electronics

MDPI

Article

Enhancing QoS with LSTM-Based Prediction for Congestion-Aware Aggregation Scheduling in Edge Federated Learning

Prohim Tam [1], Seungwoo Kang [1], Seyha Ros [1] and Seokhoon Kim [1,2,*]

[1] Department of Software Convergence, Soonchunhyang University, Asan 31538, Republic of Korea; prohimtam@gmail.com (P.T.); oooksw12@sch.ac.kr (S.K.); seyha@sch.ac.kr (S.R.)
[2] Department of Computer Software Engineering, Soonchunhyang University, Asan 31538, Republic of Korea
* Correspondence: seokhoon@sch.ac.kr

Abstract: The advancement of the sensing capabilities of end devices drives a variety of data-intensive insights, yielding valuable information for modelling intelligent industrial applications. To apply intelligent models in 5G and beyond, edge intelligence integrates edge computing systems and deep learning solutions, which enables distributed model training and inference. Edge federated learning (EFL) offers collaborative edge intelligence learning with distributed aggregation capabilities, promoting resource efficiency, participant inclusivity, and privacy preservation. However, the quality of service (QoS) faces challenges due to congestion problems that arise from the diverse models and data in practical architectures. In this paper, we develop a modified long short-term memory (LSTM)-based congestion-aware EFL (MLSTM-CEFL) approach that aims to enhance QoS in the final model convergence between end devices, edge aggregators, and the global server. Given the diversity of service types, MLSTM-CEFL proactively detects the congestion rates, adequately schedules the edge aggregations, and effectively prioritizes high mission-critical serving resources. The proposed system is formulated to handle time series analysis from local/edge model parameter loading, weighing the configuration of resource pooling properties at specific congestion intervals. The MLSTM-CEFL policy orchestrates the establishment of long-term paths for participant-aggregator scheduling and follows the expected QoS metrics after final averaging in multiple industrial application classes.

Keywords: congestion-aware collaborative learning; edge federated learning; industrial applications; quality of service; long short-term memory

check for updates

Citation: Tam, P.; Kang, S.; Ros, S.; Kim, S. Enhancing QoS with LSTM-Based Prediction for Congestion-Aware Aggregation Scheduling in Edge Federated Learning. *Electronics* **2023**, *12*, 3615. https://doi.org/10.3390/electronics12173615

Academic Editors: Irida Shallari and Mattias O'Nils

Received: 31 July 2023
Revised: 22 August 2023
Accepted: 25 August 2023
Published: 27 August 2023

Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

1.1. Motivation and Problem Statement

Applied artificial intelligence (AI) for real-time image processing applications in the industrial Internet of Things (IIoT) within 5G and beyond networks has the potential to gain popularity due to its remarkable performance outcomes [1,2]. Advancements in end devices, the quality of data, sensing capabilities, and computing adequacy, together enable comprehensive services to achieve compelling performance targets. Although application services benefit from real-time considerations, the requirements for ultra-reliable low-latency performance are crucial. These requirements involve efficient utilization of support paradigms in both access and core networks, including new radio interfaces, green communications [3,4], multiple-input multiple-output [5], software-defined networking (SDN) [6], multi-access edge computing (MEC) [7], network functions virtualization (NFV) [8], etc. Simultaneously, despite various enablers and framework supports, challenges persist within the IIoT, particularly in the era of big data generated from massive structured and unstructured data sources. These challenges include accommodating multiple types of vertical and horizontal services on a single platform, managing the heterogeneity of standard and proprietary devices, and handling extensive interfaces [9,10]. Given these circumstances,

the application of AI in the IIoT is motivated to extend into edge IIoT networks that aim to mitigate backbone congestion, enhance energy efficiency, and optimize the utilization of network resources.

Edge AI enables intelligent applications through low-latency model learning and enhanced privacy protection. By combining MEC with deep learning (DL), the concept of edge intelligence is well established, particularly for industrial imaging and visual-based platforms. It offers agile multi-service responses and efficient utilization of resources in caching, communication, and computation [11–13]. However, the transmission of raw image/video data batches to the input gate significantly impairs fronthaul performance. Stringent privacy regulations and the inaccessibility of private data present challenges in accessing end-sensing data nodes. As a result, a privacy-enhanced framework becomes a crucial prerequisite to facilitate the advancement of edge AI model development.

Federated learning (FL) was conceptualized in 2016 [14], offering a promising solution to tackle the challenges related to communication costs, data privacy, and regulatory compliance. Edge federated learning (EFL) in wireless IIoT communications represents a cutting-edge collaborative AI framework designed to address industrial image processing learning models [15]. By harnessing resources in close proximity to user equipment within distributed areas, EFL is explored to optimize battery consumption for local IIoT devices and mitigate the need for direct local–global round communications [16–18]. In the context of 5G core networks, 3GPP and ETSI drive the utilization of MEC resources and interaction with network data analytics function (NWDAF) to facilitate AI-driven execution services. Through a service-based architecture, network functions and data exposure capabilities provide insightful information for comprehensive global network analytics. Additionally, supported by SDN/NFV-enabled systems, the network application programming interface (API) streamlines request procedures [19–23]. Nevertheless, despite these advancements, the diversity among EFL-based service types gives rise to challenges. The tasks of prioritizing service classes, harmonizing update aggregation policies, and accurately predicting congestion become intricate issues demanding real-time solutions.

Regarding congestion predictions, ML/DL-based methods are given significant consideration. In [24], the authors monitored anomalous traffic as an illustrative use case of ML tasks within network operations; moreover, the authors developed formulations for transmission latency, bandwidth congestion, and the accuracy of ML tasks while incorporating edge–cloud collaboration. The proposed method optimized the data preprocessing ratio between edge servers and cloud servers, considering network bandwidth constraints and congestion, as well as the real-time function of anomalous traffic detection. Additionally, the domain of prediction modelling has acknowledged the effectiveness of modified or optimized long short-term memory (LSTM) techniques [25,26]. Within the context of communication networks, the motivations behind employing LSTM can be categorized into three main perspectives, including sequential dependencies with variable/temporal patterns, adaptive learning, and enhanced prediction accuracy with proactive capability.

In [27], the authors presented the problem of massive traffic and proposed a novel approach to network traffic prediction using LSTM and transfer learning. By transferring knowledge from a source domain to a target domain, the method enhanced predictions in low-data scenarios, addressing overfitting and data scarcity. The LSTM-based approach extended transfer learning's application, established a prediction architecture, and improved accuracy, even with limited time series datasets. Furthermore, the LSTM-based approach is also studied in throughput prediction for minimizing the delay of mission-critical services [28]. The authors investigated ML and DL integration for accurate throughput prediction and introduced a modified LSTM-based prediction system with an attention mechanism. The system used TCP log data traces and achieved lower loss in predicting throughput compared with existing methods (e.g., LSTM without attention).

1.2. Paper Contributions

This paper delves into the algorithm designs of modified LSTM-based congestion-aware EFL (MLSTM-CEFL) to address the aforementioned problem statements, structuring the solutions into three contributing phases as follows:

1. The system architecture is given to handle the virtualization approach for orchestrating resource pooling properties within the NFV infrastructure (NFVI). The framework divides the plane into participants, edge aggregators, and controllers, strengthened by a modified LSTM (MLSTM) algorithm for predicting congestion levels. The structural planes enhance interactivity and interface connections to facilitate scalability for the multi-service IIoT model aggregation.
2. MLSTM-CEFL is introduced for collecting feature inputs, predicting congestion rates, determining high-impact latency-efficient conditions, and optimizing the final model learning objectives. The procedural flow of feature collection and EFL-based IIoT model communications are presented in this paper.
3. The proposed reliable model aggregation policy is presented by considering the output of quality of service (QoS) guarantees and orchestration capabilities in configuring proactive congestion detection and prioritizing model flows. The simulation is implemented with DL policy modelling in a TensorFlow and Keras-based environment and network topology in Mininet and MiniNFV.

1.3. Paper Organization

This paper is organized as follows. Section 2 outlines the related studies. Section 3 presents the proposed approach, including the system architecture, algorithm designs, and congestion prediction algorithms for efficient orchestration. The experiment and analysis are provided in Section 4, covering the simulation environment, QoS evaluation metrics, result, and discussion. Finally, Section 5 concludes the paper.

2. Related Works

This section provides complementary studies that highlight the key contributions of LSTM in network prediction and the applicability of (edge) FL integration. FL has been applied to enhance industrial applications in various aspects, such as privacy-preserving schemes, massive data management, model offloading decisions, and resource efficiency.

2.1. LSTM-Based Prediction for QoS Enhancements

The prediction of network traffic has gained significance in contemporary scenarios, such as anomaly control, congestion regulation, and bandwidth management. In [29], a framework utilizing LSTM for network traffic prediction was proposed by employing real network traces and aiming for predictions within very short timeframes. To tackle the diversity of network traffic, a feature-based clustering method was employed as a preprocessing stage to group similar time series. The experimental results indicated that the LSTM-based framework performed better in predicting network traffic with minimal errors. In [30], the effectiveness of LSTM in predicting mobile traffic was studied. The authors used datasets that contained multivariate traffic information directly from the LTE control channel's downlink control information. This paper aimed to enhance QoS by focusing on network delays, which include sending, propagation, processing, and queuing delays. In [31], a model named Nefopam was introduced for predicting network flow delay. It combined LSTM and graph convolutional neural networks to effectively capture the spatiotemporal attributes of network flow data. Nefopam incorporated three components (recent, daily cycle, and weekly cycle) to characterize the temporal and spatial aspects. These component outputs were integrated for the final prediction result.

