by using the information bottleneck concept and a Lagrangian loss function. FL can control smart energy grids. FL solutions cover data security and privacy. Traditional systems store user data in centralized databases, compromising privacy and security. To apply FL methods to smart grid use cases like energy forecasting and prosumer pattern categorization, deploy distributed smart meters at prosumer sites. Six industrial domains are identified by Li and colleagues in FL applications. Su et al. study smart grids utilizing decentralized deep-learning, whereas Husnoo et al. predict prosumer energy demand using FL. Used FedAvg and LSTM neural network. Singh et al. propose a dataset-accurate serverless FL smart grid. Tak et al. chose LSTM for their FL method. Even after multiple transmission cycles, accuracy was poor without prosumer clustering. Gholizadeh et al. assessed FL and proposed a novel clustering method to accelerate convergence. Su et al. propose a secure FL mechanism for smart grid prosumers to send data without IID repercussions. Training a two-layer deep reinforcement learning system yields promising results and efficient communication. Wang et al. suggest connecting smart meters and prosumer social features with FL. Saputra et al. forecast EV energy needs using FL design. The solution lets the charging station avoid sharing data with the service provider, reducing communication expenses.

Making predictions more accurate. Liu et al. offer a FL framework for energy network power data security and use trend learning. The method uses horizontal and vertical FL.

Cyberattacks threaten the learning model and data dissemination, therefore security and decentralization must be balanced. Usynin et al. handle model inversion threats, when an attacker reverse-engineers the federated model and leaks training data. Gradient-based methods reveal picture data to attackers and show mitigation. The successful privacy-preserving data aggregation method by Song et al. connects weights without revealing models, reducing data leaks. The technique can calculate a suitable model even when many users disconnect due to communication issues. Ganjoo et al. poisoned FL designs and Liu et al. encrypted FL systems to protect data. Ma et al. present a FL-based Byzantine attack solution. A two-party calculation protocol implements safe, efficient, and privacy-preserving gradient aggregation. Others use multi-party computing and Paillier's additive homomorphism for privacy and security. For social Internet of Vehicles authentication and data encryption, Zhao et al. use key exchanges. Hou et al. conclude by reviewing FL security and privacy challenges such model extraction, poisoning assaults, and incentives for members to comment on their resources and local findings. FL's current directions are in Table 1.

Table1. The importance of Florida's smart grid..

Issues in Smart Grid Scenarios		FL Solutions
Data privacy preservation and security	Non-blockchain	Distributed perturbation [illegible], sequential learning [illegible], model inversion [illegible], data aggregation mechanism [illegible], poisoning attacks mitigation [illegible], Byzantine robust FL [illegible], homomorphic encryption [illegible], collaborative authentication protocol [illegible]
	Blockchain enabled	Incentivization and avoidance of model poisoning [illegible], blockchain for data sharing and serverless computing [illegible], swarm learning [illegible]
Optimization of communication costs, devices, and data heterogeneity		DANE [illegible], FedDANE [illegible], Structured and sketched updates [illegible], FedAvg [illegible], iterative algorithms [illegible], FedProx [illegible]
Analytics and energy efficiency		AI of things [illegible], load forecasting [illegible], energy data sharing [illegible], prosumer profiling [illegible], learning consumption patterns [illegible]

FL and blockchain will address issues of model centralization, trustworthiness, and data privacy. Very few methods were documented. Warnat-Herresthal et al.'s FL system architecture does not rely on a single point of coordination. A distributed smart contract handles some model activities, such as averaging and weight distribution. This method increases security because all clients may verify the validity of the master model and track its evolution in real time. We analyzed the IID's use of clinical data to categorize illnesses.

Few studies have considered the need of protecting user data, centralizing learning models, and preventing unauthorized changes in smart grid ML-based FL and energy demand prediction applications. This research proposes a blockchain-based distributed FL energy-demand forecasting approach to improve the field by protecting energy consumers' personal information. Only the weight of the local model that has been trained is transmitted to the blockchain; all other sensitive energy data remains at the edge prosumer nodes. Since the global model is replicated over the blockchain, it is impossible to alter it. As a result, understanding the impact of prosumer behavior is impossible.

