BFRT: **B**lockchained **F**ederated Learning for **R**eal-time **T**raffic Flow Prediction

Collin Meese*†   Hang Chen*‡   Syed Ali Asif‡   Wanxin Li†   Chien-Chung Shen‡   Mark Nejad†

†Department of Civil and Environmental Engineering, University of Delaware, U.S.A.

‡Department of Computer and Information Sciences, University of Delaware, U.S.A.

{cmeese,chenhang,asifrabi,wanxinli,cshen,nejad}@udel.edu   *These authors contributed equally to this work.

Abstract—**Accurate real-time traffic flow prediction can be leveraged to relieve traffic congestion and associated negative impacts. The existing centralized deep learning methodologies have demonstrated high prediction accuracy, but suffer from privacy concerns due to the sensitive nature of transportation data. Moreover, the emerging literature on traffic prediction by distributed learning approaches, including federated learning, primarily focuses on offline learning. This paper proposes BFRT, a blockchained federated learning architecture for online traffic flow prediction using real-time data and edge computing. The proposed approach provides privacy for the underlying data, while enabling decentralized model training in real-time at the Internet of Vehicles edge. We federate GRU and LSTM models and conduct extensive experiments with dynamically collected arterial traffic data shards. We prototype the proposed permissioned blockchain network on Hyperledger Fabric and perform extensive tests using virtual machines to simulate the edge nodes. Experimental results outperform the centralized models, highlighting the feasibility of our approach for facilitating privacy-preserving and decentralized real-time traffic flow prediction.**

Index Terms—**Blockchain, Federated Learning, Traffic Flow Prediction.**

I. INTRODUCTION

Traffic prediction plays a critical role in alleviating traffic congestion and associated negative impacts (e.g., unreliable travel time estimates, increased fuel consumption, adverse environmental effects) [1]. Recently, as vehicle miles traveled (VMT) continues to grow annually, traffic congestion has become a pervasive societal problem. For example, the 2019 Urban Mobility Report estimated congestion in the United States resulted in an additional 8.8 billion hours of travel time and 3.3 billion gallons of extra fuel consumption in 2019 alone [2]. Consequently, real-time traffic prediction can help alleviate congestion by providing roadway users and businesses accurate real-time traffic conditions for route planning and offering rerouting options once the vehicles are en route.

Recently, deep learning (DL) has become a promising method for traffic flow prediction (TFP), having demonstrated much success in the literature [3] [4], with prediction accuracy as high as 93%. However, in the context of the TFP problem, existing DL models are centrally trained, requiring vast amounts of data to be collected and aggregated by a data center or the cloud for processing. This process makes it difficult to comply with the data privacy regulations [5] if all of the collected data must be directly shared to facilitate model creation, or when the crowd-sourcing techniques are used [6].

In addition, the emergence of connected and autonomous vehicles (CAVs) into the internet of vehicles (IoV) network will necessitate new methodologies, including real-time learning and prediction, for utilizing the wealth of data dynamically generated by sensors on CAVs [7]. Therefore, a paradigm shift in DL methodologies becomes necessary to enable efficient and distributed online model training leveraging real-time data while protecting data privacy.

Federated learning (FL), where multiple participants collaboratively train a learning model without exposing the underlying data [8], has been investigated as a way to enable efficient and distributed knowledge sharing for various applications [9]. For TFP, FL can train the model in real-time and update it dynamically as traffic sensors and edge devices continuously collect incoming traffic data. Moreover, FL removes the need to share or aggregate locally-collected traffic data, providing improved communication efficiency, privacy, and security for stakeholders (e.g., data collectors).

This paper proposes a novel framework for real-time traffic prediction by integrating FL with a permissioned blockchain network of edge devices. Our FL approach to real-time TFP utilizes the *Federated Averaging* (*FedAvg*) algorithm [8] with deep learning models. Specifically, the roadside units (RSUs) collect local traffic data within their respective observance areas and then leverage it to train a localized traffic flow prediction model. After training, the RSUs share their knowledge with other participants by sending only the up-to-date model parameters, in contrast to sharing the local traffic data. In addition to preserving privacy, our approach has the added benefit of distributing training workload to the RSUs at the network edge.

We propose to use a permissioned blockchain network as the framework for FL. Blockchain is a decentralized network technology that can provide the benefits of reliability, security, and integrity for all stored local models in comparison to centralized storage alternatives (e.g., cloud service providers) [10] [11]. Regarding the IoV, blockchain has been recently demonstrated as a strong candidate for improving security of the networking layer [12]. In the case of permissioned blockchains, the consortium of participating nodes control both system usage (writes) and data access (reads) [13] [14]. This is in stark contrast with public permissionless blockchains (e.g., Bitcoin), where anyone can join the network at will, access all of the data, and participate in consensus processing [15].

These properties of permissioned blockchain make it an ideal candidate for the FL framework.

Together, permissioned blockchain and FL provides a novel way to train TFP models in real-time using locally collected live data, with additional benefits of being dynamic, privacy-preserving, decentralized and low-latency in comparison with the existing, centralized model training approaches. Moreover, the continued integration of newly learned model parameters from RSUs enables the global model to better react to dynamic network conditions. In contrast, existing static and centralized models suffer from a significant delay to learn from newly collected data [16]. Notably, these benefits provide strong justification for further research into integrated blockchain and federated learning-based approaches for real-time TFP.

Contributions. This paper presents *BFRT*, a blockchain-enabled Federated Learning approach to real-time traffic flow prediction. In this work, we make the following contributions:

- We propose a privacy-preserving and secure-by-design FL framework for collaborative and real-time traffic prediction leveraging RSUs, edge devices, and permissioned blockchain. Specifically, we design federated versions of both the LSTM and GRU models and use traffic flow prediction as the example case study.
- Unlike existing works [5] [17] [18], we design a real-time FL process where the data shards for each participant (e.g., RSU) are distinct and private. Online learning simulations are conducted to replicate real-time data collection and training of the federated model. We evaluate the learning performance of our FL models for predicting traffic in a real-world scenario using dynamically collected incoming arterial data. Experimental results show that our FL approach can generally outperform the centralized baseline models.
- We prototype a permissioned blockchain network using Hyperledger Fabric [19] and simulate the edge nodes as resource-constrained virtual machines. We evaluate the performance of the permissioned blockchain network using the Hyperledger Caliper [20] benchmarking tool. Experimental results demonstrate the blockchain network can provide suitable throughput and latency for our proposed FL architecture.

II. Related Work

In recent years, FL has emerged as a novel approach for ensuring the privacy of user data in a collaborative learning setting [21]–[24]. Specifically, FL is a distributed machine learning approach that enables a model to be trained locally by a decentralized dataset hosted across various devices with different locations. Blockchain technology has been applied in some FL studies to achieve true decentralization during the FL model aggregation [25]–[27]. Additionally, blockchain-based FL implementations leverage the blockchain features such as auditability, traceability, and immutability to make the FL approach more robust, reliable, and trustworthy [8] [28].

Presently, FL has been at the forefront of various collaborative learning research areas where a large amount of data is required to train the model and achieve desirable performance. The authors of [29] discussed how FL could be used instead of centralized machine learning (ML) for building vehicular network applications in intelligent transportation systems. FL has also been implemented in several collaborative data-preserving schemes in smart AI-based IoV systems. Specifically, in the study of [30] the authors proposed an FL approach to enable privacy-preserving collaborative ML across a federation of independent Drones-as-a-Service (DaaS) providers for traffic prediction and parking occupancy management. In [17] the authors simulated a hierarchical blockchain-enabled FL algorithm for knowledge sharing within the IoV to improve reliability and security.