2.2. Applicability of LSTM with (Edge) FL and IIoT Applications

The architecture of the IIoT-FL framework is provided to illustrate the collaborative procedures involved in model training and inference between local IIoT nodes and the

parameter server (see Figure 1). In the primary framework, (edge) FL policy can be jointly optimized by multi-agent deep Q-networks in terms of resource allocation and computation offloading [32]. However, QoS-specific prediction can be separately studied and integrated for improving the congestion-aware control platform.

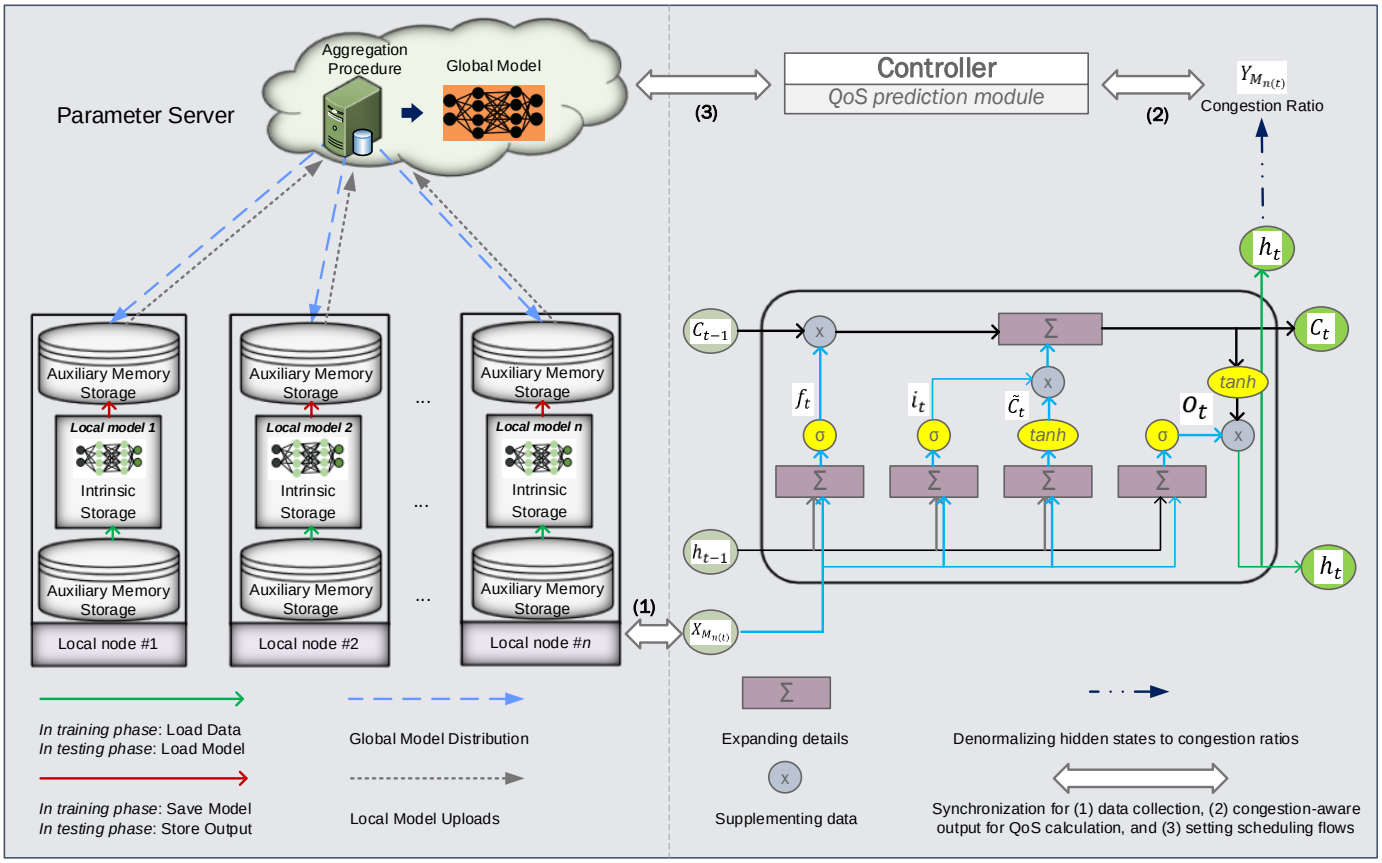

Figure 1. Architecture of IIoT-FL framework with LSTM-based congestion-aware aggregation.

LSTM-based prediction can be deployed to detect the congestion rates of local model updates and optimize the aggregation scheduling. In the initial phase, the global model is initialized with primary parameters and hyperparameters, and it is subsequently distributed to the local nodes. To participate in the FL execution, an IIoT node needs to have auxiliary memory and intrinsic storage for tasks such as (1) loading data from local storage, (2) computing the local model, (3) saving the model to local auxiliary memory storage, (4) loading the saved model for testing, and (5) storing the final model output [15,33]. Loss optimization occurs at each local IIoT node. The optimal local models are then uploaded to the parameter server for the aggregation process. The congestion-aware scheduling (using the LSTM-based approach) assists the policy settings to avoid model drops. Through this procedure, the integration of FL brings forth several key advantages for deploying distributed AI models in IIoT services, including personalized data protection, service differentiation, efficient low-latency communications, and optimization of AI model learning [34–37].

In the context of controlling FL networking states, an LSTM-based controller can be used to model and manage the dynamic flows of model updates and connections, due to its capability of learning and predicting patterns, adapting to changing conditions, and making decisions based on the history of states and aggregation procedures. In Figure 1, there are key components in LSTM for FL, including: input data $\left(X_{M_{n(t)}}\right)$, input gate (i_t), forget gate (f_t), output gate (o_t), cell state $(\tilde{C}_t)$, new cell state (C_t), hidden state (h_t), and congestion

rate prediction $Y_{M_{n(t)}}$. $M_{n(t)}$ represents the set of local loss-minimized model parameters from participant n at t-index round communications. With LSTM-based settings, the FL controller can be significantly improved with proactive hidden state information to predict the scheduling load congestion.

In [38], a privacy-preserving scheme was proposed to enhance the security of local model updating procedures within FL model aggregation. The proposed scheme served to mitigate the impact of malicious participants and strengthen the flexibility of participant selection. Furthermore, considering the expanding scale of IIoT equipment, ref. [39] proposed the management of data by converging deep reinforcement learning with FL to handle the training of extensive datasets. By converting raw data into model parameters, FL optimizes offloading decisions for IIoT tasks and directs them towards optimal edge aggregators equipped with sufficient serving resources. Consecutively, the utilization of fronthaul communication resources is minimized.

From an edge perspective, an MEC-assisted FL framework was coupled with a digital twin to tackle computation offloading and resource allocation problems [40]. MEC was employed to mitigate the resource limitations of local participants and helped the core network congestion. The proposed framework greatly optimized the key performance indicators in system costs and FL training efficiencies. Furthermore, QoS-centric resource allocation in FL policy was also studied over edge IIoT networks for minimizing delay, energy consumption, and cost expenditure [41]. The authors separated the sequential and concurrent mechanisms by tackling the diversity of IIoT devices and obtaining the Nash bargaining solutions. As a result, EFL serves as a significant mechanism to enhance QoS and other learning metrics by leveraging edge computing capacity to assist in offloading decisions and resource allocation [42].

3. Congestion-Aware EFL-Based Model Aggregation Policy in Edge Computing

3.1. System Architecture

In this section, we present the system architectures used to implement the proposed software-defined approach, which are organized into three tiers as follows:

1. **Participant tier** refers to local devices with image datasets, computing models, and service type indicators.
2. **Edge tier** involves a heterogeneity of multi-service image processing for industrial applications, model labels/features, and allocatable virtual capacities.
3. **Controller tier** comprises three primary processing phases. First, the modified LSTM congestion prediction (MLSTM-CP) module assists in predicting congestion levels. Next, the offloading decision-maker module executes to facilitate the selection of the optimal edge aggregator for offloading. Lastly, the policy installation for model aggregation concludes the tier's functionalities. Comprehensive descriptions of the primary notations are given in Table 1.

3.1.1. Participant Tier

In the proposed EFL framework, participants are expected to fulfill three fundamental requirements, outlined as follows:

1. **Adequate local resources**: Participants need to ensure the connectivity, energy capacities, and model computing capabilities within both auxiliary and intrinsic memories.
2. **High-quality data input**: The integrity and reliability of the data are required in contributing to the overall performance of our framework.
3. **Non-malicious intent**: This requirement ensures the security and trustworthiness of the collaborative learning environment for all selected participants.

Meeting these prerequisites grants participants the authorization to engage in collaborative training and be considered for selection when scheduling interactions with the global server. Let $N = \{1, 2, \ldots, n\}$ denote the set of selected IIoT participants. As each participant is associated with a specific image processing service, we collect the corresponding features

of m services between the participant and aggregator by denoting $Y_n^m = \{Y_n^1, Y_n^2, \ldots, Y_n^m\}$ to participant n. $M_{n(t)}$ is based on individual data contributions $D_{n(t)}$ and linked to a unique service type identified as Y_n^m.

Table 1. Primary notations and descriptions.