3.SMARTCONTRACTSFORFEDERATEDLEARNING

Private information about energy consumption is stored on prosumer nodes. After users have trained a local machine learning (ML) model, smart contracts on a blockchain network will update the global model.

Consumer machine learning will be used to discover the vector-dependent function w: Rp R.

In RP, weights are simply weights. We will use n pairs of datapoints (xi, yi) to train the model w, where xiR are time-stamped feature vectors and yiR are meter-read energy values. As can be seen in Figure 2, the FL technique employs a network of k nodes to hone in on the global model function w. Separate sets of k energy datapoints (ni n, i 1..k, and i ni = n) are generated. FL estimates the amount of energy consumed by prosumers.

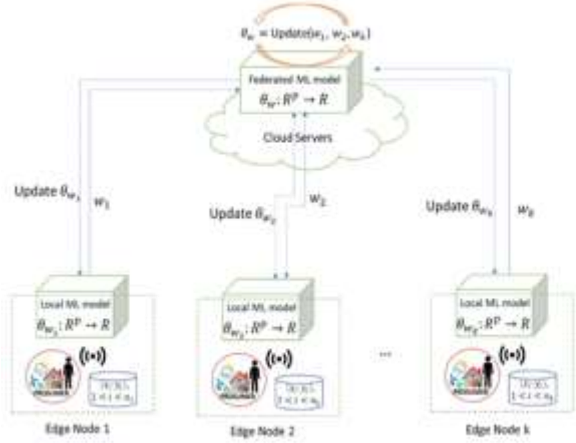

Figure2.FL estimates prosumers' energy needs.

Each of k energy consumer edge nodes trains the models locally, yielding parameter vectors wi that minimize prediction error:

$$f_w : R \rightarrow R, \quad f_w(x_i, y_i) = Error(y_i, \theta_w(x_i))$$

Finding the optimal weight vector is a goal for each edge node i.

To reduce local prediction error, a centralized node uses a federated model function and local model parameters to generate a weight vector wF.

The weight vector wF of the federated model is typically calculated by a function by adding the weights of the edge models. The federated weight vector can be modified in two distinct ways. Using SGD to obtain the mean of the edge node weight vectors is the first approach, which modifies the global model weights. Next, DANE modifies the federated weight by averaging the gradients of the local weights [40,41]. The average gradient of the edge model is used to make adjustments to the weight vector at various phases.

In federated machine learning optimization, the k-edge energy data plays a crucial role.

Prosumers are shown by dots in Figure 3. A weight vector is delivered to the edge consumer

JNAO Vol. 14, Issue. 01: 2023

nodes to kick off the global model. The improvement in the federated weight vector wF after many runs of s is displayed in lines 4-9. Local gradients are gathered, a global gradient is computed, federated weights are updated, and a new round of local model training is initiated by sending the weight vector ws to the edge nodes. The weight vector from the federated model keeps being returned by the program.

Input: k edge nodes associated with prosumers, central node for storing the global ML model for energy prediction, number of iterations
Output: weight vector w_g of the global ML prediction model
Begin
1. Collect energy data points (x_i, y_i) in k edge nodes
2. Compute an initial weight vector for the global prediction model w_g^0
3. Distribute an initial weight vector w_g^0 to k edge nodes
4. **For each** s **in** 1 to $no_{iterations}$
5. Train the local ML prediction models
6. Collect edge training results as local weights $w_i^s, i \in \{1..k\}$
7. Compute global prediction model weights $w_g^s = \sum_{i=1}^{k} \frac{n_i}{n} * w_i$
8. Distribute the global model weights vector w_g^s to k edge nodes
9. **End For**
10. **Return** w_g^s

End

Figure3. Energy forecast for prosumers.

Our federated prosumer energy prediction technique utilizes blockchain and smart contracts to ensure that the global ML model is always accurate (Figure 4). We employ smart contracts to share on the blockchain the weight vectors used in local energy forecast models. The following step is to encrypt the model and then store it on the blockchain. Simultaneously, all of the network overlay nodes receive a copy of the local model's weight.