With the ever-growing amount of traffic data and the introduction of DL, traffic flow prediction has achieved tremendous success. However, the current prediction approaches are also concerned with the challenges regarding user privacy. Considering the performance trade-off, the authors of [5] proposed an FL-based gated recurrent unit neural network algorithm (Fed-GRU) and a clustering-based ensemble scheme to organize the data-generating entities for TFP before applying FedGRU. Additionally, in [31], a multi-task FL framework implementing hierarchical clustering to partition collected traffic data for traffic flow prediction and route planning was proposed to cover the diverse traffic situations. Although these works have used FL and DL approaches for TFP, they have primarily evaluated their design with identical data shards using offline FL training and inference. To the best of our knowledge, no other FL studies for TFP have focused on the real-time setting for online FL learning and inference with distinct data shards.

III. BFRT System Design

A. Problem Definition

In the context of our work, we define real-time FL-based TFP as a traffic prediction problem along an arterial corridor. Overseeing the corridor, there is an administration (e.g., the State's Department of Transportation) that manages and operates a set of deployed RSUs acting as traffic collection devices along the corridor. Alongside the RSUs are the edge devices that cooperatively train an online traffic prediction model leveraging the continuously collected traffic data from the RSUs in real-time, without sharing their underlying data. The edge devices also interact with the backend edge computing servers to execute the *FedAvg* algorithm and participate in blockchain-related operations.

B. System Architecture of BFRT

In the framework of Hyperledger Fabric, the entities of *clients*, *peers*, and *orderers* are defined. Clients $\mathcal{C} = \{c_1, c_2, ..., c_{|\mathcal{C}|}\}$ are the participants who interact with the Hyperledger Fabric blockchain network but do not expend additional resources on blockchain operations (such as transaction evaluation, ordering, validation, and block creation and storage). Peers $\mathcal{P} = \{p_1, p_2, ..., p_{|\mathcal{P}|}\}$ represent the nodes who perform transaction execution, endorsement, and validation, as well as maintain the blockchain ledger. Lastly, orderers
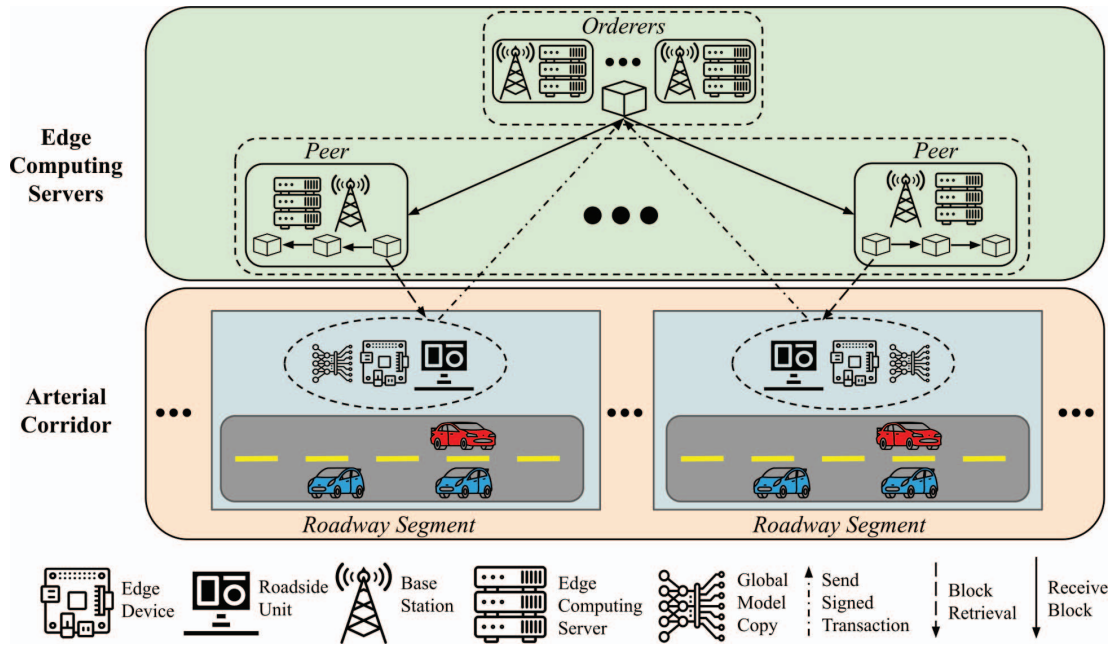
318

Figure 1: System architecture of *BFRT*.

$\mathcal{O} = \{o_1, o_2, ..., o_{|\mathcal{O}|}\}$ are the nodes tasked with consensus processing that order the transactions and batch them into blocks, ensuring consistency and fault tolerance for the blockchain network.

Fig. 1 depicts the architectural components of *BFRT*, the mapping of these components to the entities of Hyperledger Fabric, and their operations and interactions. All the devices (RSUs, edge devices, and edge computing servers) are owned and operated by the administration to have the most deployment flexibility as the deployment strategy for peers and orderers can significantly impact the blockchain's performance (throughput and latency) for traffic prediction.

RSUs deployed along the roadway collects local traffic data in real-time. The **edge device** co-located with each RSU possesses a current copy of the **global traffic prediction model** and acts as a client to participate in the FL process by training the global model with its locally collected traffic data by the RSU. In *BFRT*, the global model can be either the Federated GRU model or the Federated LSTM model. After one round of training, each edge device submits a *transaction proposal* containing the updated parameters of its locally trained model, encapsulated in the HDF5 file format, to all the **edge computing servers** acting as peers for endorsements.

In *BFRT*, the backend edge computing servers are deployed alongside base stations. Each peer edge computing server independently evaluates the transaction proposal and sends back an endorsement to the client. The client edge device packages all the received endorsements into a transaction, then digitally signs it, and submits it to the edge computing servers that act as orderers. The orderers execute the *RAFT* consensus algorithm [32] to establish an unambiguous order on transactions and batch them into a new block. Note that the

edge computing servers can be configured to run as either peer, orderer, or both. Each peer edge computing server receives the new block, validates the transactions, and commits it to its blockchain ledger copy. Each client edge device downloads the block, then retrieves the encapsulated transactions, and lastly performs *FedAvg* on the sets of most recent model parameters to generate a new version of the global model.

C. Real-time Federated Learning of Traffic Flow

BFRT trains a model in a real-time manner without requiring exchange or aggregation of any collected data, resulting in dynamic and efficient-to-update traffic prediction models. *BFRT* accomplishes this by letting the clients $\mathcal{C}$ collaboratively train a single, continuously updated global prediction model G using *FedAvg*, leveraging the new incoming traffic data, in a series of communication rounds $R = \langle r_1, r_2, \cdots, r_i, \cdots \rangle$. During each round r, every client $c \in \mathcal{C}$ performs the following sequence of operations: (1) collect local data; (2) train the global model with local data; (3) generate a transaction proposal with the updated model parameters; (4) send the transaction proposal to all the peers; (5) upon receiving endorsements, package the received endorsements into a transaction, digitally sign it, and submit the transaction to the orderers; (6) retrieve the newly created block; (7) extract parameters from the transactions in the block; and (8) update the global model. These operations are presented in Algo. 1 and Fig. 2.

Initially, each client c is provided an identical model G^0 before the first round r_1. G^0 could optionally be a pretrained model to jump-start the learning process. At the start of round r_i, each client c collects the incoming local traffic data $d_c^{in,i}$ (Algo. 1: line 2) for a predefined period p to be combined with its local historical traffic data $d_c^{old,i}$ to form $d_c^{new,i}$ (Algo. 2: lines 3-6). To mitigate overfitting the old

319

data, data samples in $d_c^{new,i}$ should be limited to a maximum data sample size $MaxDataSize$, where data from the older rounds is excluded from the training dataset in the new rounds (Algo. 2: lines 7-10).