Notation	Description
$N = \{1, 2, \ldots, n\}$	Set of local IIoT participants
$E = \{1, 2, \ldots, e\}$	Set of edge aggregators with specified resource pooling properties
$Y_n^m = \{Y_n^1, Y_n^2, \ldots, Y_n^m\}$	Set of m services for participant n
$M_{n(t)}$	Set of local loss-minimized model parameters from participant n at t-index round communications
$M_{e(t)}$	Loss-minimized model parameters in edge aggregator e at round t
$D_{n(t)}$	Data distribution of participant n at round t
$pt\left[M_{n(t)}\right]$	Size-based processing time of local model $M_{n(t)}$
$tl\left[M_{n(t)}\right]$	Tolerable latency of local model $M_{n(t)}$
$sc\left[M_{n(t)}\right]$	Service criticality of local model $M_{n(t)}$
$ts\left[M_{n(t)}\right]$	Time slot of local model $M_{n(t)}$ at edge aggregator
$bs\left[M_{n(t)}\right]$	Buffer size of adjusted edge aggregator that loaded $M_{n(t)}$
$qi\left[M_{n(t)}\right]$	Queueing index of $M_{n(t)}$ at current time slot at e
τ_m	Upper-bound QoS (e.g., delay) for service m
$T[M_m^{G(e)}]$	Latency of averaging service m toward final model in e
$T_{(n)}^{comp}$	Computation latency at participant n
$T_{(e)}^{comm}$	Communication latency between participants to edge aggregator e
$T_{(e)}^{comp}$	Computation latency in edge aggregator e at round t
T_{update}	Uplink data rate for updating the local model
$\left(X_{M_{n(t)}}, Y_{M_{n(t)}}\right)$	Input data and target features
$\left(X_{M_{n(t)}}^{tr}, Y_{M_{n(t)}}^{tr}\right)$	Training input data and target features
$\left(X_{M_{n(t)}}^{te}, Y_{M_{n(t)}}^{te}\right)$	Testing input data and target features
$N(m)$	Number of models
$B(s)$	Batch sizes
$D(u)$	Dense units
$MLSTM(u)$	MLSTM units
$N(e)$	Number of epochs
W (*e.g.*, W_{xi})	Sampling weights (e.g., weight matrices that determine how the input data $X_{M_{n(t)}}$ contribute to the computation of input gate i)
b (*e.g.*, b_i)	Bias (e.g., bias terms associated with the input gate i)
$(\sigma, \tanh)$	Sigmoid and tanh activation functions
i_t, f_t, o_t, $\widetilde{C}_t$, C_t, and h_t	LSTM components: input gate, forget gate, output gate, cell state, new cell state, and hidden state
$\odot$	Element-wise multiplication

3.1.2. Edge Tier

In our proposed framework, the adaptive allocation of edge resources is facilitated through the implementation of an NFV-enabled system. Virtualization procedures are leveraged to configure resource attributes in a manner that caters to varying criticality levels associated with each slicing service. Specifically, the image processing services are categorized into three distinct classes. A comprehensive description of these class attributes, criticalities, and application scenarios when FL is applied to industrial services can be found in Table 2. Class-1 represents a high mission-critical scenario, where unsatisfactory QoS performance can lead to significant consequences. Instances of this class include the modelling of visually assisted robots and low-latency manufacturing control systems. Class-2 denotes a mission-critical context with an upper-bound tolerable delay of 150 ms.

This class is applicable to FL-based industrial visual or surveillance modelling systems. For the low mission-critical category, Class-3 is characterized by a delay tolerance of 300 ms. QoS class indicators (QCI) define the characteristics of each class, aligning with QCI priority assurance in the recently released standardization. Each indicator specifies class characteristics based on example service's packet delay budget (PDB) for different scenarios: (1) vehicle-to-everything (V2X) messages or real-time gaming, (2) live streaming or mission-critical user plane, and (3) non-mission-critical user plane or non-conversational video. Although the specification provided is designed for a single use-case implementation, serving as a simulation template, we set each value as dynamic and adjustable within the proposed algorithm procedure. The upper-bound maximum delays of each class are denoted as τ_m, which are multi-dimensional containers corresponding to the various service types (m).

Table 2. The class characteristics, criticalities, and FL-based scenarios in industrial services.

Conditions	Upper-Bound Delays	Criticalities	FL-Based Scenarios
Class-1	50 ms	High mission-critical	Visual-assisted robot, low-latency manufacturing control
Class-2	150 ms	Mission-critical	Industrial visual/surveillance systems
Class-3	300 ms	Low mission-critical	Imaging compression for storage and retrieval

3.1.3. Controller Tier

In this sub-section, we delve into the controller and virtualization layers, addressing their roles in adapting virtual edge model storage, network functions, and computing capacities in accordance with the proposed policy. The controller functions as an integrated virtualized infrastructure manager (VIM), overseeing the orchestration between pooled virtualization and physical resources. To efficiently apply policies to the primary controller, the incoming local model M_n with defined Y_n^m is channeled into a subordinate processing module. This module pre-calculates metrics such as size-based processing time ($pt\left[M_{n(t)}\right]$), tolerable latency ($tl\left[M_{n(t)}\right]$), and service criticality ($sc\left[M_{n(t)}\right]$) for the given model. These metrics enhance the hidden states of processing speed, latency tolerance, and service criticality. The resulting time series of model loading durations are stored for the purposes of congestion control and adaptive configuration adjustments. Within the subordinate module of each virtual EFL server, specific parameters are retrieved for the selected participant's models $M_{n(t)}$, including the time slot ($ts\left[M_{n(t)}\right]$), buffer size ($bs\left[M_{n(t)}\right]$), and queuing index ($qi\left[M_{n(t)}\right]$). The observation of these parameters is enabled through SDN interfaces integrated into the system architecture. The gathered network status conditions at the current t-index are appended to the policy orchestration. Through the formulation of predictive models, the congestion levels are translated into three subsequent threads of action as follows:

1. During non-congestion states, the offloading decision-maker modules adhere to the current policy, retaining incoming local models.
2. In cases of congestion, the scheduling and aggregation destinations for that specific time slot are altered by modifying forwarding rules within the primary controller.
3. For heavy congestion, adjustments are made to the virtual resource placement properties within each functional instance's descriptor. This adaptation is guided by the proposed policy and prioritized service types at that time slot.

3.2. Primary Objectives and Algorithm Designs

A reliable model aggregation policy with MLSTM-CEFL offers four primary objectives in industrial EFL-based applications, which include minimizing model aggregation latencies, preventing excessive model drops, ensuring efficient utilization of model computing resources, and maximizing model communication throughput. In multi-service industrial

applications, MLSTM-CEFL distinguishes between each application class through adaptive resource pooling modifications. Consequently, the scheduling policy and offloading decisions are formulated based on these four primary outputs.

To minimize the overall latencies, the three-tuple elements are jointly considered, including the latencies of local model computation, model communication, and edge aggregation. The local computing time is considered based on the size of input data batch $D_{n(t)}$ to complete the $M_{n(t)}$. The final local learning model at t-index is the optimal model parameter that minimizes the loss of each mini-batch input. Once the optimal loss-minimized $M^*_{n(t)}$ is acquired, the updating process in a time slot ts, denoted as T_{update}, is executed across all IIoT participant nodes. The models are directed towards a selected edge aggregator. This process is formulated at the t-index by summing up the model updating latencies of each local node. Upon receiving the local models for the t-index, the edge aggregation procedure is performed to minimize the edge loss function within each matched industrial imaging service. Given the distinct requirements of each service for model construction and target features, traditional federated averaging algorithms are employed in this design to aggregate service m iteratively, culminating in the final learning model denoted as $T\left[M_m^{G(e)}\right]$. Equations (1)–(3) are presented as follows: (1) the consumed latencies of local model computing from a single participant, (2) model communication between a set of matched participants–edge in a single time slot, and (3) edge computing for the aggregated model in a single time slot. The final objective is provided in Equation (4). $T^{comp}_{(n)}$, $T^{comm}_{(e)}$, $T^{comp}_{(e)}$, and T_{update} denote the following descriptions: (1) computation time in the local IIoT participant n, (2) communication time between a set of participants and a single edge aggregator e, (3) computation time in edge aggregator e, including global loss minimization and federated averaging, and (4) uplink data rate for model updating under specific IIoT network conditions. Additionally, L represents the loss function of imaging or visual-assisted model training in industrial applications capable of utilizing the mean squared error to calculate gradients within each specified convolutional neural network model.

$$T^{comp}_{(n)}\left[M^*_{n(t)}\right] \triangleq T[\underset{M_{n(t)}}{\operatorname{argmin}} L\left(M_{n(t)}\,|D_{n(t)}\right)] \tag{1}$$

$$T^{comm}_{(e)} \triangleq \sum_{ts}\sum_{n} T_{update}[M^*_{n(t)}] \tag{2}$$

$$T^{comp}_{(e)} \triangleq T[L(M_{e(t)}\Big|\textstyle\sum_{n=1}^{N} D_{n(t)})] + T[M_m^{G(e)}] \tag{3}$$

$$min\left(\sum T^{comp}_{(n)} + T^{comm}_{(e)} + T^{comp}_{(e)}\right)_m \leq \tau_m \tag{4}$$

High model drops significantly degrade the final learning accuracy within EFL-based applications, primarily due to the absence of high-quality data contributions and incomplete training procedures. To prevent this issue, the priority and congestion detection are pivotal to consider with the high-impact features of τ_m and output of MLSTM-based estimation. Moreover, to enable resource-efficient model computing, each edge aggregator requires elastic resource placement orchestrated through virtualized and softwarized management entities such as NFV orchestrator (NFVO), virtual network function (VNF) manager (VNFM), and VIM. By allowing modifiable properties in each instance descriptor, NFVI consists of efficient virtual mapping allocation that is extensively controlled by an SDN-enabled controller connected to the proposed modules. Addressing throughput-maximized model communications involves activating model updates as opposed to sharing raw data. Moreover, real-time scheduling is further extended in the following section.