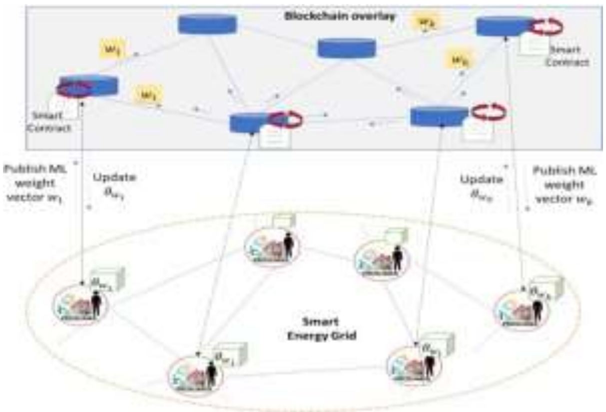

Figure4. Energy forecasting for prosumers utilizing distributed blockchain FL.

Smart contracts on the blockchain network overlay control the energy demand prediction weights vector for the global shared ML model. Each learning node is equipped with the addresses of the contracts that modify the global model's weights. The second table illustrates the relationship between the key functions of smart contracts and FL levels. Since acquire functions do not alter the current state of the contract, participants will not be compensated for asking about the central weight.

Decentralized Learning Steps	Smart Contract
Weight vector of global model	Int256 [] globalModelWeights
Initialize the global model	function set Initial Weights (int256[] memory weights)
Retrieve the initial weights of the global model	function get Initial Weights () public view returns (int256[] memory)
Push the edge nodes training results	function post Local Weights (int256 [] memory weights)
Update the global model weights using the local models	function update Global Model () public view returns (int256[] memory)

Figure 5 depicts the impact of smart contracts on the relative importance of different global modes. Due to mapping restrictions, we were only able to log the addresses of ultra-peripheral prosumer accounts and Present accounts. To make a modification to globalModelWeights, it is essential to take an average of localModelWeights, which stores the weights for edge accounts. After calling "update Global Model," the globalModelWeights are reset, and a loop iteratively processes all of the prosumer accounts, appending each local weight vector to its proper place and averaging the global weights by the number of edge node participants.

```
mapping (address => bool) accountPresent;
address [] public accounts;
int256[] globalModelWeights;
mapping (address => int256[]) localModelWeights;

Function updateGlobalModel
    Input: msg.sender, initWeights, localModelWeights
    Output: averageweights
    Modifiers: public override
    Begin:
        globalModelWeights = new int256[](initWeights.length);
        for (uint256 i = 0; i < accounts.length; i++) {
            for (uint256 j = 0; j < initWeights.length; j++) {
                globalModelWeights [j] += localModelWeights[accounts[i]][j];
            }
        }
        for (uint256 i = 0; i < initWeights.length; i++){
            globalModelWeights [i]/= int256(accounts.length);
        }
End
```

Figure5 Smart contracts update global energy-demand prediction model weights.

Smart contracts are activated by nodes at the smart grid's consumer edge (Table 2). Figure 6 depicts the pseudocode for the function used to maintain local ML models. Edge nodes must be in sync with each other to prevent model update mistakes during interaction cycles (local or global model update), such as when local model weights are adjusted and global model updates are broadcast to the blockchain. We synchronize the timestamps on edge nodes to provide benchmarks for the training iteration. Each training cycle will have a time constraint for retrieving global model weights from the blockchain and transmitting local model weights. Edge node local weights are considered here; in other contexts, they are disregarded.

First, make sure the start and end dates are in sync with other training edge prosumer nodes by setting the time zone (lines 5-6). Node (lines 8-12) iteratively trains local model using blockchain weights using local data. When the post milestone is reached in the local timezone, the smart contracts will upload the local weights vector ws to the blockchain. When the current local time reaches the get milestone (lines 10 and 11), a request is sent to the blockchain for a new weight vector to be used in the construction of the next local model.

1: Function updateLocalModel
2: Input: w_g^s global model weight vector corresponding to training iteration s
3: Output: Local weight vector w_l^s
4: Begin
5: $local_{time}$ local clock synchronization with UTC time
6: $local_{milestones}$ model updating milestones synchronization
7: w_L^0 = getInitialWeights ()
8: for s in 1 to $no_{iterations}$
9: $w_L^s = train(\theta_{w_L}, \{(x_i, y_i) | i \in \{1,, n_k\})$
10: if $(local_{time} = local^{post}_{milestones})$ then postLocalWeights(w_L^s)
11: if $(local_{time} = local^{get}_{milestones})$ then w_L^{s+1}= retrieve updateGlobalModel()
12: end for
13: return w_L^s
14: End

Figure6. Updated ML models reach edge prosumer nodes.