Following data collection, c trains (updates) its local copy of the global model obtained from the last round G^{i-1} using $d_c^{new,i}$ (Algo. 1: line 3). After that, the parameter set of the updated local model L_c^i is encapsulated in the HDF5 file format and saved in a transaction proposal tx_c^i (Algo. 1: line 4). tx_c^i is sent to all the peers $\mathcal{P}$ who evaluate and endorse tx_c^i (Algo. 1: lines 5-6). Afterwards, c packages all the endorsements into a transaction etx_c^i and submits etx_c^i together with its digital signature $signature_c$ to orderers $\mathcal{O}$, who reach consensus on the order of submitted transactions (Algo. 1: line 7) and create the new block $Block_i$ containing all the transactions. After the block $Block_i$ has been committed by peers $\mathcal{P}$ to their copy of the blockchain ledger, client c downloads $Block_i$ (Algo. 1: line 8) from peers, extracts the model parameter sets trained by all the client edge devices from the transactions within $Block_i$, and performs *FedAvg* on the new model parameter sets to generate a new version of the global model G^i (Algo. 1: lines 9-10).

Algorithm 1 Operations of clients $\mathcal{C}$ in round r_i

1: For each client $c \in \mathcal{C}$ in round r_i of R in parallel do:
2: $d_c^{new,i} \leftarrow c.\text{UPDDATASET}(d_c^{old,i})$; ▷ Algo. 2
   ▷ $d_c^{new,i}$ becomes $d_c^{old,i+1}$ in r_{i+1}
3: $L_c^i \leftarrow c.\text{TRAINLOCALMODEL}(G^{i-1}, d_c^{new,i})$;
4: $tx_c^i \leftarrow c.\text{FORMTRANSACTIONPROPOSAL}(\text{HDF5}(L_c^i))$;
5: $c.\text{SENDTOPEERS}(tx_c^i, \mathcal{P})$;
6: $etx_c^i \leftarrow c.\text{GETENDORSEDTRANSACTION}(\mathcal{P})$;
7: $c.\text{SENDTOORDERERS}(etx_c^i, , signature_c, \mathcal{O})$;
8: $Block_i \leftarrow c.\text{BLOCKRETRIEVAL}(\mathcal{P})$;
9: $\{L_c^i, \forall c \in \mathcal{C}\} \leftarrow$
       $c.\text{EXTRACTPARAMS}(\{\text{HDF5}(L_c^i)\} \text{ in } Block_i)$;
10: $G^i \leftarrow c.\text{GLOBALMODELUPD}(\{L_c^i, \forall c \in \mathcal{C}\})$;
       ▷ by *FedAvg*

Algorithm 2 UPDDATASET() Real-time traffic data collection and training data update of client c in round r_i

---

1: **Input:** $d_c^{old,i}$;
2: **Output:** $d_c^{new,i}$;
3: **while** within the data collection period p **do**
4:     $d_c^{in,i} \leftarrow c.\text{COLLECTDATA}()$;
5:     $d_c^{new,i} \leftarrow d_c^{old,i} \cup d_c^{in,i}$;
6: **end while**
7: **if** $d_c^{new,i}.size > MaxDataSize$ **then**
8:     $RemoveSize \leftarrow d_c^{new,i}.size - MaxDataSize$;
9:     $d_c^{new,i}.\text{REMOVEOLDDATA}(RemoveSize)$;
10: **end if**

D. Permissioned Blockchain Network

BFRT adopts a permissioned blockchain network as the FL framework. The permissioned blockchain controls and

manages the FL workflow by processing the transactions tx_c's, for all $c \in \mathcal{C}$ containing the clients' uploaded model parameters (i.e., local model updates L_c's, for all $c \in \mathcal{C}$). Specifically, each transaction in a block consists of an HDF5 file containing the local model parameters for each RSU in the given FL round and the associated metadata (e.g., cryptographic signature). By storing the model updates on the blockchain, the history of FL can be retrieved and reviewed for quality assurance, preservation, auditing, and other purposes.

For the blockchain network, the data structure used for the stored model updates is as follows:

```
type ModelUpdate struct {
    FederatedID        string
    DetectorID         string
    RoundNumber        int
    ModelParameters    HDF5
}
```

The parameters within the $ModelUpdate$ data structure represent the following: $FederatedID$ refers to the name of the associated FL process (e.g., "LSTM TFP I-95"); $DetectorID$ is the identifier for a specific client $c \in \mathcal{C}$ within the blockchain network; $RoundNumber$ indicates the FL round index (i.e. i in r_i) associated with a given HDF5 file; and lastly the $ModelParameters$ field contains the HDF5 file.

The blockchain transaction workflow in *BFRT* is modeled after Hyperledger Fabric v2.2 and separates the transaction endorsement process (execution) from the transaction ordering process (consensus). The endorsement process abides by a configurable policy specifying that a subset of peers must verify and approve the transaction before it can be ordered into blocks and committed to the chain. When a client submits a transaction, it is first sent to endorsing peers. Each peer then simulates the transaction in a containerized sandbox environment, after which the peer returns the endorsed transaction to the client.

After collecting the necessary endorsements, the client forwards the endorsed transaction set to an orderer who enacts the consensus mechanism and packages the pending transactions into a new block. Following consensus, the new block is forwarded to all the peers who validate the transactions to verify each transaction's endorsements and metadata. Once verification is complete, the transactions are committed to the local copy of the blockchain and the client is notified. Lastly, the client retrieves the new block from a peer and executes *FedAvg* to update its local version of the global model.

IV. EXPERIMENTAL RESULTS

A. Experimental Design

1) Setup: The *BFRT* experiments are simulation-based, which were conducted on Google Colab with one NVIDIA P100 GPU, two Intel(R) Xeon(R) CPUs @ 2.30GHz, and 13.34 gigabytes of RAM. All experiments involved 7 clients for both LSTM and GRU federation, and their training samples were dynamically fed in sequence during each round to simulate real-time training. The parameters of G^0 are randomly
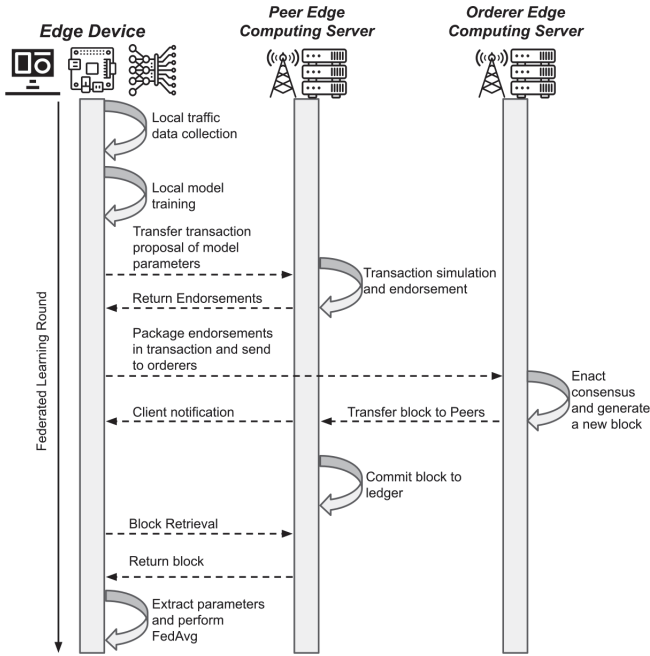
320

Figure 2: Workflow of the *BFRT* system.

initialized in every experiment instead of using a pretrained model. Each client in both LSTM and GRU federations adopts *FedAvg*, with 5 local training epochs per round.

