

Article

TransNeural: An Enhanced-Transformer-Based Performance Pre-Validation Model for Split Learning Tasks

Guangyi Liu ¹, Mancong Kang ^{2,*}, Yanhong Zhu ^{1,3,*}, Qingbi Zheng ^{1,4}, Maosheng Zhu ⁵ and Na Li ^{1,4}¹ China Mobile Research Institute, Beijing 100053, China; liuguangyi@chinamobile.com (G.L.); zhengqingbi@chinamobile.com (Q.Z.); linawx@chinamobile.com (N.L.)² School of Communications and Information Engineering, Beijing University of Posts and Telecommunications, Beijing 100876, China³ School of Electronics and Information Engineering, Beijing Jiaotong University, Beijing 100091, China⁴ ZGC Institute of Ubiquitous-X Innovation and Application, Beijing 100191, China⁵ China Mobile (Suzhou) Software Technology Co., Ltd., Suzhou 215163, China; zhumaosheng@cmss.chinamobile.com

* Correspondence: kmc@bupt.edu.cn (M.K.); zhuyanhongyjy@chinamobile.com (Y.Z.)

Abstract: While digital twin networks (DTNs) can potentially estimate network strategy performance in pre-validation environments, they are still in their infancy for split learning (SL) tasks, facing challenges like unknown non-i.i.d. data distributions, inaccurate channel states, and misreported resource availability across devices. To address these challenges, this paper proposes a TransNeural algorithm for DTN pre-validation environment to estimate SL latency and convergence. First, the TransNeural algorithm integrates transformers to efficiently model data similarities between different devices, considering different data distributions and device participate sequence greatly influence SL training convergence. Second, it leverages neural network to automatically establish the complex relationships between SL latency and convergence with data distributions, wireless and computing resources, dataset sizes, and training iterations. Deviations in user reports are also accounted for in the estimation process. Simulations show that the TransNeural algorithm improves latency estimation accuracy by 9.3% and convergence estimation accuracy by 22.4% compared to traditional equation-based methods.



Citation: Liu, G.; Kang, M.; Zhu, Y.; Zheng, Q.; Zhu, M.; Li, N. TransNeural: An Enhanced-Transformer-Based Performance Pre-Validation Model for Split Learning Tasks. *Sensors* **2024**, *24*, 5148. <https://doi.org/10.3390/s24165148>

Academic Editor: Hazer Inaltekin

Received: 21 June 2024

Revised: 25 July 2024

Accepted: 6 August 2024

Published: 9 August 2024



Copyright: © 2024 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/>).

Keywords: 6G; digital twin network; transformer; split learning; pre-validation environment

1. Introduction

In the era of the sixth generation mobile network (6G) [1,2], an increasing number of artificial intelligence (AI) applications will rely on edge network to train their continually expanding models based on massive amounts of user data [3]. While users may be willing to provide their data with payment, they are concerned about data privacy issues. Meanwhile, fully training AI models on user devices to upload only model parameters can address privacy issues, but is often impractical due to constraint user resources [4]. Additionally, companies are reluctant to fully disclose their model parameters, since these are valuable assets [5]. In this context, split learning (SL) has emerged as a promising solution [6]. In “SL for model training process”, an edge server (ES) connected with an access point (AP) offloads only a few lower layers of AI models to a user device for local training, where the device interacts intermediate results in forward and backward propagations with ES to update the upper layers. Once a user device finishes updating using its own data, it sends the updated lower layers to the next user device via AP, continuing this process until all participants have contributed their data. This model training method protects user privacy and conserves device resources, and we abbreviate it as “SL tasks”. The applications include illness model training in healthcare [7] and user behavior analysis in mobile networks [8], etc.

With the adoption of SL technology, a crucial concern for AI application companies is to pre-estimate SL latency and convergence performance under given resource allocation strategy before paying users and operators for data, wireless, and computing resources [6]. However, under unknown non-i.i.d. data distribution [9], inaccurate wireless channel state information (CSI), and misreported available computing resources on different user devices, it becomes extremely difficult to accurately estimate the SL performance before real physical executions. The reason is as follows. Suppose there are three participate user devices: A, B, and C. If the data distribution on A is similar to that on C, but significantly different from B, then the training sequences A-C-B and A-B-C will result in considerable differences in SL training convergence, despite utilizing the same amount of resources. The situation becomes even more problematic when there exists reporting deviations in CSI, amount of data and computing resources on devices. Consequently, AI application companies face high risks when investing heavily in network operators and user devices for SL training on edge, without confidence in achieving their training latency and convergence goals, which significantly dampens their enthusiasm.

The Digital Twin Network (DTN), emerging as a critical 6G technology [10], is an intelligent digital replica of a physical network that synchronizes physical states in real-time and uses AI models to analyze the characteristics and relationships of network components for accurate inferences in 6G network [11]. In the context of SL, DTN is expected to automatically model data similarities between devices, account for user misreporting and correct inaccurate CSI based on the unique profiles of different devices. Moreover, it can learn the complex relationships between service performance and network parameters. These functions can be realized based on AI models and form a pre-validation environment in DTN for testing SL strategy performance. Based on the pre-validation results, DTN can repeatedly optimize the SL strategy until it meets latency and convergence requirements before deployment in the physical world.