As illustrated in Figure 2, the interactivity of three main tiers in our proposed IIoT-EFL architecture is given, including participant, edge, and controller tiers. Each tier can be described as follows:

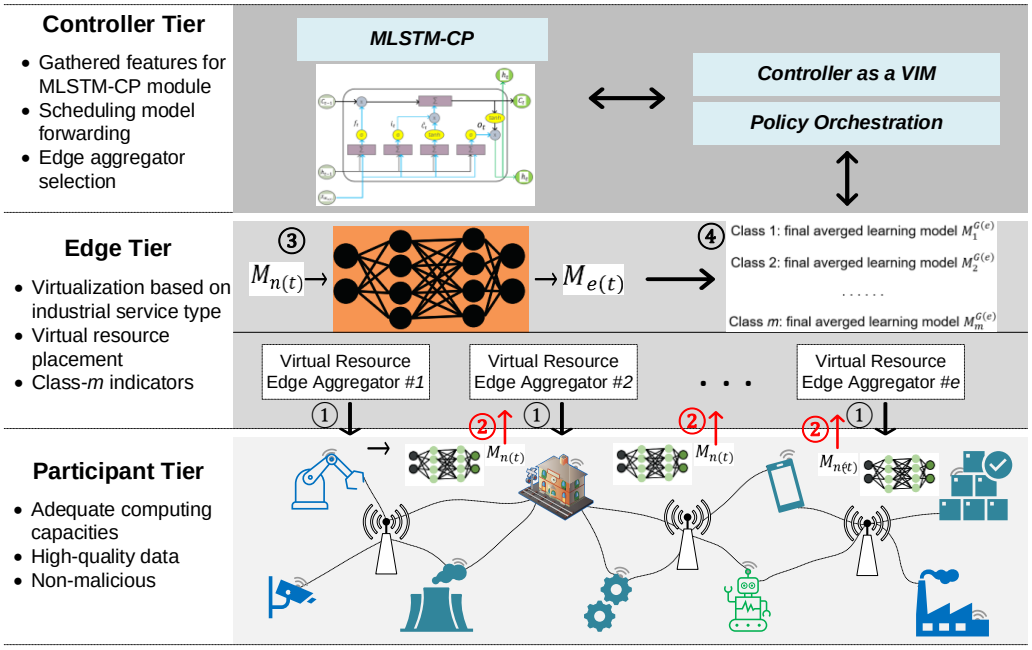

Figure 2. The interactivity of three main tiers in our proposed IIoT-EFL architecture.

- **Participant tier**: The selection criteria are listed, including the non-malicious user status, sufficiency of resource capacities, and high-quality image sensing contributions. Real-time wireless data planes in SDN-enabled architectures involve various end-user equipment, including robotic image sensing nodes, manufacturing nodes, surveillance videos, and tracking/monitoring applications.
- **Edge tier**: Virtual resource edge aggregators adapt to the formulated industrial service type and class-m prioritization orchestrated through the virtualization layer. This orchestration directly follows the controller tier. $M_{n(t)}$ values are loaded into global loss minimization processes directed towards the edge-aggregated parameters $M_{e(t)}$.
- **Controller tier**: With multi-services in IIoT-EFL-based applications, the final averaged learning model $M_m^{G(e)}$ for each application type is independently computed following the scheduling policies of the defined congestion states from MLSTM-CP output and controller-based orchestration in VIM.

The procedural flow of EFL-based IIoT applications entails four fundamental steps:

1. Distributing model structures and hyperparameters for each matching service type between distinct aggregators e and local nodes n.
2. After obtaining the primary model, local computing optimizes model parameters using local data, followed by uploading the refined model parameters back to the corresponding edge aggregator.
3. By collectively gathering multi-type local model parameters, the edge-aggregated models are executed and subsequently categorized based on their respective service types. This categorization sets the stage for the final averaging process during the last-index iteration.
4. Each service class computes the final learning model by dynamically adopting the aggregation and scheduling policies outlined by the proposed controller.

3.3. MLSTM-CP and Orchestration of EFL Model Aggregation

In the MLSTM-CP module, the forget, input, and output gates collaboratively engage with both the input data and target features ($X_{M_{n(t)}}$, $Y_{M_{n(t)}}$). Equation (5) presents the set of general features for input, which are gathered through the proposed SDN/NFV-enabled network API and RESTful API within the system architecture. The feature information

is captured within the local participant, the updated FL model, and the queuing process at the selected edge aggregator. The deployment of a model-task profiler and inspection techniques are set. The purpose of each component in our proposed MLSTM-CP can be described as follows:

- **Input data ($X_{M_{n(t)}}$)**: At each time step, the input data is fed into the MLSTM cell, and the gates and operations process this input to determine how much new information should be integrated, how much previous knowledge should be retained, and how the hidden state of the model should be updated based on the evolving conditions of the industrial process. The collected features are from various edge devices/servers deployed in the industrial setting (see (5)).

$$X_{M_{n(t)}} = \left\{ pt\left[M_{n(t)}\right],\ tl\left[M_{n(t)}\right],\ sc\left[M_{n(t)}\right], ts\left[M_{n(t)}\right], bs\left[M_{n(t)}\right], qi\left[M_{n(t)}\right] \right\} \quad (5)$$

- **Input gate (i_t)** represents the decision-making process for how much new information from the current participants–edge data should be incorporated into the model update (see (6)). Different edge devices might have varying degrees of relevancy or noise in their data due to the conditions of the industrial processes that are being monitored.

$$i_t = \sigma\left(W_{xi} X_{M_{n(t)}} + W_{hi} h_{t-1} + b_i\right) \quad (6)$$

- **Forget gate (f_t)** symbolizes the importance of retaining experienced information from previous participants–edge updates (see (7)). f_t assists our system to determine how much to retain from past states in order to make better decisions;

$$f_t = \sigma\left(W_{xf} X_{M_{n(t)}} + W_{hf} h_{t-1} + b_f\right) \quad (7)$$

- **Output gate (o_t)** acts as the filter that decides how much of the current hidden state should be exposed and utilized for influencing the global model. o_t reflects how much the insights and knowledge gained from a particular participant–edge update experience should contribute to the comprehensive learning process (see (8)).

$$o_t = \sigma\left(W_{xo} X_{M_{n(t)}} + W_{ho} h_{t-1} + b_o\right) \quad (8)$$

- **Cell state ($\widetilde{C}_t$)** acts as the "memory" of the MLSTM, which captures the accumulation of insights and patterns from various participants–edge data and updating schedules. $\widetilde{C}_t$ retains essential information about the evolving system dynamics. The updated cell state, denoted as C_t, is the result of combining between (1) the previous cell state (C_{t-1}) that was preserved based on the forget gate and (2) the new candidate cell state ($\widetilde{C}_t$) that was incorporated based on the input gate (see (9) and (10)).

$$\widetilde{C}_t = tanh\left(W_{xc} X_{M_{n(t)}} + W_{hc} h_{t-1} + b_c\right) \quad (9)$$

$$C_t = f_t \odot C_{t-1} + i_t \odot \widetilde{C}_t \quad (10)$$

- **Hidden state (h_t)** represents the encoded information that the MLSTM uses to influence the model update process (see (11)). In other words, h_t acts as the output of the MLSTM cell and encapsulates the input pattern understanding of how the various participants–edge updates contribute to the overall improvement of the model.

$$h_t = o_t \odot \tanh(C_t) \quad (11)$$

- **Returned congestion rate ($Y_{M_{n(t)}}$):** h_t and o_t components assist the output features by encapsulating the distilled knowledge and encoded patterns learned from $X_{M_{n(t)}}$ over multiple timesteps. o_t controls how much of h_t is exposed and utilized to contribute to the output features as congestion rates, which in turn influence the scheduling awareness of global model updates (see (12)).

$$0 \leq Y_{M_{n(t)}} \leq 1 \tag{12}$$

By obtaining MLSTM-CP outputs, the orchestration of resource virtualization is modified. The proposed EFL model aggregation policies consider the congestion rate of the current time slot by evaluating the total incoming model services and prioritizing the optimal path for each level of criticality. To create a forwarding graph, the SDN primary controller is used to determine whether to alter or maintain the current edge aggregator destination. To configure instance properties, the NFV MANO-enabled interface facilitates interactivity from the outputs of the predictive module towards virtual resource placement policies. The proposed MLSTM-CEFL is achieved through its capabilities of adjusting virtual resources in the edge aggregator based on congestion prediction, scheduling the long-term forwarding path, and modifying the offloading decision of FL local–edge communications.

The predicted congestion rates $Y_{M_{n(t)}}$ provide insights into IIoT-EFL network performance and enable QoS enhancements. By anticipating potential traffic bottlenecks, QoS in EFL model update processes can be predicted as $\tau_{pre}(t+1)$ to compare with the upper-bound $\tau_m(t+1)$ for dynamically scheduling flow paths and allocating edge resources where needed to prevent congestion. The overall process ensures optimal model update flow, minimizes delays, and enhances learning experience, which harmonizes network resources with demand and maintains reliable service delivery.

4. Experiment and Analysis

4.1. Simulation Environment

In simulation environments, there are three significant phases including MLSTM-based modelling, E2E IIoT networks, and FL communications for imaging services.