The simulations of blockchain operations were conducted on a virtual machine with 24 gigabytes of RAM and 8 cores of an Intel(R) Core(TM) i9-10900K CPU @ 3.70GHz. We simulate the proposed system of edge devices on Hyperledger Fabric version 2.2, using docker containers to represent the blockchain peers and blockchain orderers. We set resource restrictions on each container in accordance with the system architecture, providing a maximum of 2 gigabytes of RAM to each peer and 4 gigabytes of RAM to each orderer respectively while ensuring peers are provided with 50% of the processor clock cycles given to each orderer container. In our reference deployment, we instantiate 4 peers each and 5 orderers using the *RAFT* consensus algorithm. For all the blockchain experiments, we leverage the Hyperledger Caliper benchmarking tool to measure our deployment's transaction throughput and latency under various transaction loads.

2) Dataset: The dataset used in the experiments is from the Delaware Department of Transportation (DelDOT), which includes traffic flow data collected from DelDOT maintained roadways at a 5-minute time resolution. We select 7 nonsequential loop detectors along the I-95 north arterial to act as the FL clients in *BFRT*. The selected dataset for each detector includes data ranging from the start of August 2019 until the end of September 2019[1].

3) Models: The studies in [33] found that the RNN models, specifically LSTM and GRU, exhibit comparably good inference performance on the traffic flow data from the PeMS

dataset [34] when training offline in a centralized manner. Inspired by this finding, we choose both LSTM and GRU models for real-time FL, termed *LSTM-Fed* and *GRU-Fed*. To highlight the performance of the federated models, we also let each client train its own centralized LSTM and GRU, termed *LSTM-Base* and *GRU-Base*, using the same dataset $d_c^{new,i}$ in round r_i, without federation, as the baseline models. Specifically, each client c in r_i also uses the dynamically updated $d_c^{new,i}$ to train either *LSTM-Base* or *GRU-Base* by 5 epochs as in *BFRT*, but the updated baseline model B^i will continue to be trained in round r_{i+1}, whereas in *BFRT* the new global model G^i will replace the local model of all $c \in \mathcal{C}$ when progressing to round r_{i+1}. The baseline training algorithm is shown in Algorithm 3.

**Algorithm 3** Baseline model training of client c in round r_i

---

1: For any client $c \in \mathcal{C}$ in any round $r_i \in R$:
2: $d_c^{new,i} \leftarrow c.\text{UPDDATASET}(d_c^{old,i});$ ▷ Algo. 2
3: $B_c^i \leftarrow c.\text{TRAINBASELINEMODEL}(B_c^{i-1}, d_c^{new,i});$

All federated and baseline RNN models in our experiments have 12 neurons in their input-layer and 1 neuron in their output-layer, respectively. This corresponds to 1 hour of traffic flow data at a 5-min resolution, which produces a 5-min look ahead prediction. As a result, the sample size of $d_c^{in,i}, i > 1$ is set to equal the $input\_shape$ of the RNN models (i.e., 12), to control data flow by hours. The exception is that $d_c^i.size, i = 1$ is set to 24 (i.e., 2 times of $d_c^{in,i}.size$, where $i > 1$) in r_1 for all $c \in \mathcal{C}$, because the models need at least 13 data samples for training. The result is a total of 1165 rounds for all *BFRT* experiments.

After designing the four models (i.e, *LSTM-Base*, *LSTM-Fed*, *GRU-Base*, and *GRU-Fed*), we perform experiments with $MaxDataSize = 24, 36, 48, 60, 72$ because the sample size is found to influence the prediction accuracy of deep learning models in [35]. In our experiments, $MaxDataSize$ controls the sample size for FL training in each round. Due to the space constraint, we report and compare the performance of the four models with $MaxDataSize = 24$ and 72. Moreover, because the model architecture impacts prediction accuracy, we conducted small-scale FL simulations with varied numbers of hidden layers and neurons prior to training the LSTM models. Based on the error values, we choose 2 hidden layers and 128 neurons for each hidden layer for the LSTM model architecture. For the GRU model architecture, we make our design decisions based on the results of [5], selecting 2 hidden layers with 50 neurons in each. Due to the space limitations, we omit the report for the error values in these tuning experiments. To ensure reproducibility, the simulation code, selected dataset, and all other experimental results are made available in our GitHub repository[2].

---

[1]We separate 80% of the data for real-time training and inference and save the remaining 20% for future work involving offline prediction experiments.
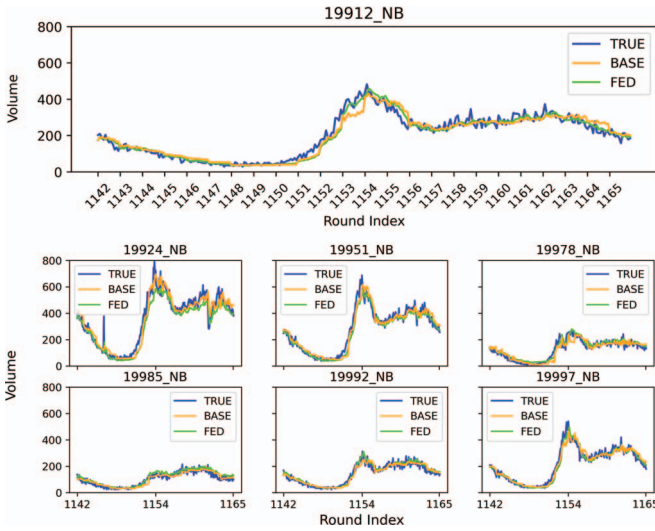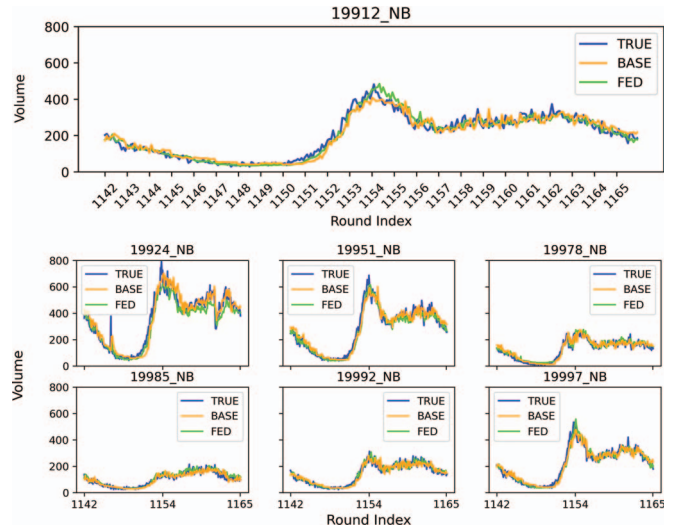
[2]https://github.com/hanglearning/BFRT