Nevertheless, several challenges exist in establishing DTN pre-validation environment for SL training tasks. First, learning data similarities between different physical devices is difficult due to data privacy issues, which prevent the acquisition and analysis of user data. Second, accurately estimating the misreported behaviors of each physical device is challenging. These deviations depend on multiple factors such as device version, user behavior, movement patterns, etc. However, the actual physical information is unattainable due to user privacy, data collection errors, and other factors, making it impossible to learn errors from historical data. Third, it is difficult to model the complex relationships between various SL training performance metrics and network factors like data similarities, dataset size, training iterations, device resources, wireless channel quality, and edge server resources, particularly when unknown parameters exist. Fourth, the model needs to be broadly applicable to various SL requirements (such as latency, convergence, and energy efficiency) and different tasks. This would enable an autonomous modeling process for pre-validation environments, advancing toward the ultimate goal of realizing fully automatic network management in DTNs [12].

Existing AI models for the DTN pre-validation environment can be divided into three categories. First, long short-term memory (LSTM) networks are used to establish DTN models for predicting future states. For example, the authors of [13] used LSTM to build a network traffic prediction model for DTN based on historical traffic data from a real physical network, which can be utilized for network optimization. Similarly, the authors of [14] employed LSTM to model and predict changes in reference signal receiving power, aiming to reduce handover failures in DTN. Second, graph neural network (GNNs) are used to model network topology for strategy pre-validation. For instance, the authors of [15] designed a GNN-based pre-validation environment for DTN to estimate delay and jitter under given queuing and routing strategies. Likewise, the authors of [16] proposed a GNN-based pre-validation model to estimate end-to-end latencies for different network slicing schemes. Third, a combination of GNN and transformer models is used to capture both temporal and spatial relationships among network components. For example, the authors

of [17] developed a DTN pre-validation environment that predicts end-to-end quality of service for resource allocation strategies in cloud-native micro-service architectures.

Although existing methods are important explorations for building an intelligent DTN pre-validation environment, several problems persist in the context of SL training tasks. First, LSTM and GNN-based models are only adept at learning the relationships between adjacent nodes, where the learning performance drops quickly as the device separation grows. This property makes them suitable for scenarios where only the relationships between adjacent nodes are critical, such as time-series predictions [18]. However, quickly learning the data relationships between all devices is crucial for accurate SL performance estimation, since every device has the potential to participate with a random sequence. Second, while transformer-based models excel in learning inter-node relationships, they may struggle with understanding the relationships within the feature vector of a single node. This capability is vital for accurately modeling and predicting the performance of SL training tasks in wireless network.

Based on these considerations, this paper presents a TransNeural algorithm for DTN pre-validating environment towards SL training tasks. This algorithm integrates transformers with neural network, leveraging the transformer's strength in inter-node learning [19] to capture data similarities among different user devices and the neural network's strength in inter-feature learning [20] to understand the complex relationships between various system variables, SL training performances, and deviation characteristics between different nodes. The contributions of this work can be summarized as follows:

- A mathematical model for SL training latency and convergence is established, which jointly considers unknown non-i.i.d. data distributions, device participate sequence, inaccurate CSI, and deviations in occupied computing resources. These are crucial factors for SL training performance, but are often overlooked in existing SL studies.
- To close the gap in SL-target DTN pre-validation environment, we propose a TransNeural algorithm to estimate SL training latency and convergence under given resource allocation strategies. This algorithm combines the transformer and neural network to model data similarities between devices, establish complex relationships between SL performance and network factors such as data distributions, wireless and computing resources, dataset sizes, and training iterations, and learn the reporting deviation characteristics of different devices.
- Simulations show that the proposed TransNeural algorithm improves latency estimation accuracy by 13.44% compared to traditional equation-based algorithms and enhances convergence estimation accuracy by 116.74%.

The remainder of our work is organized as follows. Section 2 gives the system model. Section 3 designs the TransNeural algorithm in DTN for estimating the latency and convergence of SL training tasks. Simulation results and discussions are given in Section 4. Finally, we conclude this paper in Section 5.

2. System Description

In this section, we sequentially present the system model, communication model, SL process and DTN.

2.1. System Model

Figure 1 gives the whole system of DTN-integrated network for SL training tasks. In our system, there are N user devices access to an AP equipped with an edge server (ES). Each device is characterized by some unique features, including data distributions, the amount of available computing resources, misreporting behaviors, etc. On the n -th device, the dataset is denoted as $\mathbb{Q}_n$ with total size $Q_n^{\text{sum}} = ||\mathbb{Q}_n||$; the amount of available computing resource is denoted as f_n . The data similarities between different devices depend on their users habits (such as shopping habits) and behaviors. The distance between the n -th device with AP is denoted as d . The index of key notations are listed in Table 1.

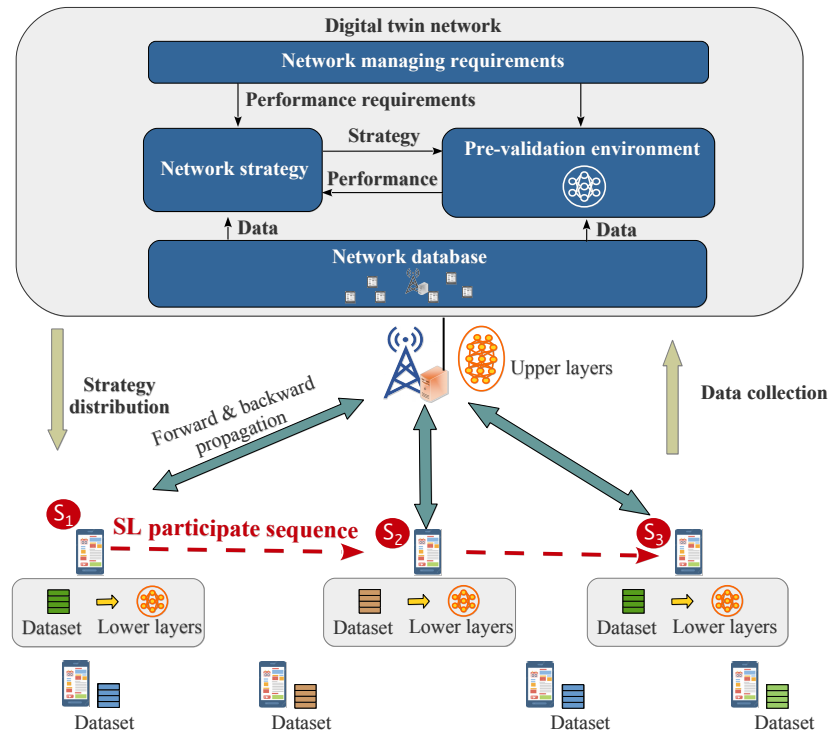


Figure 1. DTN-integrated network for SL training tasks.