4.EVALUATIONRESULTS

Prosumers' energy consumption data was collected through our platform for use in validating the blockchain-based energy-demand forecast. Each prosumer is provided with a power meter that communicates via the International Electrotechnical Commission's (IEC) 62056 protocol, as well as an HTTP-based power quality analyzer. The meters update their local data model with the information they communicate over MQTT every five seconds.

Data from 15-minute energy readings were compiled for five months. Twenty-four energy values, or one value for each hour, are utilized to predict what individuals will wish to buy the next day. Because prosumers' energy consumption might vary greatly, a clustering technique is utilized to identify individuals with comparable peak energy demands. Prosumer monitoring edge devices do not collect energy data that complies with the IID standard. Furthermore, their consumption habits may vary. Therefore, in the first four months, each local model was trained on random samples of data from energy meters, validated using 10% of data, and tested with data from the next month.

Local prediction at each prosumer edge node is handled by a fully-connected MLP and feedforward neural network. Many MLP configurations were trained and examined to determine the learning meta-parameters (weight vector). Each iteration consisted of one epoch since FedAvg averages the local model weights at regular intervals. Through testing, we were able to determine how often an optimal average should be calculated. Meta-parameters such as the number of hidden layers, neurons, and learning rate were also altered. Our strategy for selecting features relied heavily on the current date and time in addition to the total quantity of energy consumed.

The MLP model for predicting energy has a single hidden layer of 30 neurons. In addition, the output layer and rectified linear units (ReLu) are both active (Table 3). We employed a 32-stage He uniform-variance scaling initializer in conjunction with a minimum squared error (MSE) loss function and a random gradient descent (RGD) approach. Our optimal model required 26 inputs: 24 hours of energy data from the previous day, one weekday, and a Boolean value indicating whether or not the expected day was a weekend. Each time before a round,

The input data was normalized in the range [1, 1] using a min-max scaler before being fed into the network. Applying the inverse scaling function after each prediction will produce fewer typical outcomes.

Table3. Local energy prediction model MLP configuration.

MLP Configuration	
Number of input neurons	26
Number of output neurons	24
Number of hidden layers	1
Number of neurons in hidden layer	30
Activation function at hidden layer	ReLu
Activation function at output layer	Linear
Optimizer	SGD
Loss function	MSE
Kernel initializer	He uniform
Batch size	32

To keep the global machine learning model up-to-date, smart contracts were developed on a private Ethereum blockchain using Solidity to communicate with the blockchain. Because of its compatibility with Turing-complete programming languages like Solidity, Ethereum was selected as the platform for smart contracts. The user can also modify other aspects of the chain, including as the consensus algorithm, the number of blocks per minute, the size of each block, the cost of gas, and more.

Models of regional nodes in the prosumer value chain were built with Keras. Through a Node.js-and web3-based blockchain API, the smart contract and edge prosumer nodes were able to communicate with one another. Each edge prosumer node has its own encrypted blockchain account and can process HTTP GET and POST requests. Edge prosumer nodes retrieve the central model weights from the blockchain at the beginning of each iteration, train the model, and then release the weights for their local area.

The starting weights of the global model and the weights of the edge prosumer nodes are stored in an array and a map, respectively. To reduce unnecessary blockchain overhead, we stored the weights of local prediction models in one-dimensional arrays and made updates available at the edge prosumer node. The edge prosumer node must compress the local model weights array before adding it to the smart contract. The weights array from the smart contract must be converted from a 1D array to a Keras model before the edge prosumer device may utilize it to re-create a model (Figure 7).

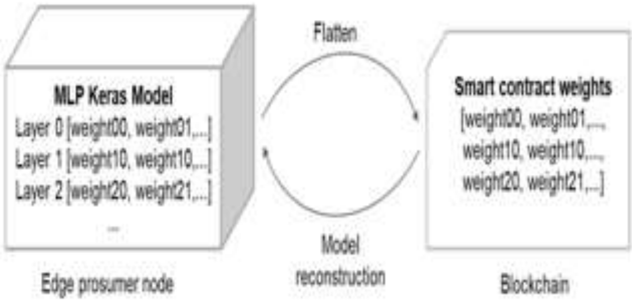

Figure7. Keras model integration simplifies smart contracts.