For the first phase, the Python programming language is used for MLSTM-based modelling experimentation [43–45]. The proposed algorithms contain specified parameters and algorithmic flow, as described above, in collaboration with the deployed system architecture. The interaction between the prediction module and the orchestration entity enhances real-time and elastic performance. The parameters, including the number of models, batch sizes, dense units, MLSTM units, and the number of testing epochs, are set to 3, 512, 4 times the number of MLSTM units, 128, and 125, respectively. The input data of $X_{M_{n(t)}}$ is gathered through device and resource abstractions using OpenFlow-based techniques, flow monitoring, and RESTful API techniques. The overview procedures of the MLSTM-CEFL simulation are illustrated in Figure 3. The primary stage encompasses inputting the gathered FL model and network states until the evaluation of both training and testing batches is completed.

Algorithm 1 demonstrates the pseudocode for implementing our proposed controller, MLSTM-CP, in TensorFlow and Keras-based functions (e.g., *SpatialDropout1D*, etc.), integrating virtual containers at the edge and network interfaces through SDN/NFV MANO to converge with the final MLSTM-CEFL policy (lines 18–23). Utilizing general features collected, as given in Equation (5), the model selection and splitting processes are executed for training batches $\left(X^{tr}_{M_{n(t)}}, Y^{tr}_{M_{n(t)}}\right)$ and testing batches $\left(X^{te}_{M_{n(t)}}, Y^{te}_{M_{n(t)}}\right)$. The procedure begins by initializing the number of models, batch sizes, dense units, MLSTM units, and the number of epochs, denoted as $N(m)$, $B(s)$, $D(u)$, $MLSTM(u)$, and $N(e)$, respectively. The **modelBuilder** function is executed with a subordinate computing controller, linked to the SDN database, to construct the model's structure. Parameters, hyperparameters, loss, and optimizer functions are configurable to maximize the optimality of **finalModel**. With

multiple models, each model type is specified using the **modelBuilder** function and fitted with different batch sizes and weights. For sustained reliability, an averaging formulation is applied to each output of the **finalModel**.

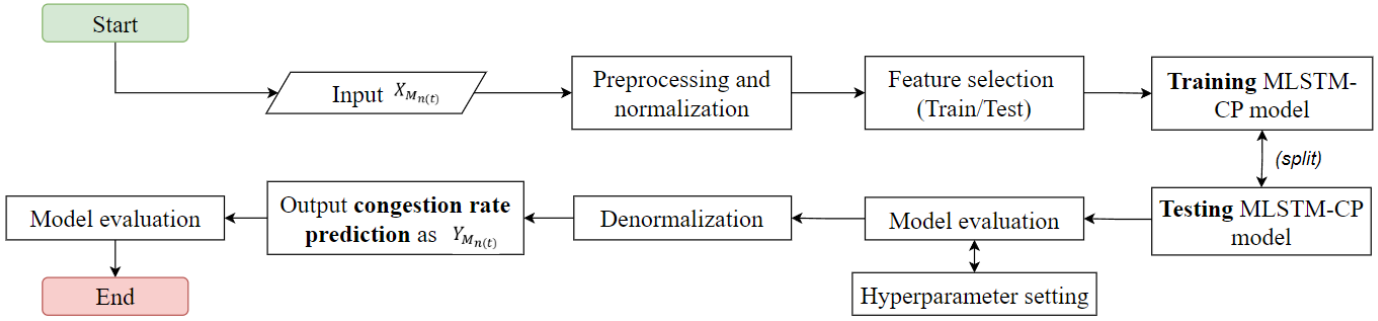

Figure 3. Overview of MLSTM-CEFL procedures for denormalizing the hidden state into congestion rate $Y_{M_{n(t)}}$.

Algorithm 1 Pseudocode of softwarized MLSTM-CP

Requires: $\left\{pt\left[M_{n(t)}\right],\ tl\left[M_{n(t)}\right],\ sc\left[M_{n(t)}\right],\ ts\left[M_{n(t)}\right], bs\left[M_{n(t)}\right], qi\left[M_{n(t)}\right]\right\}$
Ensure: Optimal schedule flow batches
Initialize $N(m)$, $B(s)$, $D(u)$, $MLSTM(u)$, $N(e)$, and $\left(X_{M_{n(t)}}, Y_{M_{n(t)}}\right)$
def **modelBuilder():**
  Embedding and *SpatialDropout1D* of features $\boldsymbol{X_{M_{n(t)}}}$
  X = **Bidirectional** $(CuDNNLSTM\ (MLSTM(u)\ ))\ \left(\boldsymbol{X_{M_{n(t)}}}\right)$
  hiddenLayer = **concatenate**([*GlobalMaxPooling1D*, *GlobalAveragaePooling1D*])
  hiddenLayer = **add** $([hiddenLayer,\ Dense(D(u),\ relu)]$
  ouputLayer = **Dense** $(1,\ sigmoid)$
  finalModel = **Model**(inputs = X, outputs = *ouputLayer*)
  finalModel.compile (*loss*, *optimizer*)
  return **finalModel**
Split $\left(X^{tr}_{M_{n(t)}},\ Y^{tr}_{M_{n(t)}}\right)$ *and* $\left(X^{te}_{M_{n(t)}}, Y^{te}_{M_{n(t)}}\right)$
Sampling weights: $\boldsymbol{W}$
for each model m in range of $N(m)$ ***do***
  finalModel = **modelBuilder**()
  for global iteration e in range of $N(e)$ ***do***
    finalModel.fit $\left(X^{tr}_{M_{n(t)}},\ Y^{tr}_{M_{n(t)}},\ B(s),\ e,\ W\ formulation\right)$
    Executing prediction and flatten output
Average $Y_{M_{n(t)}}$ based on **finalModel** and $\boldsymbol{W}$ $formulation$
Obtain $Y_{M_{n(t)}}$ to EFL-controller and pre-calculate QoS $\tau_{pre}(t+1)$
if QoS $\tau_{pre}(t+1) \leq \tau_m(t+1)$ ***do***
  Obtain optimal schedule flow batches to MLSTM-CEFL
else
  Re-schedule model flow rules and resource descriptors (Mininet, MiniNFV)

Two baseline approaches are employed for comparison: (1) the conventional LSTM-based congestion-aware FL aggregation scheduling and (2) MLSTM for centralized congestion-aware FL aggregation scheduling, denoted as CLSTM-CFL and MLSTM-CCFL respectively. Each method is explained as follows:

- **MLSTM-CCFL** employs the same MLSTM architecture to predict congestion rates within a centralized FL framework. This approach capitalizes on MLSTM's proficiency in capturing temporal dependencies and adapting to dynamic input sequences. However, in this method, data from multiple participants and edge devices, along with their update statuses, are collected and transmitted to a central processing unit

where the MLSTM model predicts congestion. The centralized feature of this approach introduces a trade-off between two factors: (1) the increased latency due to the need to transmit input features to a central server for processing and (2) the benefits of globalized and uniform decision making.

- **CLSTM-CFL** relies on the capacity of traditional LSTM to recognize temporal patterns and relationships in FL input data. CLSTM-CFL can be limited by its conventional LSTM architecture, which may hinder its ability to promptly respond to real-time variations and adapt effectively to evolving state dynamics, such as multi-round participant model updates and edge aggregation. Nevertheless, its advantage lies in resource consumption, as it operates with lightweight execution.

For the second phase, in E2E IIoT networks, the primary processes on SDN/NFV MANO are addressed in the simulation using the configured testbed with TOSCA language [46,47]. The descriptors are programmable, following the outputs of the MLSTM module. In this stage, the final orchestration between the MLSTM-CP outputs and controllers takes the form of the proposed MLSTM-CEFL. The parameter setup is detailed in Table 3, and the system is executed across three consecutive congestion states for 250 s, including non-congestion, normal congestion, and heavy congestion scenarios. Traffic generation occurs for each congestion state evaluation. For each state transition from non-congestion to heavy congestion, queueing algorithms within each path's gateway buffer are observed. Fair-queueing-based pacing is employed in this study. To evaluate the proposed policy, metrics assessing the overall control QoS performance are analyzed by conducting simulations comparing the proposed MLSTM-CEFL with other baseline policies.

Table 3. Primary parameter configuration for our experiment setup.

Purpose/Platform	Specification
Hosting infrastructure	Intel(R) Xeon(R) Silver 4280 CPU @ 2.10 GHz, 128 GB, NVIDIA Quadro RTX 4000 GPU
DL platform	Python (TensorFlow and Keras)
FL platform	TensorFlow Federated (MNIST dataset)
Number of models	3 (CLSTM-CFL, MLSTM-CCFL, and MLSTM-CEFL)
Input shape	(Number of time steps, number of features)
Dropout rate	0.2
Recurrent dropout rate	0.1
Batch sizes	512
Dense units	4 times of MLSTM units
MLSTM units	128
Number of epochs (training, testing)	(1000, 125)
Optimizer	Adam
Loss	Mean Squared Error
Activation function	Sigmoid and Tanh
Learning rate	0.001
Validation split	0.2
Simulation times for capturing QoS	250 s (Fair-queueing-based pacing)
SDN/NFV-UE, control, and interfaces	Mininet+MiniNFV, RESTful API
Number of participants, edge aggregator, and server	(500, 5, 1)

In the final phase, after orchestrating the proposed MLSTM-CEFL, EFL communications are executed for the development of the final learning model using TensorFlow

Federated with data contributions from the MNIST handwritten digits dataset. The adapted configuration of the EFL framework is rooted in the differentiation of drop ratios from the second phase, aimed at constraining the count of participants and model updates during specific congestion state intervals. The performance evaluation of this phase refers to accuracy and loss for comparative analysis.

4.2. QoS Evaluation Metrics

Various organizations, such as the International Telecommunication Union (ITU) [48], develop standards and guidelines for QoS to ensure consistent service quality globally. QoS in EFL is a critical concept that governs the performance and reliability of the learning process within a decentralized network of edge devices. EFL leverages QoS to ensure that the transmission, aggregation, and processing of ML/DL models occur efficiently. By optimizing metrics, such as latency and reliability (e.g., model delivery ratios), QoS ensures that industrial applications and services meet predefined standards, resulting in enhanced user experiences and efficient operations across diverse industries [49].