There is a DTN deployed on the ES, denoted as $\mathcal{D}$. It synchronizes network states via AP, and generates network strategies (such as resource allocation strategies) according to requirements of oncoming tasks (such as SL training tasks). Specifically, it has a pre-validation environment for testing and optimizing strategies before they are executed in the real physical network, forming a closed-loop strategy optimization in DTN. The contribution of our work is concentrated on the establishment of the pre-validation environment for SL training tasks.

In SL training process, a target AI model is partitioned into “upper layers” and “lower layers”. Under a given resource allocation strategy, the AP would sequentially offload the lower layers to the participate devices, which are responsible for computing forward and backward propagation for lower layers. The updated lower layers would be transmitted to the next participate device via AP until all participants have contributed and the final updated lower layers will be sent back to ES to merge with upper layer. This process can train the target AI model while protecting user data privacy and conserving device resources.

Table 1. Index of key notations.

Notation	Description
Q_n, Q_n^{sum}	Dataset on the n -th device, size of dataset on the n -th device
$\mathcal{D}, \mathcal{O}, \mathcal{G}, \mathcal{E}, \mathcal{Z}$	DTN notation, network management module, network strategy module, pre-validation environment and network database in DTN
$f_{av,n}, f_{av,n}^*, f_n, f_n^*$	Reported and real available computing resources, expected and real computing resources allocating strategy
$R_{n,dn}, R_{n,up}; R_{n,dn}^*, R_{n,up}^*$	Expected transmission rate in downlink and uplink; Real transmission rate in downlink and uplink
$\varepsilon_{s_k, \text{tr}}^*, \varepsilon_{s_k, \text{cp}}^Y, \varepsilon_{s_k}^Y$	Outage probability, computing divergence, final divergence on the s_k -th device
C^Y, I_{s_k}	Data correlation matrix, training iteration
$b_{lw}, b_{up}, b_{mid, fw}, b_{mid, bk}$	Bit-size of lower and upper layers in target model, bit-size of middle parameter in forward and backward propagation

2.2. Communication Model

The AP is equipped with M antenna, and each user device is equipped with one antenna. The transmission rate of each user device is

$$R_{n,i} = B_n \cdot \log_2 \left(1 + \frac{P_{n,i,t} \cdot H_{n,t}}{N_0 B_n} \right), \quad (1)$$

where B_n is the allocated bandwidth for the n -th device, $i = \{\text{up}, \text{dn}\}$ represent uplink and downlink, $P_{n,i,t}$ is the instant uplink or downlink transmitting powers between the n -th device and AP in the t -th time slot, N_0 is the power spectral density of noise, $H_{n,t} = L_n \cdot h_{n,t}$ is the instant channel gain, where L_n is large-scale fading and $h_{n,t}$ is small-scale fading. The channel gain $H_{n,t}$ is a circularly symmetric complex Gaussian with zero mean and variance σ_n^2 [21], which randomly changes across different time slots. The probability density functions (PDF) of $H_{n,t}$ in uplink and downlink are given in Theorem 1.

Theorem 1 (PDF for SIMO and MISO channels). *The PDF of channel gain for SIMO and MISO can be derived as $u(x) = \frac{1}{\sigma_n^2 \Gamma(M)} (x/\sigma_n^2)^{M-1} e^{-x/\sigma_n^2}$, according to [22,23].*

To maximize transmission rate and cancel the influence from small-scale channel gain changing, we adopt a small-scale power adaptation scheme from [24] on each time slot. That is, the uplink allocated transmitting power P_n of device n and the downlink transmitting power P_{AP} of AP are taken as the uplink and downlink average power between different time slots. The scheme adapts the instantaneous power according to the channel gain in the t -th time slot by

$$P_{n,i,t} = \begin{cases} \frac{P_{n,i,\text{tar}}}{H_{n,t}}, & \text{if } H_{n,t} \geq H_L \\ 0, & \text{else,} \end{cases} \quad (2)$$

where H_L is the lower bound of channel gain, $P_{n,i,\text{tar}}$ is target receiving power. Take the uplink transmission process of the n -th device as an example, the relationship between the target receiving power with the average transmitting power P_n is

$$\begin{aligned} P_n &= \mathbb{E}[P_{n,\text{up},t}] = P_{n,\text{up},\text{tar}} \cdot \int_{H_L}^{\infty} \frac{(x/\sigma_n^2)^{M-1} e^{-x/\sigma_n^2}}{\sigma_n^2 \Gamma(M) \cdot x} dx \\ &= \frac{P_{n,\text{up},\text{tar}}}{\sigma_n^2} \cdot \frac{\Gamma^u(M-1, H_L/\sigma_n^2)}{\Gamma(M)}, \end{aligned} \quad (3)$$

where $\Gamma^u(\cdot)$ and $\Gamma^D(\cdot)$ are the upper and lower incomplete Gamma functions, respectively. Therefore, the target receiving power at AP and the n -th device can be derived as

$$P_{n,\text{up},\text{tar}} = P_n \cdot \frac{\Gamma^u(M-1, H_L/\sigma_n^2)}{\sigma_n^2 \cdot \Gamma(M)}, \quad P_{n,\text{dn},\text{tar}} = P_{AP} \cdot \frac{\Gamma^u(M-1, H_L/\sigma_n^2)}{\sigma_n^2 \cdot \Gamma(M)}. \quad (4)$$