321
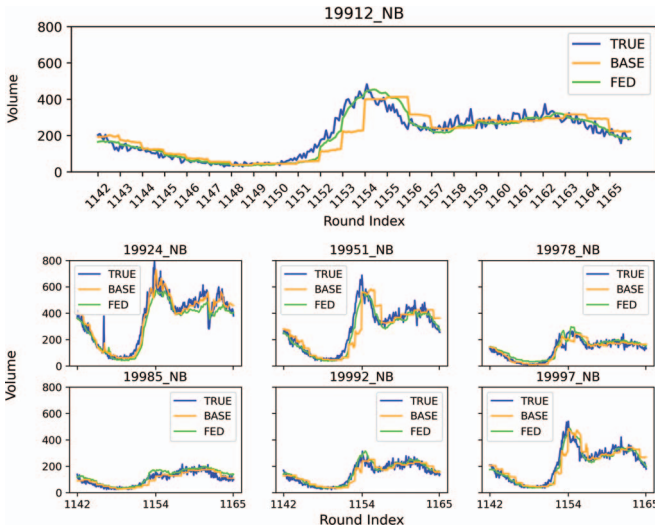
(a) LSTM with $MaxDataSize = 24$
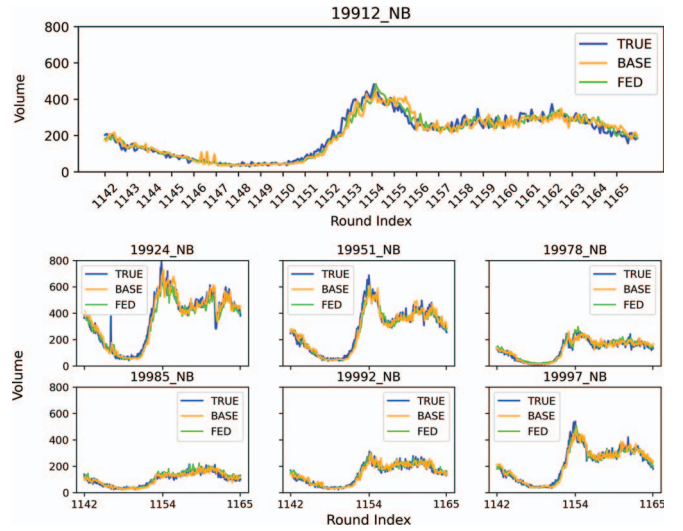
(b) GRU with $MaxDataSize = 24$

(c) LSTM with $MaxDataSize = 72$

(d) GRU with $MaxDataSize = 72$

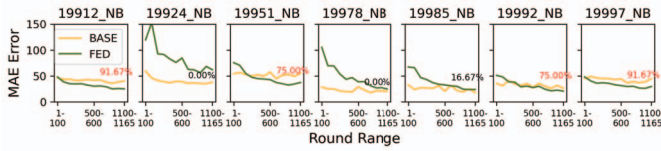Figure 3: Real-time inferences curves of all 7 detectors during the last 24 rounds in *BFRT*

B. BFRT *Performance on Real-time Inferences*

Fig. 3 shows the partial real-time inference curves of the four models for the 7 detectors with $MaxDataSize = 24$ and 72. Out of the inference values in the entire 1165 rounds, we report the last 24 FL rounds which includes 1 day of traffic flow data. Algo. 4 describes how we obtained the inference values (i.e., the values of *BASE* and *FED* in Fig. 3).

For any client $c \in \mathcal{C}$ in round $r_i \in R, i > 1$, a temporary dataset $d_c^{pred,j}$ is first initialized by extracting an $input\_shape$ (i.e., 12 in our experiments, the same as $d_c^{in,i}.size, i > 1$) number of the latest data points from $d_c^{old,i}$ (Algo. 4: line 5). As a special case in r_1, c only performs learning on $d_c^{in,1}$, and therefore the inference starts from r_2. The 5-min look-ahead real-time inference in $r_i, i > 1$ is done by $c \in \mathcal{C}$ using B_c^{i-1} or G^{i-1} to predict on the continuously updated $d_c^{pred,j}$ (Algo. 4: lines 6-12). Specifically, the real-time inference has $input\_shape$ steps (Algo. 4: line 6), which is assumed to be

exactly within the data collection period p. In our experiments, we set p to be 1 hour to be consistent with the $input\_shape$ (i.e. 12) of the models, which in turn results in 12 steps of inference. During each step, c uses B_c^{i-1} or G^{i-1} to predict on $d_c^{pred,j}$, and output a 5-min look-ahead prediction to be added to $BASE_c^i$ or FED_c^i (Algo. 4: lines 7-8). For instance, $d_c^{pred,1}$, which is assigned at Algo. 4: line 5, will be used to output the first prediction of the traffic volume in r_i. Then, c waits to collect one incoming data (i.e., at a 5-min interval in our experiments) and adds it to $d_c^{in,i}$ (Algo. 4: line 9). Notably, Algo. 4: line 6, line 9 and line 10 make up the COLLECTDATA() function appearing in Algo. 2: line 4. After that, $d_c^{pred,j}$ is updated by popping out its oldest data point, and merging with $d_c^{in,i}$ (4: line 11).
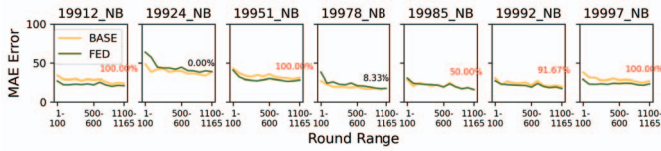
After completing an $input\_shape$ number of prediction steps, the resulted $BASE_c^i$ and FED_c^i will contain the inference values from the corresponding models for $r_i, i > 1$.
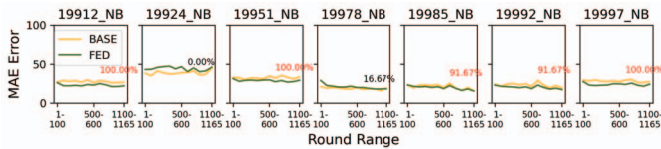
(a) MAE Error of LSTM with $MaxDataSize = 24$
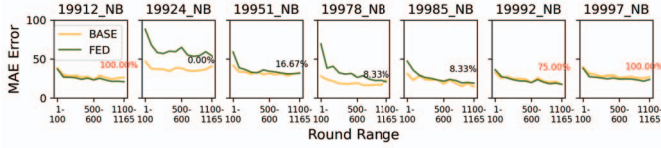
(b) MAE Error of GRU with $MaxDataSize = 24$

(c) MAE Error of LSTM with $MaxDataSize = 72$

(d) MAE Error of GRU with $MaxDataSize = 72$

Figure 4: The MAE errors of real-time inferences of all 7 detectors in all rounds of *BFRT*

Lastly, $d_c^{in,i}$ is assigned to $TRUE_c^i$ (Algo. 4: line 13), which represents the *TRUE* curve(s) in Fig. 3.

In each subfigure of Fig. 3, we picked the detector with the DetectorID *19912_NB* as the focus while keeping the rest of the six detectors' plots small for reference. The x-axis of all the 28 plots indicates the round index for the last 24 rounds, and the y-axis represents the traffic volume. Therefore, for instance, index 1165 on the x-axis represents r_{1165}, and the three curves plotted after index 1165 are $TRUE^{1165}$, $BASE^{1165}$ and FED^{1165}.

1) Baseline models vs. federated models: By comparing the proximity of the *BASE* and the *FED* prediction curves to the *TRUE* data curve for *19912_NB* in Fig. 3a, we observe that *LSTM-Fed* has better overall prediction accuracy than *LSTM-Base* with $MaxDataSize = 24$ from r_{1142} to r_{1165}. This trend is also seen in Table I when comparing the error values between *LSTM-Base* and *LSTM-Fed* for *19912_NB*. Table I denotes four types of error values (i.e. MAE, MSE, RMSE, MAPE) for analyzing predictions resulting from the four models of all 7 detectors with $MaxDataSize = 24$ and 72 during the last 24 rounds. The number following the error type indicates $MaxDataSize$ (e.g., RMSE-72 denotes the RMSE inference error with $MaxDataSize = 72$ over the entire last 24 rounds). The values highlighted in green or beige are the smaller error values between the corresponding baseline and federated models within the relative 2-value groups. For instance, the *LSTM-Fed* MAE-24 error value 23.07 of *19912_NB* is highlighted, to compare with 39.68 to show that *LSTM-Fed* has smaller MAE-24 error compared to *LSTM-Base* for *19912_NB*.