In our study, we prioritize the minimization of delay (see (4)) and optimization of reliability metrics (FL model drop and delivery ratios). By leveraging MLSTM-CP on predictive capabilities, the system anticipates congestion points and allocates resources proactively. Reduced latency guarantees real-time interactions and timely model updates, thus enhancing the final averaged learning model. Equation (13) presents the total latency T_{TOTAL} of E2E IIoT. The total latency is the sum of radio access delay t_{RAN}, edge aggregation delay t_{EDGE}, and orchestration control delay $t_{CONTROL}$. The edge latency encompasses the transmission time between gateways in the fronthaul networks. The control latency refers to the model updating time between edge aggregators and federated averaging entities in the control plane. The latencies associated with resource adjustment and scheduling, subsequent to MLSTM-CEFL orchestration, are considered by $t_{CONTROL}$.

$$T_{TOTAL} = t_{RAN} + t_{EDGE} + t_{CONTROL} \tag{13}$$

FL model drop ratios indicate the proportion of models that were discarded during the FL round communication process due to congestion or resource limitations. It reflects the effectiveness of the controller as VIM and policy orchestration. The drop ratios highlight the potential challenges faced in maintaining a stable connection between participants, edge aggregators, and the central server. In contrast, FL model delivery ratios quantify the success rate of delivering the model updates in IIoT-EFL systems.

4.3. Results and Discussion

In this sub-section, the results of precision, accuracy/loss of IIoT-EFL-based imaging services, and QoS performances of E2E IIoT networks are given to allow comparison between the proposed MLSTM-CEFL, MLSTM-CCFL, and CLSTM-CFL. The percentage of training and testing accuracies are given for MLSTM-based modules in predicting the congestion rates at every input time slot. Figure 4 illustrates the accuracy and loss value of the proposed and baseline approaches.

Within 125 sampling time slots, the prediction rates from non-congestion to a heavy-congestion level are acceptable, illustrated by an insignificant difference between them. With precise congestion rate prediction, the proposed orchestrator and controller significantly improve the proactive resource placement towards the optimal edge aggregator for particular conditional intervals. In terms of (training, testing), the proposed module achieves (4.1979%, 7.2369%) and (10.6698%, 13.9758%) higher accuracies than MLSTM-CCFL and CLSTM-CFL, respectively. MLSTM-CCFL fails to focus on the distinct edge-specific service that could contribute to more accurate centralized congestion predictions. Accuracy is a metric that is primarily used for classification problems where the goal is to correctly classify data points into different classes. In the context of time series prediction (multi-round communication in EFL) with MLSTM, accuracy acts as a certain case in predicting the exact congestion rate values. In our simulation setup, we convert the

congestion rate predictions into three discrete classes (low, normal, and high congestion) and then calculate accuracy based on those classes. Our approach transforms the regression problem into a classification problem by dividing it into a limited number of classes for prediction. We delineate congestion rate ranges: 0–30% as low congestion, 30–70% as normal congestion, and above 70% as high congestion. Subsequently, we map the predicted congestion rate to one of these classes and calculate accuracy based on how effectively the model categorizes the congestion rates into the correct states.

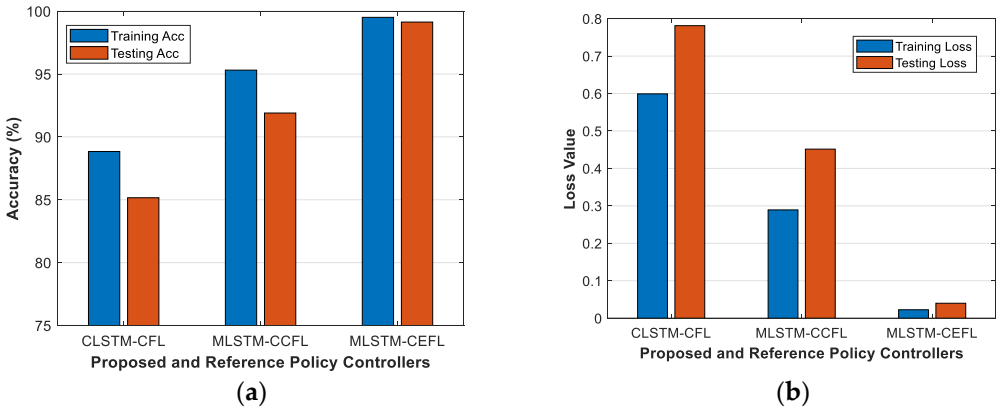

Figure 4. (**a**) Accuracy and (**b**) loss value of proposed and reference policy controllers.

Another primary focus is the loss metric (MSE); the goal is to minimize it to ensure accurate predictions of congestion rates at the aggregator buffers over multi-round communications. The loss represents the discrepancy between the predicted values and the actual target values. In our setup, the loss value measures how well the model is able to predict the congestion rate at the edge for a given time step. MSE measures the average squared difference between the predicted values and the actual target values, and the lower the MSE, the better the model is at fitting the training data. Table 4 presents each approach output in terms of accuracy and loss values. After installing the forwarding path and orchestrating the virtual edge resources by following the congestion thread notifications from the proposed MLSTM-CP module, the FL simulation on the imaging dataset is conducted to represent each policy rule. With appropriate offloading of decision makers, the number of round communications is alleviated and the drop possibility of local distribution is significantly reduced. The training and testing loss values of MLSTM-CEFL reach 0.0225 and 0.0399, respectively.

Table 4. Losses and accuracies of proposed schemes.

	CLSTM-CFL	**MLSTM-CCFL**	**MLSTM-CEFL**
Training accuracy	88.8403%	95.3122%	99.5101%
Training loss	0.5991	0.2892	0.0225
Testing accuracy	85.1598%	91.8991%	99.1356%
Testing loss	0.7812	0.4515	0.0399

Figure 5 presents the primary QoS metrics captured for evaluation in this simulation setup. The accuracy and reliability of predicting the congestion rate at an edge aggregator outputs good results using MLSTM-CEFL. The predicted rate and actual experiment rate configuration are acceptable; the policy created a model that can effectively forecast the congestion rate at the edge to help in managing network resources efficiently. A higher similarity in Figure 5a illustrates a predicting congestion rate that allows FL central policy administrators to (1) proactively allocate resources, (2) balance loads, and (3) prevent potential network congestion. The MLSTM-CEFL framework, collaboratively trained across multiple edge aggregators while preserving data privacy and featuring accurate congestion

rate predictions, results in a more effective converged final model. Delays T_{TOTAL} are captured as Equation (13) and illustrated in Figure 5b, serving as a major evaluation metric to determine the performance stability of the proposed MLSTM-CEFL, MLSTM-CCFL, and CLSTM-CFL models. In our architecture, t_{RAN}, t_{EDGE}, and $t_{CONTROL}$ are jointly considered in the Mininet platform. The average delays of the proposed policy reach 30.81 ms among all service settings, which is a 46.8961 ms and 111.2661 ms improvement (lower delays) on the MLSTM-CCFL and CLSTM-CFL policies, respectively. Through the convergence of MLSTM-CP modules with system orchestrators, resources are effectively allocated in each iteration of the local-aggregator updates. Our proposed scheme ensures the system's ability to reliably serve each IIoT application class, with particular emphasis on the mission-critical service class. Accurate congestion rate prediction facilitates the well-adjusted update scheduling of local model parameters, which is particularly applicable in asynchronous FL. Given stable delay variation, experience-based forwarding is efficient for both reactive and proactive configurations. The proposed approach contributes to lower and more stable latencies, enabling the final learning model to achieve latency-efficient characteristics while preventing local models from exceeding the upper-bound maximum delays.

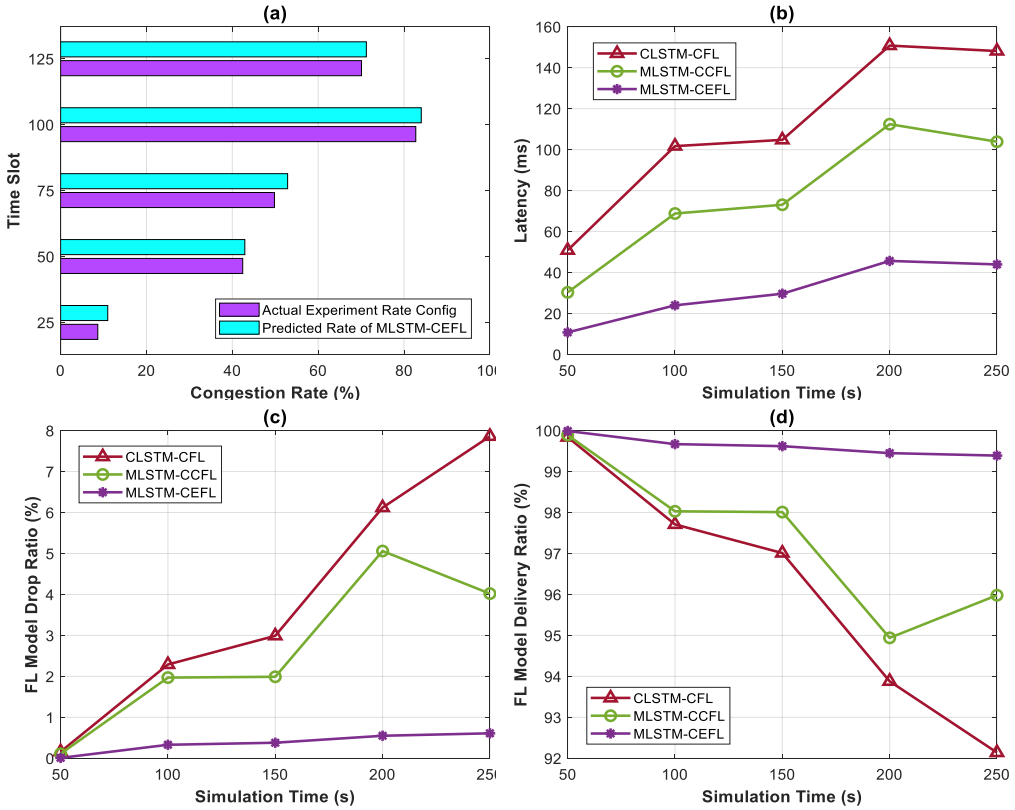

Figure 5. (**a**) Congestion rate prediction in each time slot, (**b**) latency, (**c**) FL model drop ratios, and (**d**) FL model delivery ratios between the proposed and reference schemes.