By introducing the small-scale power adaptation scheme, the maximized transmission rate is converted from Equation (1) to

$$R_{n,i} = B_n \log_2 \left(1 + \frac{P_{n,i,\text{tar}}}{N_0 B_n} \right). \quad (5)$$

2.3. Split Learning Process

The SL process trains its target AI model based on a given resource allocation scheme, which chooses a set of user devices to participate the SL training process and decide their working sequence. We use a variable s_k ($k = 1, 2, \dots, K$) to contain the device number, which is assigned as the k -th working device participating in the SL training process. Then, the target AI model is split into lower layers $\mathcal{M}_{\text{lw}}^{\text{APP}}$ and upper layers $\mathcal{M}_{\text{u}}^{\text{APP}}$. The bit-size

of $\mathcal{M}_{lw}^{APP}$ is b_{lw} . Then, the lower layer $\mathcal{M}_{lw}^{APP}$ is offloaded to the device whose number is s_1 . Then, the s_1 -th device extract a mini-batch from its local dataset to perform forward propagation with computing workload ($Q_{s_k} \cdot c_{lw}^{forward}$), where Q_{s_k} is the size of a mini-batch, which need to be smaller than the total amount of data ($Q_{s_k} < Q_{s_k}^{sum}$) on the s_k -th device. Then, it passes the middle parameters of forward propagation to AP with bit-size ($Q_{s_k} \cdot b_{mid, fw}$), where $b_{mid, fw}$ is the bit-size of the middle results of one training sample. Based on the received middle parameters, the ES continues to perform forward propagation for upper layers with computing workload ($Q_{s_k} \cdot c_{up, fw}$). After that, it computes backward propagation for upper layers with computing workload ($Q_{s_k} \cdot c_{up, bk}$). Then, it sends the middle parameter of backward propagation back to user device via AP, with bit-size ($Q_{s_k} \cdot b_{mid, bk}$). Next, user device computes backward propagation with computing workload ($Q_{s_k} \cdot c_{lw, bk}$). This process iterates I_k times. After that, the s_k user device would send the updated lower layers to AP, which sends them to the next participate user s_{k+1} to continue SL training process, until all participants has trained the target AI model.

2.4. Digital Twin Network

The DTN can be model as

$$\mathcal{D} = \{\mathbb{O}, \mathbb{G}, \mathbb{E}, \mathbb{Z}\}, \quad (6)$$

where $\mathbb{O}, \mathbb{G}, \mathbb{E}, \mathbb{Z}$ are network management module, network strategy module, pre-validation environment and network database, respectively. Among them, the network management module $\mathbb{O}$ is responsible for analyzing the requirements of different tasks (such as SL tasks), and orchestrating different modules to reach the requirements. Network strategy module $\mathbb{G}$ can generate resource allocation solutions based on intelligent algorithms (such as deep reinforcement learning algorithms, generative AI algorithms, etc.), whose performance can be tested in the pre-validation environment $\mathbb{E}$, and based on the testing results, the module $\mathbb{G}$ can improve its strategy. The design of network strategy module is beyond the scope of this work. Specifically, this paper focus on the establishment of pre-validation environment $\mathbb{E}$, which needs to estimate the performance of a resource allocation solution for SL training tasks, which is essential to guarantee the SL performance. Its input includes current network states and resource allocation solution; its output is the SL performance metrics such as latency and convergence. Notice that the estimating process is challenge because of unknown data distributions and misreported wireless channel qualities and available computing resources. Finally, network database $\mathbb{Z}$ can be expressed as

$$\mathbb{Z} = \{f_{av,n}^*, L_n^*, ||Q_n||, f_{AP}\}, \quad (7)$$

where the superscript “*” represents that the recorded data in the DTN has some deviation from the actual data. L_n^* is the recorded large-scale fading of the n -th device in the DTN, corresponding to a deviated root square variant σ_n^* of channel gain. Notice that the DTN does not need to collect the small-scale fading, since we deploy a small-scale power adaptation scheme on user devices and AP to deal with it in Section 2.2. $f_{av,n}^*$ is the recorded available amount of computing resources of the n -th device in the DTN, which may have error from the real amount of computing resources $f_{av,n}$. The error becomes from user misreported behavior, data collecting error, and sudden appeared emergency tasks. Under a computing resource allocation solution f_n generated by network strategy module $\mathbb{G}$, the actual occupied computing resource on the n -th device is inaccurate compared with the allocated computing resource f_n , which can be expressed as

$$f_n^* = \min\{f_{av,n}^*, f_n\}. \quad (8)$$

3. Transformer-Based Pre-Validation Model for DTN

The aim of this paper is to establish a pre-validation environment for estimating latency and convergence of SL training tasks under given network states and strategies. To do this, this section first analyzes the relationships between the estimation objects, i.e., SL

training latency and convergence, with network states and strategies from mathematical view, where several unknown parameters exist and makes the estimation process difficult. To cope with the difficulties, we propose a TransNeural algorithm which combines transformer and neural network to learn inter-nodes and inter-feature relationships with unknown parameters.