From Table I, it is observed that most of the inferences from the federated models outperform the baseline models. This is especially true for detectors *19912_NB*, *19951_NB*, *19992_NB*, *19997_NB*. For *19985_NB*, the baseline models always outperform the federated models with $MaxDataSize = 24$, whereas the federated models always have smaller errors than the baseline models with $MaxDataSize = 72$. This may be the result of differences in the real traffic flow trend of *19985_NB* compared to the other detectors (*19912_NB*, *19951_NB*, *19992_NB*, and *19997_NB*), due to changes in network topology (e.g., interstate exits or merges) between detector locations. Notably, in Table I, increasing $MaxDataSize$ results in improved accuracy for the federated model on *19985_NB*, highlighting the importance of additional historical data to rectify the difference in data trends. On the other hand, *19924_NB* and *19978_NB* have mixed results. For *19978_NB*, 9 out of the 16 pairs of error values indicate that federated models outperform its baseline models. However, For *19924_NB*, only 6 out of the 16 pairs of error values indicate that federated models outperform its baseline models. This may be the result of a non-recurrent traffic event or recurrent congestion occurring around r_{1160}, where a sharp drop in volume is observed.

In summary, the error values in Table I imply that most federated models outperform their corresponding baseline models in our experiments. However, as model performance is dataset dependent, we plan to perform the comparative experiments using different datasets in future work.

2) LSTM models vs. GRU models: The error values highlighted in green also indicate the smallest error value among all four models for the respective detector. For instance, the *GRU-Fed* MAE-24 error value 19.79 of *19912_NB* is highlighted in green, to compare with the values 39.68, 23.07 and 24.04, and highlight that *GRU-Fed* has the smallest MAE-24 error compared to the three other models on detector *19912_NB*. The total counts of values highlighted in green with respect to the four models for each detector are also summarized in Table II and Table III. As seen in Table II, *GRU-Fed* has the highest smallest error value count for all four types of errors across all the detectors with $MaxDataSize = 24$, whereas *GRU-Fed* and *LSTM-Fed* have a tie with $MaxDataSize = 72$. In conclusion, the *GRU* models generally outperform the *LSTM* models in our experiments.

3) Real-time MAE errors across all 1165 rounds: Similar to reporting error/loss values across training epochs in centralized training, we collected the real-time training errors to examine the model performance as the *BFRT* rounds progress. Fig. 4 shows the normalized MAE errors in 100 round intervals for the four models with $MaxDataSize = 24$ and 72 across all 1165 rounds for all 7 sensors. In all 28 subplots, the x-axis represents round range while the y-axis denotes the

323

Table I: Real-time Inference Errors of the Last 24 Rounds

DetectorID	Model	MAE-24	MAE-72	MSE-24	MSE-72	RMSE-24	RMSE-72	MAPE-24	MAPE-72
19912_NB	LSTM-Base	39.68	22.54	3444.57	992.92	58.69	31.51	0.22	0.13
	LSTM-Fed	23.07	19.82	1076.03	749.4	32.8	27.38	0.13	0.11
	GRU-Base	24.04	23.77	1088.77	919.01	33	30.32	0.15	0.15
	GRU-Fed	19.79	19.91	717.44	738.07	26.79	27.17	0.12	0.11
19924_NB	LSTM-Base	37.11	36.91	2863.12	2577.12	53.51	50.77	0.13	0.13
	LSTM-Fed	50.47	35.88	4761.38	2590.1	69	50.89	0.17	0.11
	GRU-Base	40.14	43.78	2981.27	3441.6	54.6	58.67	0.15	0.17
	GRU-Fed	45.8	41.49	4001.79	3306.1	63.26	57.5	0.15	0.13
19951_NB	LSTM-Base	56.6	28.39	8178.46	1469.3	90.43	38.33	0.22	0.12
	LSTM-Fed	33.16	26.07	2116.6	1288.38	46.01	35.89	0.14	0.11
	GRU-Base	29.19	31.4	1639.82	1669.34	40.49	40.86	0.14	0.15
	GRU-Fed	28.48	27.84	1563.93	1396.44	39.55	37.37	0.12	0.12
19978_NB	LSTM-Base	20.91	16.79	830.04	517.22	28.81	23.9	0.21	0.16
	LSTM-Fed	25.79	16.65	1039.69	466.47	32.24	21.6	0.47	0.26
	GRU-Base	21.81	18.56	877.46	585.14	29.62	24.19	0.26	0.24
	GRU-Fed	20.31	17.62	639.12	515.08	25.28	22.7	0.39	0.29
19985_NB	LSTM-Base	17.11	15.88	512.8	424.23	22.65	20.6	0.19	0.18
	LSTM-Fed	22.18	15.43	787.55	412.16	28.06	20.3	0.25	0.17
	GRU-Base	15.07	14.72	376.12	390.9	19.39	19.77	0.18	0.16
	GRU-Fed	17.2	14.7	479.97	375.67	21.91	19.38	0.2	0.16
19992_NB	LSTM-Base	25.71	20.59	1155.86	730.62	34	27.03	0.22	0.16
	LSTM-Fed	19.94	17.05	718.44	514.14	26.8	22.67	0.16	0.13
	GRU-Base	17.1	18.77	537.46	613.88	23.18	24.78	0.14	0.16
	GRU-Fed	16.72	16.69	489.48	494.51	22.12	22.24	0.14	0.13
19997_NB	LSTM-Base	39.53	22.91	3288.69	968.6	57.35	31.12	0.2	0.12
	LSTM-Fed	25.74	19.76	1305.35	788.8	36.13	28.09	0.13	0.11
	GRU-Base	22.4	23.89	1042.65	994.79	32.29	31.54	0.12	0.14
	GRU-Fed	19.79	21.44	840.6	950.62	28.99	30.83	0.12	0.11

Table II: Best Model Count with $MaxDataSize = 24$

Model	MAE-24	MSE-24	RMSE-24	MAPE-24
LSTM-Base	1	1	1	2
LSTM-Fed	0	0	0	0
GRU-Base	1	1	1	1
GRU-Fed	5	5	5	4

Table III: Best Model Count with $MaxDataSize = 72$

Model	MAE-72	MSE-72	RMSE-72	MAPE-72
LSTM-Base	0	1	1	1
LSTM-Fed	4	3	3	2
GRU-Base	0	0	0	0
GRU-Fed	3	3	3	4

Algorithm 4 Real-time inference of client c in round $r_i, i > 1$

1: For each client $c \in \mathcal{C}$ in round r_i of $R, i > 1$:
2: **Input:** $G^{i-1}, B_c^{i-1}, d_c^{old,i}$;
3: $d_c^{in,i}, BASE_c^i, FED_c^i \leftarrow [], [], [];$ ▷ Empty arrays.
4: $j \leftarrow 0;$
5: $d_c^{pred,j} \leftarrow d_c^{old,i}[: input\_shape];$
     ▷ Extract $input\_shape$ number of the latest data.
6: **while** $j < input\_shape$ **do**
7:     $BASE_c^i.\text{ADD}(c.\text{PREDICTBY}(B_c^{i-1}, d_c^{pred,j}));$
8:     $FED_c^i.\text{ADD}(c.\text{PREDICTBY}(G^{i-1}, d_c^{pred,j}));$
9:     $d_c^{in,i}.\text{ADD}(c.\text{COLLECTONEDATA}());$
10:     $j \leftarrow j + 1;$
11:     $d_c^{pred,j} \leftarrow d_c^{pred,j}.\text{POPLEFT}() \cup d_c^{in,i};$
12: **end while**
13: $TRUE_c^i \leftarrow d_c^{in,i};$

---

normalized MAE error for that round range. There are 12 values on the x-axis, each representing a range of 100 rounds, except the last round representing 65 rounds. For example, the y value corresponding to the x value of 500-600 represents the normalized MAE error comparing $[TRUE^{500}, TRUE^{600}]$ and $[BASE^{500}, BASE^{600}]$ (yellow line), or comparing $[TRUE^{500}, TRUE^{600}]$ and $[FED^{500}, FED^{600}]$ (green line). The percentage value in each plot indicates the percent of 100 round groups where the federated model inference error is lower than the baseline model. The percentage is colored red when the value is greater than or equal to 50%, highlighting that the federated model outperformed the baseline model in that particular experiment.