We compared the accuracy of predictions made by centralized learning, edge learning, and distributed FL using IID data for energy use by prosumers. Centralized learning incorporated all the data from the edge prosumer nodes into a single global model. Figure 8 depicts the optimal results achieved by a model with a single hidden layer of 35 neurons, an SGD optimizer with a learning rate of 0.90, and 128 batches trained for 1100 epochs. It turns out that this method of learning provides the highest accuracy (smallest MAD).

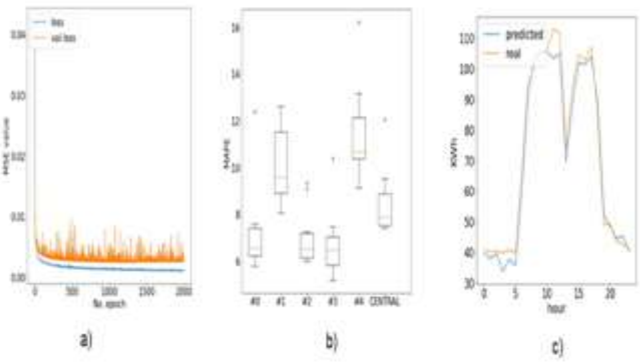

Figure8.(**a**)Trainingcentralmodelon2000epochs(after 1100epochstheimprovementsareminimal); (**b**)CentralizedlearningMAPE;(**c**)Energyprediction resultsforaprosumer.

Each consumer node at the edge trains its model using only local energy data in the local edge approach. Nodes never share their model parameters or energy data with one another. Each node is responsible for keeping its own data and optimizing its own local model. Finally, we found the average MAPE to display the overall accuracy results and generated a graph of the accuracy for each prosumer node. Figure 9 demonstrates that some prosumers, notably Prosumer 4, generate many errors since local datasets aren't significantly diverse from one another, despite the fact that the mean MAPE is 10.82.

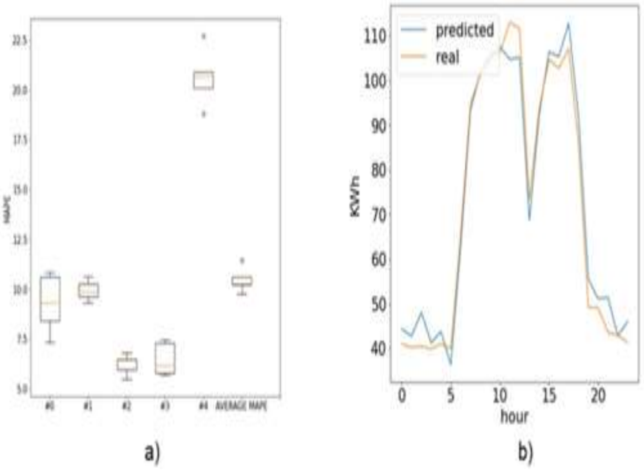

Figure9.(**a**)EdgelearningMAPE(**b**)Energypredictionfora Prosumer#3.

In the local edge technique, prosumer nodes at the periphery only use data from their immediate vicinity to train their models. Model parameters and energy data are not communicated between nodes. Local nodes are responsible for data collection and model tuning. Finally, we found the average MAPE to display the accuracy across all prosumer nodes in a graph. As can be seen in Figure 9, Prosumer #4 causes significant errors even if the average MAPE is 10.82. This is because there is little variation in local datasets.

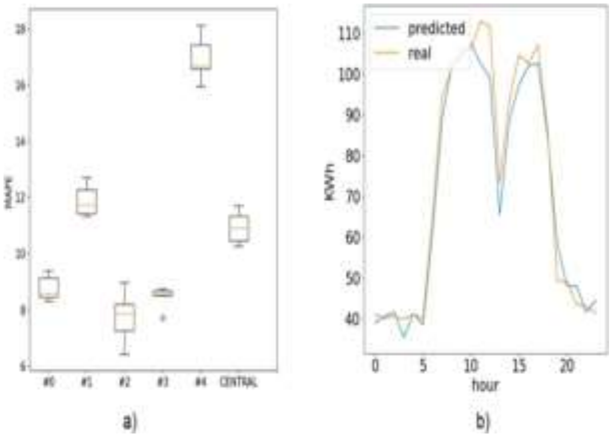

Figure 10.(Edge learning MAPE and Prosumer #3 energy prediction.