3.1. Problem Analysis for SL Convergence Estimation

The SL training process comprises iterative wireless transmission process and computing process on AP and different devices. Without causing ambiguity, we use divergence ε to replace convergence $(1 - \varepsilon)$. First, the wireless outage can cause transmission failure in middle parameter forward and backward propagations. Based on the small-scale power adaptation scheme in Section 2.2 and PDF of channel gain in Theorem 1, the outage probability in different wireless transmission process can be expressed as

$$\varepsilon_{s_k, \text{tr}}^* = \int_0^{H_L} \frac{(x/(\sigma_{s_k}^*)^2)^{M-1}}{(\sigma_{s_k}^*)^2 \cdot \Gamma(M)} e^{-x/(\sigma_{s_k}^*)^2} dx = \frac{\Gamma^D(M, H_L/(\sigma_{s_k}^*)^2)}{\Gamma(M)}. \quad (9)$$

The outage probability is inaccurate because the root square variant σ_n^* of channel gain is inaccurate. Because of outage probability in uplink and downlink, the usable amount of data in a mini-batch in one iteration changes from Q_{s_k} to $((1 - \varepsilon_{s_k, \text{tr}}^*)^2 \cdot Q_{s_k})$.

In addition, the original divergency rate in computing process can be expressed as

$$\varepsilon_{\text{cp}}^\gamma = \exp(-\mu^\gamma QI), \quad (10)$$

where μ^γ is an unknown divergence parameter depending on the data distribution on different devices. The superscript “ γ ” denotes that the value of a variable is unknown. Then, considering outage probability in wireless transmission process, the finally divergence on s_k -th device is

$$\varepsilon_{s_k}^\gamma = \exp\left(-\mu_{s_k}^\gamma \cdot (1 - \varepsilon_{s_k, \text{tr}}^*)^2 \cdot Q_{s_k} \cdot I_{s_k}\right). \quad (11)$$

The overall training divergence throughout different devices needs to consider data similarity among different devices, which can be expressed with an unknown data correlation matrix

$$C^\gamma = \begin{bmatrix} C_{1,1}^\gamma & \cdots & C_{1,N-1}^\gamma & C_{1,N}^\gamma \\ C_{2,1}^\gamma & \cdots & C_{2,N-1}^\gamma & C_{2,N}^\gamma \\ \vdots & \ddots & \vdots & \vdots \\ C_{N,1}^\gamma & \cdots & C_{N,N-1}^\gamma & C_{N,N}^\gamma \end{bmatrix}, \quad (12)$$

where $C_k^\gamma = [C_{k,1}^\gamma, C_{k,2}^\gamma, \dots, C_{k,N}^\gamma]$ is the data similarity vector of device k . Then, the overall divergence can be expressed as

$$\varepsilon_{\text{total}}^\gamma = \exp\left(-\left(\sum_{k=1}^K (1 - \varepsilon_{s_k, \text{tr}}^*)^2 \mu_{s_k}^\gamma Q_{s_k} I_{s_k} - \sum_{k=1}^{K-1} (1 - \varepsilon_{s_k, \text{tr}}^*)^2 C_{s_k, s_{k+1}}^\gamma Q_{s_k} I_{s_k}\right)\right). \quad (13)$$

3.2. Problem Analysis for SL Latency Estimation

The latency of SL training process is the sum of transmitting and computing latencies on AP and different participate devices. In detail, first, the lower layers $\mathcal{M}_{\text{lw}}^{\text{APP}}$ with bit-size b_{lw} is wirelessly transmitted to s_k -th device, where the downlink transmission latency is

$$D_{s_k, \text{tr}, \text{ofd}}^* = b_{\text{lw}} / R_{n, \text{dn}}^*, \quad (14)$$

where the transmission rate is inaccurate. The reason is that the root square variant σ_n^* of channel gain is inaccurate, which leads the calculated target power $P_{n, \text{dn}, \text{tar}}$ to be inaccurate according to Equation (4); thus, the transmission rate $R_{n, \text{dn}}^*$ is inaccurate according

to Equation (5). Then, the s_k -th user device computes forward propagation with computing latency

$$D_{s_k,lc,fw}^* = \frac{Q_{s_k} \cdot c_{lw,fw}}{f_{s_k}^*}, \quad (15)$$

where $f_{s_k}^*$ is actual allocated computing resources on the s_k -th device, which is inaccurate according to Equation (8). Then, the middle results of forward propagation is transmitted to AP with uplink transmission latency

$$D_{s_k,tr,fw}^* = \frac{Q_{s_k} \cdot b_{mid,fw}}{R_{n,up}^*}. \quad (16)$$

The AP performs forward propagation for the upper layers using the accepted middle data. Then, it computes the loss function and performs backward propagation for the upper layers. The total computing latency of this forward and backward propagation process is

$$D_{s_k,AP}^* = \frac{(1 - \varepsilon_{s_k,tr}^*) \cdot Q_{s_k} \cdot (c_{up,fw} + c_{up,bk})}{f_{AP}}, \quad (17)$$

where outage probability in uplink transmission process leads an average of $(\varepsilon_{s_k,tr} \cdot Q_{s_k})$ become unusable. After that, AP sends the backward middle results to user device with latency

$$D_{s_k,tr,bk}^* = \frac{(1 - \varepsilon_{s_k,tr}^*) \cdot Q_{s_k} \cdot b_{mid,bk}}{R_{n,dn}^*}. \quad (18)$$