By comparing Fig. 4a and Fig. 4b, we observe that the *GRU* models exhibit flatter error curves compared to the *LSTM* models. This indicates that with $MaxDataSize = 24$, the *GRU* models have better real-time prediction ability. Also, the *FED* curves of the *GRU* models for *19924_NB*, *19951_NB*, *19978_NB*, and *19951_NB* in Fig. 4b have lower MAE errors in earlier rounds compared to their respective *FED* curves for

the *LSTM* models in Fig. 4a. Additionally, when comparing Fig. 4a and Fig. 4c, and also Fig. 4b and Fig. 4d, we observe that increasing the $MaxDataSize$ from 24 to 72 may improve the real-time prediction accuracy in earlier rounds as the error curves quickly become smooth for all 7 sensors. Lastly, when comparing Fig. 4c and Fig. 4d, we observe that both *LSTM* and *GRU* models have comparably good inference performance. This finding is consistent with the result in Table III. 17 out of the 28 plots have the percentage values in red, implying that the federated models outperform the corresponding baseline models in real-time prediction over the entire 1165 rounds in over 60% of our experiments.

C. Performance of Blockchain Network

In our blockchain experiments, we analyze the operation performance using two metrics: transaction throughput and

(a) READ operations
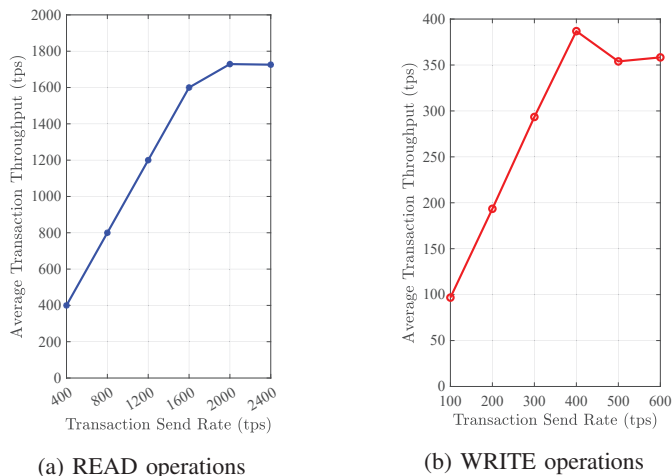

(b) WRITE operations

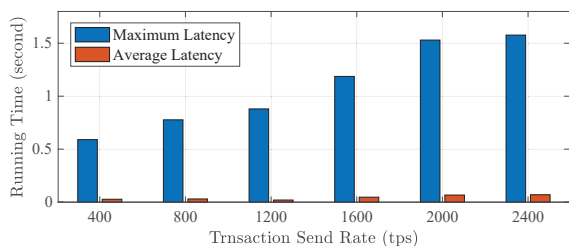Figure 5: Average transaction throughputs for READ and WRITE operations.

Figure 6: Maximum and average transaction latencies for READ operations.

transaction latency. Transaction throughput quantifies the number of transactions per second (TPS) which can be successfully processed by the blockchain network, while the latency indicates the running time of a single transaction from the initial construction by the client until the time it is successfully committed to the ledger. In all the experiments, we analyze both metrics under increasing send rates, where the transaction send rate indicates the number of TPS input by the blockchain clients. Notably, we choose to differentiate the operations of READ and WRITE, as the cost of both operations is not the same, and the selected operation has a notable impact on the network performance.

1) Transaction Throughput: In Figs. 5a and 5b, we illustrate the throughput of our blockchain network under the READ and WRITE operations, respectively. We report the average transaction throughput over multiple testing cycles for quantifying the performance. Fig. 5a shows that the transaction throughput increases as the send rate increases to 2000. However, at send rates above 2000, the performance levels off, and throughput remains relatively constant. This indicates that the maximum network throughput for the READ operation is about 1750 TPS.

Likewise, for the WRITE operation, Fig. 5b illustrates that the network performance begins to stabilize when the peak TPS of 390 is reached at send rate 400. These results highlight that writing is a more expensive blockchain operation than
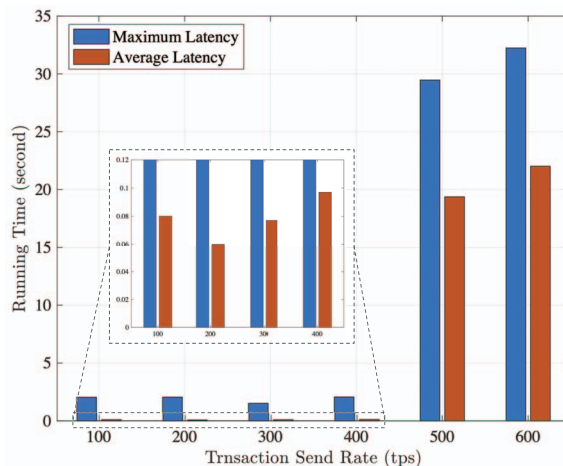

Figure 7: Maximum and average transaction latencies for WRITE operations.

reading. Notably, the performance degrades slightly as the send rate increases beyond 400. One reason for this is the increased processing time of the WRITE operation compared to the READ operation. For the WRITE operation, as the send rate increases beyond the maximum network throughput, transactions will amass in the pending transaction pool causing a bottleneck. This issue is less visible in the READ results, as pending transactions are processed faster.

2) Transaction Latency: Figs. 6 and 7 report both the average and minimum transaction latency for READ and WRITE operations, respectively. In Fig. 6 there is a clear uptrend in the latency when increasing send rate because higher transaction loads on each peer and orderer will increase confirmation times. However, it is notable that even at the highest send rate of 2400 TPS, the average latency value is still only 0.07 seconds. This demonstrates that the blockchain network can perform well for retrieval operations even under considerable transaction loads. On the other hand, Fig. 7 shows severely degraded performance when the WRITE load exceeds 400 TPS, with the average latency value rapidly converging to the maximum latency. This confirms that the blockchain will not perform well in our architecture if the WRITE load exceeds 400 TPS. Comparing both operations, it is clear that even at lower send rates, the average and maximum latency values for the WRITE operation are significantly higher than that of the READ operation.

V. CONCLUSION

This paper proposes *BFRT*, a blockchained federated learning architecture for online traffic flow prediction using real-time data. *BFRT* protects the privacy of underlying traffic data, while also decentralizing computation to the network edge. We prototype both the FL process and blockchain network using a combination of Python and Hyperledger Fabric, and conduct extensive experiments. In our FL experiments, we federated LSTM and GRU models and measured the real-time training and prediction performance using newly collected and

distinct arterial traffic data shards. Additionally, we simulated the edge devices using a Hyperledger Fabric blockchain network with resource-limited docker containers and measured the READ and WRITE performance using the Hyperledger Caliper benchmarking tool. The results show that our FL process and models can generally outperform the centrally trained baseline models, while the permissioned blockchain network can provide high throughput and low latency. For future work, we plan to investigate new methodologies for online multi-output prediction and also experiment with various blockchain architectures to better streamline the FL workflow. We anticipate this work will lay the foundation for future research into real-time traffic flow prediction models.