Both a blockchain-based distributed FL configuration with IID data and one without were analyzed. To facilitate joint learning, the IID architecture dispersed energy data randomly amongst nodes at the periphery of the prosumer network (Figure 10). Long training sessions were used to determine how many times the learning model should be performed by comparing validation and train loss. The local energy data at the periphery is representative of the entire case population of prosumers. While it improves forecast accuracy, it cannot compete with centralized models. This dispersion of information, however, aids the distributed FL prediction up to a certain point.

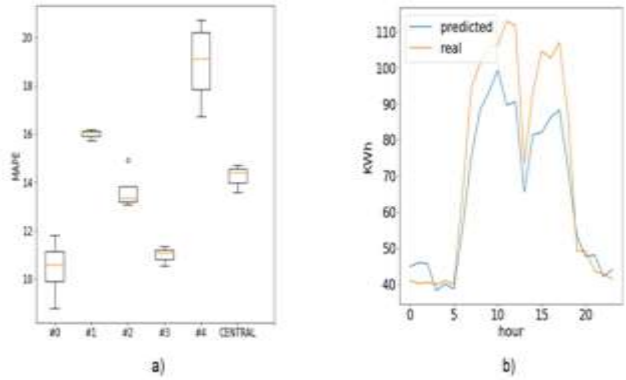

Figure11.(IID data-trained blockchain-based distributed FL model accuracy.Prosumer #3 energy forecast.

The distribution of data is the only way in which our model diverges from the distributed FL setup with non-IID data. Figure 11 shows that the accuracy of a distributed FL model built on blockchain technology without IID suffers. The average MAPE, on the other hand, is 14.35, which is excellent for DR programs and ensures the security of prosumers' energy data. This strategy may be more effective for prosumers with a higher MAPE score. In the local edge test, Prosumer No. 4 had the lowest MAPE value, but the learning solution improved its accuracy. This was due to the fact that the distributed learning blockchain model had access to a much larger pool of information than the local test case data.

Table4. The accuracy of blockchain-based distributed FL models trained with non-IID data; (b) Prosumer #3 energy forecast..

JNAO Vol. 14, Issue. 01: 2023

Prosumer	Centralized Learning	Local Learning	Blockchain-Based Distributed FL Non-IID	IID
0	7.58	7.96	10.80	8.83
1	12.03	9.95	16.22	11.74
2	7.15	7.63	13.97	7.66
3	7.43	6.70	11.08	8.27
4	13.37	21.85	19.69	17.02

Table 4 displays the results of a comparison between the test energy data set and the mean absolute percentage inaccuracy for each scenario. Our learning approach achieves the same outcomes as state-of-the-art centralized and local trained models without violating privacy regulations, even though we adjusted each scenario manually.

Table5. Average prosumer test case MAPE findings.

Test Case		MAPE Min	Average	Max
Blockchain-based federated learning	Non-IID	10.80	14.35	19.69
	IID	7.66	10.62	17.02
Centralized		7.15	9.51	13.37
Local		6.70	10.82	21.85

Table 5 shows the average minimum and maximum MAPE values for the centralized model, which outperformed locally trained models due to data variability.

5.DISCUSSION

Social and human issues such as privacy, households, and community sustainability goals are not yet considered in local energy management practices. The energy system is transitioning toward distributed generation as a result of technological and economic advancements and ambitious goals set by energy regulators. Customers and residents don't become involved enough. Due to the increasing digitalization of the energy infrastructure and the proliferation of new energy services, prosumers have less and less ability to regulate or monitor the dissemination of private information to those involved in disaster relief efforts. It's likely that utility providers aren't doing a good job of handling energy data, leading to a loss of agency for end users. This prevents them from engaging in energy-related pursuits. Since energy data is maintained on prosumer edge nodes and only model parameters are communicated, these issues may be mitigated by the blockchain-based distributed FL solution. Safekeeping data that complies with GDPR and tracking the origin of local machine learning parameters are both tasks that can be done with the use of blockchain technology. Support for FL frameworks is included.