Then, user device performs local backward propagation with computing latency

$$D_{s_k,lc,bk}^* = \frac{(1 - \varepsilon_{s_k,tr}^*)^2 \cdot Q_{s_k} \cdot c_{lw,bk}}{f_{s_k}^*}, \quad (19)$$

where outage probability in downlink transmission process leads another $\varepsilon_{s_k,tr}^*$ rate of unstable data. The above process iterates I_{s_k} times. Finally, user device upload lower layers to AP with latency

$$D_{s_k,tr,upld}^* = \frac{b_{lw}}{R_{n,up}^*} \quad (20)$$

Therefore, the total latency with one device is

$$D_{s_k}^* = D_{s_k,tr,ofld}^* + I_{s_k} \cdot (D_{s_k,lc,fw}^* + D_{s_k,tr,fw}^* + D_{s_k,AP}^* + D_{s_k,tr,bk}^* + D_{s_k,lc,bk}^*) + D_{s_k,tr,upld}^* \quad (21)$$

The total latency of SL training process with all participate devices is

$$D_{total}^* = \sum_k D_{s_k}^* \quad (22)$$

3.3. Proposed Transformer-Based Pre-Validation Model

3.3.1. Overall Architecture

To accurately estimate SL training latency and divergence, a pre-validation model needs to simultaneously model the inter-node relationships, the relationships between different features and estimate output, and the unknown or inaccurate parameters, which is challenging work. In detail, first, the data correlation $C_{s_k, s_{k+1}}^Y$ ($k = 1, \dots, K - 1$) between adjacent participating devices greatly influence the overall training divergence, according to Equation (13). Second, the relationships between network states (such as data similarities, wireless channel quality, bit-size of lower layers b_{lw} , etc.) and network resource allocation

variables (such as size of mini-batch Q_{s_k} , training iteration I_{s_k} , wireless transmitting powers P_n , etc.) with the training divergence and latency need to be intelligently and automatically established to realize automatic network management in DTN, where the relationships are obviously complex according to Equations (13) and (22). Third, the unknown parameters (such as divergence parameter $\mu_{s_k}^Y$) and inaccurate parameters (such as root square variant σ_n^* of channel gain) need to learn without collecting their accurate values.

To do this, we propose a TransNeural algorithm, which combines transformer with neural network to automatically learn the acquired models. The whole architecture is shown in Figure 2. First, we use neural-based feature extracting layers to automatically classify which group of features needs to model their relationships between different devices, and which group of features needs to model their inner-relationship. Then, we design encoder-based line to learn the inter-node relationships and neural-based line to learn inter-feature relationships, respectively. Finally, the outputs of two lines are aggregated using neural network to further learn the complex relationship between two lines with the estimate outputs. In addition, thanks to the added neural network in different positions, the model can learn the unknown or inaccurate parameters automatically.

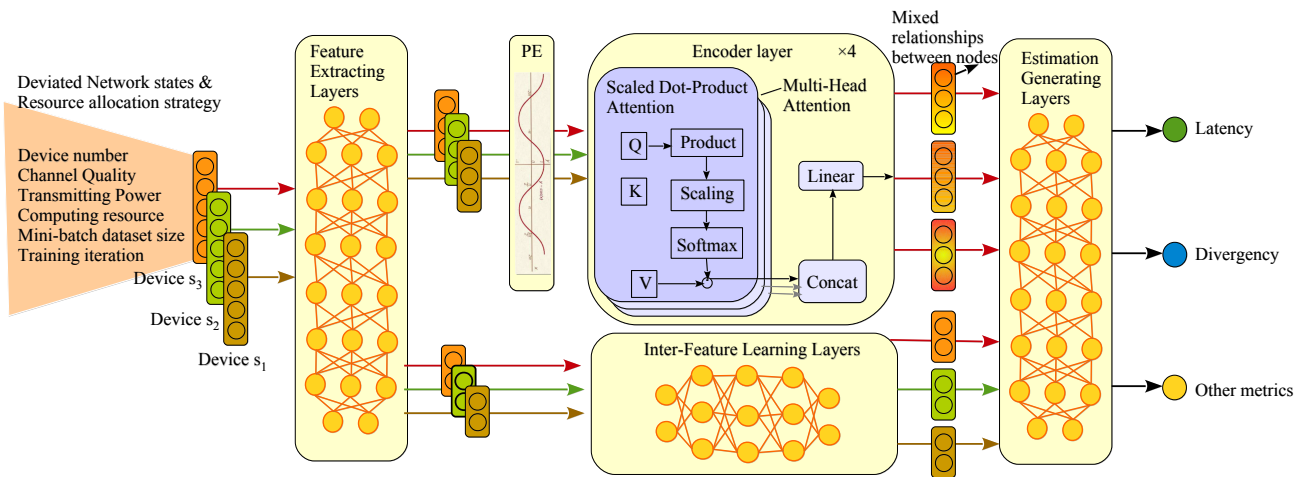


Figure 2. Proposed TransNeural algorithm by combining encoder with neural network.

The proposed TransNeural algorithm can be expressed as a function

$$\begin{aligned} \{D_i, \ln(\varepsilon_i)\} &= \text{TransNeural}([x_{i,s_1}; x_{i,s_2}; \dots; x_{i,s_K}]) \\ &= \text{TransNeural}\left(\begin{bmatrix} s_1 & H_{s_1}^* & P_{s_1} & f_{s_1}^* & Q_{s_1} & I_{s_1} \\ s_2 & H_{s_2}^* & P_{s_2} & f_{s_2}^* & Q_{s_2} & I_{s_2} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ s_K & H_{s_K}^* & P_{s_K} & f_{s_K}^* & Q_{s_K} & I_{s_K} \end{bmatrix}\right), \end{aligned} \quad (23)$$

where all of the elements are mean normalized. The loss function is defined as

$$\text{Loss} = \frac{1}{\mathbb{B}} \sum_i \left(\frac{D_i - D_{\text{real}}}{D_{\text{max}} - D_{\text{min}}} + \frac{\ln(\varepsilon_i) - \ln(\varepsilon_{\text{real}})}{\ln(\varepsilon_{\text{max}}) - \ln(\varepsilon_{\text{min}})} \right)^2, \quad (24)$$

where $\mathbb{B}$ is the size of training batch-size for the TransNeural algorithm.