The FL model drop ratio is presented in Figure 5c. This measure indicates the proportion of local/edge models that are dropped or discarded during the FL process due to various reasons such as (1) communication errors, (2) failures, (3) model convergence issues, or (4) insufficiency of policy orchestration. High model drop ratios can be problematic, as this indicates inefficient communication and collaboration between local devices and the aggregator. High ratios lead to delays in model updates, suboptimal performance, and increased training costs. To maintain the final model performance, our proposed scheme achieved the minimization of model drop ratios to an average of 0.376% among various congestion states in our 250 s simulation. This output provides three primary benefits: (1) enhancing communication reliability, (2) resolving convergence issues, and (3) handling

aggregation scheduling failures. In contrast to the drop ratio, the FL model delivery ratio is presented in Figure 5d. This parameter measures the successful model updates. A high model delivery ratio is significant for ensuring that the latest model updates from edge devices are successfully integrated into the central model. The proposed approach reached a delivery ratio of 99.9922%, which is applicable for IIoT application services following the criteria standardization. The proposed approach has delivery ratios 2.2541% and 3.5081% higher than MLSTM-CCFL and CLSTM-CFL, respectively. By identifying future congestion intervals, a proactive model aggregation update orchestrates the forwarding path to optimize matching local–aggregator throughputs and minimize total latency, as described in Equation (4). The proposed policy achieves a high delivery ratio that aims to deliver great precision performances in each IIoT service model.

The non-monotonic changes observed in the results for the proposed MLSTM-CEFL are attributed to the inherent dynamics of network congestion. Stability is achieved by accommodating variations in congestion states throughout the 250 s simulation setup. The performance metrics—latency, drop ratio, and delivery ratio—are influenced by real-time fluctuations in traffic loads, network conditions, and resource allocations. However, the capability of the proposed MLSTM-CEFL to adapt and optimize within these changing conditions ensures a stable response to varying congestion levels, enhancing the overall QoS and resource utilization.

5. Conclusions and Future Works

This paper introduces an MLSTM-based congestion-aware EFL, namely MLSTM-CEFL, as a prediction handler and controller to orchestrate the aggregation scheduling policy. By leveraging the MLSTM-CEFL model, congestion prediction activates the capability for the controller to proactively measure the next-state QoS metrics and allocates the resource on edge aggregators efficiently for multi-service model averaging. Our network architecture is divided into participants, edge aggregators, and controllers, which enables scalable IIoT model aggregation. Integration of MLSTM-CP and the controller as a VIM enhances adaptability and facilitates policy decision making to obtain the optimal scheduling flows. The simulation results showcased the precision and stability of MLSTM-CEFL throughout various congestion network states in order to ensure multi-class QoS expectations and efficient resource placement. Our proposed approach minimized the E2E delays and optimized the reliability in terms of model drop/delivery ratios.

In future studies, the E2E network slicing based on the EFL framework for IIoT applications will be extended. Another perspective involving multi-convolutional neural networks for various industrial sensing image datasets will be explored to capture the differentiation of computing latencies. The slicing framework aims to enhance the multi-class handling of time-sensitive model communications in EFL-based real-time image processing applications.

Author Contributions: Conceptualization, P.T. and S.K. (Seokhoon Kim); methodology, P.T.; software, P.T. and S.K. (Seungwoo Kang); validation, S.K. (Seungwoo Kang), and S.R.; formal analysis, S.K. (Seungwoo Kang), and S.R.; investigation, S.K. (Seokhoon Kim); resources, S.K. (Seokhoon Kim); data curation, P.T.; writing—original draft preparation, P.T.; writing—review and editing, S.K. (Seungwoo Kang) and S.R.; visualization, S.K. (Seungwoo Kang); supervision, S.K. (Seokhoon Kim); project administration, S.K. (Seokhoon Kim); funding acquisition, S.K. (Seokhoon Kim). All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by an Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. RS-2022-00167197, Development of Intelligent 5G/6G Infrastructure Technology for The Smart City), in part by the National Research Foundation of Korea (NRF), Ministry of Education, through the Basic Science Research Program under Grant NRF-2020R1I1A3066543, in part by BK21 FOUR (Fostering Outstanding Universities for Research) under Grant 5199990914048, and in part by the Soonchunhyang University Research Fund.

Data Availability Statement: Not applicable.

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

References

1. Khalil, R.A.; Saeed, N.; Masood, M.; Fard, Y.M.; Alouini, M.-S.; Al-Naffouri, T.Y. Deep Learning in the Industrial Internet of Things: Potentials, Challenges, and Emerging Applications. *IEEE Internet Things J.* **2021**, *8*, 11016–11040. [CrossRef]
2. Khan, A.I.; Al-Badi, A. Open Source Machine Learning Frameworks for Industrial Internet of Things. *Procedia Comput. Sci.* **2020**, *170*, 571–577. [CrossRef]
3. Lin, Z.; Lin, M.; Champagne, B.; Zhu, W.-P.; Al-Dhahir, N. Secrecy-Energy Efficient Hybrid Beamforming for Satellite-Terrestrial Integrated Networks. *IEEE Trans. Commu.* **2021**, *69*, 6345–6360. [CrossRef]
4. Lin, Z.; An, K.; Niu, H.; Hu, Y.; Chatzinotas, S.; Zheng, G.; Wang, J. SLNR-Based Secure Energy Efficient Beamforming in Multibeam Satellite Systems. *IEEE Trans. Aerosp. Electron. Syst.* **2023**, *59*, 2085–2088. [CrossRef]
5. Sun, Y.; An, K.; Zhu, Y.; Zheng, G.; Wong, K.-K.; Chatzinotas, S.; Ng, D.W.K.; Guan, D. Energy-Efficient Hybrid Beamforming for Multilayer RIS-Assisted Secure Integrated Terrestrial-Aerial Networks. *IEEE Trans. Commun.* **2022**, *70*, 4189–4210. [CrossRef]
6. Tam, P.; Math, S.; Kim, S. Efficient Resource Slicing Scheme for Optimizing Federated Learning Communications in Software-Defined IoT Networks. *J. Internet Serv. Appl.* **2021**, *22*, 27–33.
7. European Telecommunications Standards Institute (ETSI). Deployment of Mobile Edge Computing in an NFV Environment. *ETSI Group Rep. MEC* **2018**, *17*, V1.
8. Qu, K.; Zhuang, W.; Ye, Q.; Shen, X.; Li, X.; Rao, J. Dynamic Flow Migration for Embedded Services in SDN/NFV-Enabled 5G Core Networks. *IEEE Trans. Commun.* **2020**, *68*, 2394–2408. [CrossRef]
9. Mahmood, A.; Beltramelli, L.; Fakhrul Abedin, S.; Zeb, S.; Mowla, N.; Hassan, S.A.; Sisinni, E.; Gidlund, M. Industrial IoT in 5G-And-beyond Networks: Vision, Architecture, and Design Trends. *IEEE Trans. Ind. Inform.* **2021**, *18*, 4122–4137. [CrossRef]
10. Yang, H.; Alphones, A.; Zhong, W.-D.; Chen, C.; Xie, X. Learning-Based Energy-Efficient Resource Management by Heterogeneous RF/VLC for Ultra-Reliable Low-Latency Industrial IoT Networks. *IEEE Trans. Ind. Inform.* **2020**, *16*, 5565–5576. [CrossRef]
11. Jin, H.; Jia, L.; Zhou, Z. Boosting Edge Intelligence with Collaborative Cross-Edge Analytics. *IEEE Internet Things J.* **2021**, *8*, 2444–2458. [CrossRef]
12. Li, E.; Zeng, L.; Zhou, Z.; Chen, X. Edge AI: On-Demand Accelerating Deep Neural Network Inference via Edge Computing. *IEEE Trans. Wirel. Commun.* **2019**, *19*, 447–457. [CrossRef]
13. Lin, Z.; Bi, S.; Zhang, Y.-J.A. Optimizing AI Service Placement and Resource Allocation in Mobile Edge Intelligence Systems. *IEEE Trans. Wirel. Commun.* **2021**, *20*, 7257–7271. [CrossRef]
14. Brendan, M.H.; Moore, E.; Ramage, D.; Hampson, S.; Arcas, B.A. Communication-Efficient Learning of Deep Networks from Decentralized Data. *arXiv* **2016**. Available online: https://arXiv.org/abs/1602.05629 (accessed on 10 June 2023).
15. Park, J.; Samarakoon, S.; Elgabli, A.; Kim, J.; Bennis, M.; Kim, S.-L.; Debbah, M. Communication-Efficient and Distributed Learning over Wireless Networks: Principles and Applications. *Proc. IEEE* **2021**, *109*, 796–819. [CrossRef]
16. Hu, C.-H.; Chen, Z.; Larsson, E.G. Device Scheduling and Update Aggregation Policies for Asynchronous Federated Learning. *arXiv* **2021**. Available online: https://arXiv.org/abs/2107.11415 (accessed on 15 June 2023).
17. Khan, L.U.; Saad, W.; Han, Z.; Hossain, E.; Hong, C.S. Federated Learning for Internet of Things: Recent Advances, Taxonomy, and Open Challenges. *arXiv* **2021**. [CrossRef]
18. Raj, J.T. Building Decentralized Image Classifiers with Federated Learning. In Proceedings of the 2020 IEEE Region 10 Symposium (TENSYMP), Dhaka, Bangladesh, 5–7 June 2020; pp. 489–494.
19. Ren, Y.; Guo, A.; Song, C. Multi-Slice Joint Task Offloading and Resource Allocation Scheme for Massive MIMO Enabled Network. *KSII Trans. Internet Inf. Syst.* **2023**, *17*, 794–815.
20. Zhu, Y.; Liu, C.; Zhang, Y.; You, W. Research on 5G Core Network Trust Model Based on NF Interaction Behavior. *KSII Trans. Internet Inf. Syst.* **2022**, *16*, 3333–3354.
21. Hu, Y.; Zhu, L.; Zhang, J.; Cai, Z.; Han, J. Migration and Energy Aware Network Traffic Prediction Method Based on LSTM in NFV Environment. *KSII Trans. Internet Inf. Syst.* **2023**, *17*, 896–915.
22. Pei, J.; Hong, P.; Xue, K.; Li, D.; Wei, D.S.L.; Wu, F. Two-Phase Virtual Network Function Selection and Chaining Algorithm Based on Deep Learning in SDN/NFV-Enabled Networks. *IEEE J. Sel. Areas Commun.* **2020**, *38*, 1102–1117. [CrossRef]
23. Xie, J.; Fang, J.; Liu, C.; Li, X. Deep Learning-Based Spectrum Sensing in Cognitive Radio: A CNN-LSTM Approach. *IEEE Commun. Lett.* **2020**, *24*, 2196–2200. [CrossRef]
24. Tajiri, K.; Kawahara, R.; Matsuo, Y. Optimizing Edge-Cloud Cooperation for Machine Learning Accuracy Considering Transmission Latency and Bandwidth Congestion. In Proceedings of the NOMS 2022—IEEE/IFIP Network Operations and Management Symposium, Budapest, Hungary, 25–29 April 2022; pp. 1–9.
25. Zhang, F.; Li, X.; Fan, G. Optimized LSTM Network Based on Particle Swarm Algorithm for Power Time Series Data Prediction. In Proceedings of the 13th International Conference on Intelligent Computation Technology and Automation (ICICTA), Xi'an, China, 24–25 October 2020; pp. 394–398.
26. Bi, J.; Zhang, X.; Yuan, H.; Zhang, J.; Zhou, M. A Hybrid Prediction Method for Realistic Network Traffic with Temporal Convolutional Network and LSTM. *IEEE Trans. Autom. Sci. Eng.* **2022**, *19*, 1869–1879. [CrossRef]