Concerns and questions remain about how to best combine FL with blockchain technology. One limitation is that the platform and configuration choices affect the processing cost of integrating blockchain technology. The degree of complexity of the global model is influenced by factors such as the number and dimensions of edge prosumer node weight vectors. Consequently, deploying complicated machine learning models on a public blockchain is impossible unless at least certain parameters in the global model are considered, either in a timely fashion or at all. Each prosumer can have their own private model parameters, eliminating the need for the blockchain.

The cost of the gas required to store the global model and power smart contracts might add up quickly. Public infrastructure blockchain deployment costs could be significantly inflated due to the learning convergence time, which regulates the frequency of communication between edge prosumer devices and smart contracts. Therefore, the blockchain-based distributed FL design is well-suited for private blockchains or low-computing-power networks, such as Proof of Stake systems. Inconsistencies in the training data could hinder the ability of a distributed FL built on the blockchain to reliably forecast future energy use. Energy consumption and availability among prosumers varies widely. It's possible that not all predictors are equally well trained and matched when they all use the same model. This is common in non-IID FL models, thus it's important to identify them and eliminate them as early as possible in the training process. FedProx can help with the statistical inconsistency in FL. With so much data and processing capacity at their disposal, prosumer nodes may perform a wide variety of computations locally.

In order for our blockchain to function with the first wave of prosumers, we propose dividing them up into smaller groups (called "clusters") and giving each cluster its own set of FL models. The scaler needs to function for all participants regardless of the classification technique employed and whether or not it is aware of the energy data samples. Prosumers' energy amplitudes were normalized to the range from zero to maximum demand because they were found to be comparable. Normalized values aid model convergence when scaling rates for prosumers are not constant. At the end of the day, a zero-knowledge proving technique can be used to prove that a user is a part of a cluster without disclosing any of the cluster's information.

From what we were able to determine, the blockchain's local prediction models were restricted to customers who had their own residences. The guidelines specified that only information from approved energy meters be used. False weights published by malicious actors might reduce confidence in the global model and cause havoc for the distributed FL process that relies on blockchain technology. The issue can be solved by conducting trials with prospective edge consumer nodes. If new blockchain-based smart contract functionalities were developed, validation might be simpler to grasp. This could also be done by someone with a vested interest in the system, like as the Distribution System Operator, who is concerned with safety and reliability. The blockchain makes it easy to keep track of transactions and can identify peers who feed false data into learning models. This method can be used in conjunction with incentives to motivate prosumers to meet consumer demand. Incentives can be tied to a person's ability to learn and forecast, in addition to their flexibility.

Encourage anybody interested in participating in the decentralized learning system for energy consumption forecasting to do so. They can develop by participating on the blockchain. This could speed up the process by which a new member adds local energy samples without compromising privacy. It will also help new participants who don't have trained machine learning models anticipate energy more accurately. The smart grid scenario could do pre-verification on potential users to ensure that only authorized individuals gain access.

Finally, we may try to predict future energy demands using local machine learning models and privacy economics. Blockchain-based model-sharing initiatives may benefit businesses and governments alike. The blockchain can monitor parameter changes and penalize illegal conduct, which is beneficial for prosumers who benefit from sharing their machine learning models. In a blockchain overlay market, edge prosumer nodes might earn income by imparting and receiving models. The forecast is improved when the model's edge consumer nodes are modified. Some nodes in the network's periphery receive the model and apply it only locally.

There will be an additional fee for expert predictions that do not factor in training data. Users of nodes in the periphery can eliminate bad nodes by verifying trained models.

6.CONCLUSIONS AND FUTURE WORK

This study discusses a bl+ockchain-based distributed FL solution that enables prosumers to forecast their energy consumption and take part in grid management initiatives. We combine blockchain and the FL model to ensure the security of the data used to make estimates about future energy demand. Only the model parameters are communicated via blockchain to the periphery prosumer nodes; the energy data is maintained locally. Due to the distributed nature of blockchain transactions, the parameters of the global federated model are kept in a way that is immutable. Local machine learning models can be integrated with blockchain-specific capabilities through smart contracts. This equalizes the data, allows the model parameters to expand, and reduces the load on the blockchain. The distributed nature of the blockchain network makes it impossible to alter the global prediction model once it has been replicated and distributed. Because of this, inversion assaults targeting prosumer behavior are difficult to detect.