ACKNOWLEDGMENT

REFERENCES

[1] M. Akhtar and S. Moridpour, "A review of traffic congestion prediction using artificial intelligence," *Journal of Advanced Transportation*, vol. 2021, 2021.

[2] D. Schrank, T. Lomax, and B. Eisele, "2019 urban mobility report," *Texas Transportation Institute, Available: https://static.tti.tamu.edu/tti.tamu.edu/documents/mobility-report-2019.pdf*, 2019.

[3] Y. Lv, Y. Duan, W. Kang, Z. Li, and F.-Y. Wang, "Traffic flow prediction with big data: A deep learning approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 865–873, 2015.

[4] N. G. Polson and V. O. Sokolov, "Deep learning for short-term traffic flow prediction," *Transportation Research Part C: Emerging Technologies*, vol. 79, pp. 1–17, 2017.

[5] Y. Liu, J. James, J. Kang, D. Niyato, and S. Zhang, "Privacy-preserving traffic flow prediction: A federated learning approach," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7751–7763, 2020.

[6] C. Zhang, L. Zhu, J. Ni, C. Huang, and X. Shen, "Verifiable and privacy-preserving traffic flow statistics for advanced traffic management systems," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 9, pp. 10 336–10 347, 2020.

[7] M. A. Azad, S. Bag, S. Parkinson, and F. Hao, "Trustvote: Privacy-preserving node ranking in vehicular networks," *IEEE Internet of Things Journal*, vol. 6, no. 4, pp. 5878–5891, 2018.

[8] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečnỳ, S. Mazzocchi, B. McMahan *et al.*, "Towards federated learning at scale: System design," *Proc. of Machine Learning and Systems*, vol. 1, pp. 374–388, 2019.

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

[10] W. Li, C. Meese, H. Guo, and M. Nejad, "Blockchain-enabled identity verification for safe ridesharing leveraging zero-knowledge proof," in *Proc. of the 3rd IEEE International Conference on Hot Information-Centric Networking (HotICN)*, 2020, pp. 18–24.

[11] W. Li, C. Meese, Z. G. Zhong, H. Guo, and M. Nejad, "Location-aware verification for autonomous truck platooning based on blockchain and zero-knowledge proof," in *Proc. of the IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, 2021, pp. 1–5.

[12] G. Rathee, F. Ahmad, F. Kurugollu, M. A. Azad, R. Iqbal, and M. Imran, "Crt-biov: a cognitive radio technique for blockchain-enabled internet of vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 7, pp. 4005–4015, 2020.

[13] Blockchain. Accessed: 2021-12-02. [Online]. Available: https://en.wikipedia.org/wiki/Blockchain

[14] W. Li, C. Meese, M. Nejad, and H. Guo, "P-cft: A privacy-preserving and crash fault tolerant consensus algorithm for permissioned blockchains," in *Proc. of the 4th IEEE International Conference on Hot Information-Centric Networking (HotICN)*, 2021, pp. 26–31.

[15] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system (white paper)," 2008, accessed: 2020-10-03. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[16] J. Kang, Z. Xiong, D. Niyato, Y. Zou, Y. Zhang, and M. Guizani, "Reliable federated learning for mobile networks," *IEEE Wireless Communications*, vol. 27, no. 2, pp. 72–80, 2020.

[17] H. Chai, S. Leng, Y. Chen, and K. Zhang, "A hierarchical blockchain-enabled federated learning algorithm for knowledge sharing in internet of vehicles," *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[18] Y. Qi, M. S. Hossain, J. Nie, and X. Li, "Privacy-preserving blockchain-based federated learning for traffic flow prediction," *Future Generation Computer Systems*, vol. 117, pp. 328–337, 2021.

[19] "Hyperledger Fabric," accessed: 2020-10-21. [Online]. Available: https://www.hyperledger.org/projects/fabric

[20] "Hyperledger Caliper," accessed: 2020-11-15. [Online]. Available: https://www.hyperledger.org/use/caliper

[21] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, "Advances and open problems in federated learning," *Foundations and Trends® in Machine Learning*, vol. 14, no. 1–2, pp. 1–210, 2021.

[22] C. He, S. Li, J. So, M. Zhang, H. Wang, X. Wang, P. Vepakomma, A. Singh, H. Qiu, L. Shen *et al.*, "Fedml: A research library and benchmark for federated machine learning," *arXiv preprint arXiv:2007.13518*, 2020.

[23] R. Kanagavelu, Z. Li, J. Samsudin, Y. Yang, F. Yang, R. S. M. Goh, M. Cheah, P. Wiwatphonthana, K. Akkarajitsakul, and S. Wang, "Two-phase multi-party computation enabled privacy-preserving federated learning," in *Proc. of the 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID)*, 2020, pp. 410–419.

[24] J.-H. Chen, M.-R. Chen, G.-Q. Zeng, and J.-S. Weng, "BDFL: A byzantine-fault-tolerance decentralized federated learning method for autonomous vehicle," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 9, pp. 8639–8652, 2021.

[25] H. Kim, J. Park, M. Bennis, and S.-L. Kim, "Blockchained on-device federated learning," *IEEE Communications Letters*, vol. 24, no. 6, pp. 1279–1283, 2019.

[26] P. Ramanan and K. Nakayama, "Baffle: Blockchain based aggregator free federated learning," in *Proc. of the IEEE International Conference on Blockchain (Blockchain)*, 2020, pp. 72–81.

[27] C. Ma, J. Li, M. Ding, L. Shi, T. Wang, Z. Han, and H. V. Poor, "When federated learning meets blockchain: A new distributed learning paradigm," *arXiv preprint arXiv:2009.09338*, 2020.

[28] H. Chen, S. A. Asif, J. Park, C.-C. Shen, and M. Bennis, "Robust blockchained federated learning with model validation and proof-of-stake inspired consensus," *arXiv preprint arXiv:2101.03300*, 2021.

[29] A. M. Elbir, B. Soner, and S. Coleri, "Federated learning in vehicular networks," *arXiv preprint arXiv:2006.01412*, 2020.

[30] W. Y. B. Lim, J. Huang, Z. Xiong, J. Kang, D. Niyato, X.-S. Hua, C. Leung, and C. Miao, "Towards federated learning in uav-enabled internet of vehicles: A multi-dimensional contract-matching approach," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[31] T. Zeng, J. Guo, K. J. Kim, K. Parsons, P. Orlik, S. Di Cairano, and W. Saad, "Multi-task federated learning for traffic prediction and its application to route planning," in *Proc. of the IEEE Intelligent Vehicles Symposium (IV)*, 2021, pp. 451–457.

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

[33] R. Fu, Z. Zhang, and L. Li, "Using LSTM and GRU neural network methods for traffic flow prediction," in *Proc. of the 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC)*, 2016, pp. 324–328.

[34] C. Chen, *Freeway performance measurement system (PeMS)*. University of California, Berkeley, 2002.

[35] W. Ng, B. Minasny, W. d. S. Mendes, and J. A. M. Demattê, "The influence of training sample size on the accuracy of deep learning models for the prediction of soil properties with near-infrared spectroscopy data," *Soil*, vol. 6, no. 2, pp. 565–578, 2020.