27. Wan, X.; Liu, H.; Xu, H.; Zhang, X. Network Traffic Prediction Based on LSTM and Transfer Learning. *IEEE Access.* **2022**, *10*, 86181–86190. [CrossRef]
28. Na, H.; Shin, Y.; Lee, D.; Lee, J. LSTM-Based Throughput Prediction for LTE Networks. *ICT Express* **2021**, *9*, 247–252. [CrossRef]
29. Nihale, S.; Sharma, S.; Parashar, L.; Singh, U. Network Traffic Prediction Using Long Short-Term Memory. In Proceedings of the 2020 International Conference on Electronics and Sustainable Communication Systems (ICESC), Coimbatore, India, 2–4 July 2020; pp. 338–343.
30. Trinh, H.D.; Giupponi, L.; Dini, P. Mobile Traffic Prediction from Raw Data Using LSTM Networks. In Proceedings of the 2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), Bologna, Italy, 9–12 September 2018; pp. 1827–1832.
31. Chen, L.; Zhang, X.; Sun, L. Network Flow Delay Prediction Model Based on LSTM. In Proceedings of the 2021 International Conference on Electronic Information Engineering and Computer Science (EIECS), Changchun, China, 23–25 September 2021; pp. 395–399.
32. Tam, P.; Math, S.; Lee, A.; Kim, S. Multi-Agent Deep Q-Networks for Efficient Edge Federated Learning Communications in Software-Defined IoT. *Comput. Mater. Contin.* **2022**, *71*, 3319–3335. [CrossRef]
33. Bonawitz, K.; Eichner, H.; Grieskamp, W.; Huba, D.; Ingerman, A.; Ivanov, V.; Kiddon, C.; Konečný, J.; Mazzocchi, S.; Brendan, M.H.; et al. Towards Federated Learning at Scale: System Design. *arXiv* **2019**. Available online: https://arXiv.org/abs/1902.01046 (accessed on 30 June 2023).
34. Nguyen, D.C.; Ding, M.; Pathirana, P.N.; Seneviratne, A.; Li, J.; Niyato, D.; Poor, H.V. Federated Learning for Industrial Internet of Things in Future Industries. *IEEE Wirel. Commun.* **2021**, *28*, 192–199. [CrossRef]
35. Qiu, W.; Ai, W.; Chen, H.; Feng, Q.; Tang, G. Decentralized Federated Learning for Industrial IoT with Deep Echo State Networks. *IEEE Trans. Ind. Inform.* **2023**, *19*, 5849–5857. [CrossRef]
36. Hu, X.; Zhao, Y.; Huang, Y.; Zhu, C.; Yao, J.; Fang, N. Hierarchical Resource Management Framework and Multi-hop Task Scheduling Decision for Resource-Constrained VEC Networks. *KSII Trans. Internet Inf. Syst.* **2022**, *16*, 3638–3657.
37. Guo, W.; Zhao, M.; Cui, Z.; Xie, L. A Bi-objective Game-based Task Scheduling Method in Cloud Computing Environment. *KSII Trans. Internet Inf. Syst.* **2022**, *16*, 3565–3583.
38. Kong, Q.; Yin, F.; Lu, R.; Li, B.; Wang, X.; Cui, S.; Zhang, P. Privacy-Preserving Aggregation for Federated Learning-Based Navigation in Vehicular Fog. *IEEE Trans. Ind. Inform.* **2021**, *17*, 8453–8463. [CrossRef]
39. Zhang, P.; Wang, C.; Jiang, C.; Han, Z. Deep Reinforcement Learning Assisted Federated Learning Algorithm for Data Management of IIoT. *IEEE Trans. Ind. Inform.* **2021**, *17*, 8475–8484. [CrossRef]
40. He, Y.; Yang, M.; He, Z.; Guizani, M. Computation Offloading and Resource Allocation Based on DT-MEC-Assisted Federated Learning Framework. *IEEE Trans. Cogn. Commun. Netw.* **2023**. [CrossRef]
41. Chakraborty, A.; Misra, S. QoS-Aware Resource Bargaining for Federated Learning over Edge Networks in Industrial IoT. *IEEE Trans. Netw. Sci. Eng.* **2022**. [CrossRef]
42. Abreha, H.G.; Hayajneh, M.; Serhani, M.A. Federated Learning in Edge Computing: A Systematic Survey. *Sensors* **2022**, *22*, 450. [CrossRef] [PubMed]
43. Zhuo, Q.; Li, Q.; Yan, H.; Qi, Y. Long Short-Term Memory Neural Network for Network Traffic Prediction. In Proceedings of the 12th International Conference on Intelligent Systems and Knowledge Engineering (ISKE), Nanjing, China, 24–26 November 2017; pp. 1–6. [CrossRef]
44. Mogyorosi, F.; Pasic, A.; Cziva, R.; Revisnyei, P.; Kenesi, Z.; Tapolcai, J. Adaptive Protection of Scientific Backbone Networks Using Machine Learning. *IEEE Trans. Netw. Serv. Manag.* **2021**, *18*, 1064–1076. [CrossRef]
45. Zhuo, Y.; Li, B. Fedns: Improving Federated Learning for Collaborative Image Classification on Mobile Clients. In Proceeding of the 2021 IEEE International Conference on Multimedia and Expo (ICME), Shenzhen, China, 5–9 July 2021.
46. Tam, P.; Math, S.; Nam, C.; Kim, S. Adaptive Resource Optimized Edge Federated Learning in Real-Time Image Sensing Classifications. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2021**, *14*, 10929–10940. [CrossRef]
47. Tam, P.; Math, S.; Kim, S. Optimized Multi-Service Tasks Offloading for Federated Learning in Edge Virtualization. *IEEE Trans. Netw. Sci. Eng.* **2022**, *9*, 4363–4378. [CrossRef]
48. National Communications Authority. *Guidelines on Regulatory Aspects of QoS*; ITU-T E.800-series; National Communications Authority: Accra, Ghana, 2021.
49. Poryazov, S.A.; Saranova, E.T.; Andonov, V.S. Overall Model Normalization towards Adequate Prediction and Presentation of QoE in Overall Telecommunication Systems. In Proceeding of the 2019 14th International Conference on Advanced Technologies, Systems and Services in Telecommunications (TELSIKS), Nis, Serbia, 23–25 October 2019; pp. 360–363.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.