3.3.2. Feature Extracting Layers

Feature extracting layers aim to automatically project different features into two groups, deciding which group of features are highly combined with inter-node relationships (such as device number), and which contribute to inter-feature learning for establishing complex relationships between network strategy and performances. Elements in different groups

may overlap. We use full connected layers to construct such feature extracting layers, which can be expressed as a function by

$$\left\{ \text{Linear} \left(\begin{bmatrix} s_1 & Q_{s_1} & I_{s_1} \\ s_2 & Q_{s_2} & I_{s_2} \\ \vdots & \vdots & \vdots \\ s_K & Q_{s_K} & I_{s_K} \end{bmatrix} \right), \text{Linear} \left(\begin{bmatrix} s_1 & H_{s_1}^* & \cdots & I_{s_1} \\ s_2 & H_{s_2}^* & \cdots & I_{s_2} \\ \vdots & \vdots & \ddots & \vdots \\ s_K & H_{s_K}^* & \cdots & I_{s_K} \end{bmatrix} \right) \right\} \quad (25)$$

$$= \text{FtrEtrL} \left(\begin{bmatrix} s_1 & H_{s_1}^* & P_{s_1} & f_{s_1}^* & Q_{s_1} & I_{s_1} \\ s_2 & H_{s_2}^* & P_{s_2} & f_{s_2}^* & Q_{s_2} & I_{s_2} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ s_K & H_{s_K}^* & P_{s_K} & f_{s_K}^* & Q_{s_K} & I_{s_K} \end{bmatrix} \right)$$

3.3.3. Positional Encoding Layer

In the SL training task, the device participation sequence greatly influences the training performance, because the initial training divergence on a new device largely depends on its data similarity with the previous training device. Therefore, we need to take the device sequence into consideration to model the relationship between strategies and SL performance. Thus, a positional encoding layer is introduced before the encoder layers. Consistent with classical settings, the expression of positional encoding is based on sine and cosine functions

$$\begin{aligned} \text{PE}_{(k,2i)} &= \sin(k/10,000^{2i/d_{\text{model}}}) \\ \text{PE}_{(k,2i+1)} &= \cos(k/10,000^{2i/d_{\text{model}}}) \end{aligned} \quad (26)$$

where k is the participating sequence of a device, d_{model} is the input dimension of encoder layer, and i denotes different input neural of the encoder layer.

3.3.4. Encoder Layers

The encoder layers use the classical components in encoders, including K, Q, V matrices to learn the inter-node relationships by

$$\text{Attention}(Q, K, V) = \text{softmax} \left(\frac{QK^T}{\sqrt{d_{\text{model}}}} \right) V \quad (27)$$

We apply multi-head attentions to learn different kind of relationships between devices by

$$\begin{aligned} \text{MultiHead}(Q, K, V) &= \text{Concat}(\text{head}_1, \dots, \text{head}_h) W^O, \\ \text{where head}_i &= \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \end{aligned} \quad (28)$$

Finally, the encoder layer can expressed as

$$\{\mu_{s_k} Q_{s_k} I_{s_k}, C_{s_k} Q_{s_k} I_{s_k}\} = \underbrace{\text{EncoderL} \left(\begin{bmatrix} s_1 & Q_{s_1} & I_{s_1} \\ s_2 & Q_{s_2} & I_{s_2} \\ \vdots & \vdots & \vdots \\ s_K & Q_{s_K} & I_{s_K} \end{bmatrix} \right)}_{\text{Modeling data similarities } C_{s_k} \text{ and divergency parameter } \mu_{s_k}} \quad (29)$$

3.3.5. Inter-Feature Learning Layers

In the second line, we design inter-feature learning layers to non-linearly produce some key elements for estimating SL performance based on a part of the features. Considering that neural networks are inherently good at extracting the features of different levels in their layers to finally establish the complex relationships between input and output, we simply use fully connected neural network to construct the inter-feature learning layers. That is,

in our scenario, the inter-feature learning layers can output wireless outage probability based on the input channel quality, and output SL latency based on the input transmitting power, computing resources, etc. At the same time, it can automatically modify the error in inaccurate computing resource and channel gain based on the device number. The function can be expressed as

$$\{D_i, [\varepsilon_{s_1, \text{tr}}, \varepsilon_{s_2, \text{tr}}, \dots, \varepsilon_{s_K, \text{tr}}]\} = \text{InFtrL} \left(\underbrace{\begin{bmatrix} s_1 & H_{s_1}^* & P_{s_1} & f_{s_1}^* & Q_{s_1} & I_{s_1} \\ s_2 & H_{s_2}^* & P_{s_2} & f_{s_2}^* & Q_{s_2} & I_{s_2} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ s_K & H_{s_K}^* & P_{s_K} & f_{s_K}^* & Q_{s_K} & I_{s_K} \end{bmatrix}}_{\text{Modeling relationships and learn accurate } H_{s_k}, f_{s_k}} \right) \quad (30)$$

3.3.6. Estimation Generating Layer

Two lines are finally combined in the fully connected layers, which would jointly consider the inter-node relationships and inter-feature relationships to produce the final estimated system performance, i.e., the SL training latency and accuracy in this scenario.