We analyzed the prevalence of centralized, decentralized, and regional edge FL ML models. The centralized system accurately predicts future energy consumption, but at the expense of individual privacy. The replacement we propose makes no difference in precision. Energy providers may put prosumers' minds at ease about their data being kept private while still providing accurate predictions for using DR to its maximum potential.

To improve the precision of energy forecasts for prosumers, we want to implement sophisticated deep-learning models like CNN and LSTM. We propose to employ novel approaches to partially integrate learning parameters or model compression to address the known issues of modern blockchains, such as block size, transaction dimensions, and gas usage. We will investigate alternative blockchain platforms to the ones now used in the energy domain, which only reward energy flexibility, in order to get over overhead constraints and encourage prosumer learning and prediction processes.

y.

REFERENCES

1. Javid,I.;Chauhan,A.;Thappa,S.;Verma,S.K. ;Anand,Y.;Sawhney,A.;Tyagi,V.V.;Anand, S.Futuristicdecentralizedcleanenergynetworksinviewofinclusive-economicgrowthandsustainablesociety.J.Clean.Prod.**2021**,309,127304.[CrossRef]
2. Kumar,R.S.;Raghav,L.P.;Raju,D.K.;Singh, A.R.Intelligentdemandsidemanagementfor optimalenergyschedulingofgridconnected microgrids.Appl.Energy**2021**,285,116435.[CrossRef]
3. Ibrahim, B.; Rabelo, L.; Gutierrez-Franco, E.; Clavijo-Buritica, N. Machine Learning for Short-Term Load Forecasting in SmartGrids.Energies**2022**,15,8079.[CrossRef]
4. Petrican, T.; Vesa, A.V.; Antal, M.; Pop, C.; Cioara, T.; Anghel, I.; Salomie, I. Evaluating Forecasting Techniques for IntegratingHousehold Energy Prosumers into Smart Grids. In Proceedings of the 2018 IEEE 14th International Conference on IntelligentComputer Communication and Processing (ICCP), Cluj-Napoca, Romania,6–8 September 2018; pp.79–85.
5. Amasyali,K.;El-Gohary,N.M.Areviewofdata-drivenbuildingenergyconsumptionpredictionstudies.Renew.Sustain.EnergyRev.**2018**,81Pt1,1192–1205.[CrossRef]
6. Vesa,A.V.;Cioara,T.;Anghel,I.;Antal,M.;Pop,C.;Iancu,B.;Salomie,I.;Dadarlat,V.T.EnergyFlexibilityPredictionforDataCenter Engagement inDemand ResponsePrograms.Sustainability**2020**, 12,1417.[CrossRef]
7. Sha,H.;Xu,P.;Lin,M.;Peng,C.;Dou,Q.Developmentofamulti-granularityenergyforecastingtoolkitfordemandresponsebaselinecalculation.Appl.Energy**2021**,289,116652.[CrossRef]
8. Shen,M.;Lu,Y.;Wei,K.H.;Cui,Q. Predictionofhouseholdelectricityconsumptionandeffectiveness ofconcertedinterventionstrategiesbasedonoccupantbehaviourandpersonalitytraits.Renew.Sustain.EnergyRev. **2020**,127,109839.
9. Vigurs,C.;Maidment,C.;Fell,M.;Shipworth, D.CustomerPrivacyConcernsasaBarriertoSharingDataaboutEnergyUseinSmart LocalEnergySystems:ARapidRealistReview.Energies**2021**,14,1285.[CrossRef]
10. Safdarian, A.; Fotuhi-Firuzabad, M.; Lehtonen, M. Demand Response from Residential Consumers: Potentials, Barriers, andSolutions. In Smart Grids and Their Communication Systems. Energy Systems in Electrical Engineering; Kabalci, E., Kabalci, Y., Eds.;Springer:Singapore,2019.