4. Simulation

In this section, we evaluate the proposed TransNeural algorithm in Python 3.7.6. We suppose that there are N ($N \in [10, 30]$) devices access to an AP, where the path loss model is $(38.46 + 20 \log_{10}(d))$ [25], d is the distance between device and AP. The communication frequency is 2.6 GHz. Each device has a unique identity number from 1 to N , which is uniquely projected to a virtual device with that number in DTN. In addition, each device has a series of unique characteristics, which are unknown and about to be automatically learned by DTN models. Their values are randomly set as follows: distance $d_k^* \in [10, 900]$ m, average reporting deviation on distance $\Delta d_k^\gamma \in [1, 10]$ m, average deviation between allocated computing resource and actual provided computing resource $\Delta f_k^\gamma \in [0.1, 1]$ GHz, divergence related parameter $\mu_k^\gamma \in [1, 10]$, each element in data similarity vector $C_k^\gamma \in \{[1, 5]\}^N$. From the parameter settings, it can be seen that the range of value for every parameter is obviously large, with the aim to include as many as network conditions as possible. Notice that, in an SL training process, not all devices will be selected to join the training process. However, the DTN model needs to learn the characteristics of all device to better estimate the training performance for every given resource allocation strategies. Without causing ambiguity, we use SL divergence to replace SL convergence in the simulation section, to make analysis more clearly. Other parameters are given in Table 2.

Table 2. Parameter settings.

Parameter	Value	Parameter	Value
Q_k	[100, 1000]	I_k	[5, 100]
f_k	[0.1, 2] G cycles/s	f_B	100 G cycles/s
B	10 MHz	K	64
N_0	$10^{-20.4}$	H_L^k	140×10^{-6}
P_k	[0.01, 2] W	P_B	88 W
$b_{\text{mid}, \text{fw}}$	1 Mbits	$b_{\text{mid}, \text{bk}}$	1 Mbits
$c_{\text{up}, \text{fw}}$	100 M cycles	$c_{\text{up}, \text{bk}}$	100 M cycles
$c_{\text{lower}}^{\text{forward}}$	1 M cycles	$c_{\text{lw}, \text{bk}}$	1 M cycles
b_{lw}	1 Gbits		

Details of model in TransNeural algorithm are setting as follows. The dimension of input array is 5, which is the feature dimension of one device. The structure of the feature

extracting layer is 64×2 , whose first layer and second layer take ReLu function and Linear function as the activation functions, respectively. The encoder block has four encoder layers, where each layer has eight heads for multi-head attention, and the dimension of input vector is 64. The structure of inter-feature learning layer is 64×2 , whose layers take ReLu function as the activation functions. The structure of estimation generating layer is 64×2 , whose first layer and second layer take ReLu function and Linear function as the activation functions, respectively.

The baselines are set as follows:

- **Equation-based algorithm:** this algorithm estimates latency and divergency based on the equations in Sections 3.1 and 3.2.
- **LSTMNeural algorithm:** We merge the long short-term memory (LSTM) and neural network to form a LSTMNeural algorithm. That is, in the architecture of proposed TransNeural algorithm, the PE layer and encoder layer are replaced by an LSTM. The input dimension of the LSTM is 16. The architecture of its hidden layer is 32×4 .

Figure 3 gives learning convergence of the LSTMNeural and proposed TransNeural algorithm. The figure indicates that the TransNeural algorithm converges faster than the LSTMNeural algorithm, and can acquire a lower loss. The reason lies in the fact that, on each data sample, LSTM can only learn the data correlations between adjacent participate devices, while transformer can learn the data correlations between all of the participate devices. Therefore, TransNeural algorithm has a much higher learning efficiency to estimate network performance. In addition, the figure indicates that the TransNeural algorithm converges fast under different size of SL participant group.

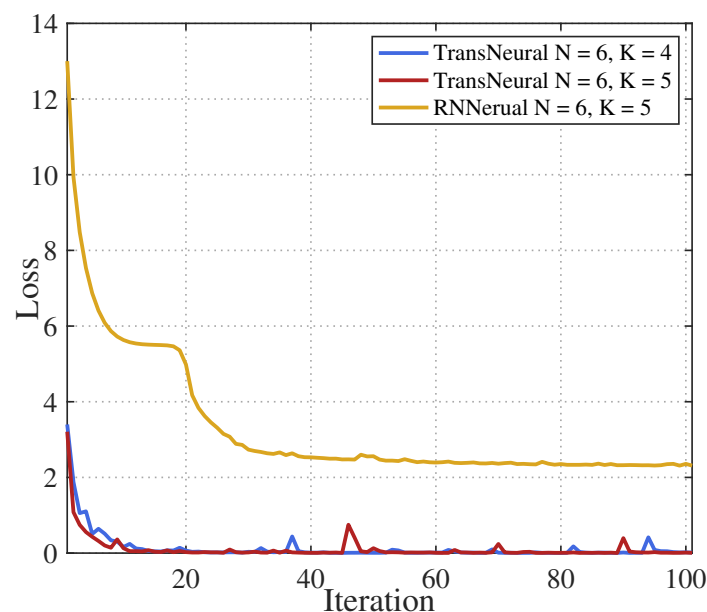


Figure 3. Learning convergence of proposed TransNeural algorithm.

Figures 4 and 5 give the total SL latency and estimation error with growing average distance between selected user devices and AP. In general, the wireless transmission rate will first drop as the distance grows. Considering the wireless transmission happens frequently in each forward and backward propagation process, the transmission delay will significantly influence the total SL training delay, which leads the total SL delay increases as the distance grows. From the figure, since the DTN may not acquire accurate CSI, latency estimation based on traditional equation-based method will have a high error, especially under larger distance where CSI error becomes sensitive for latency estimation. In comparison, the proposed TransNeural algorithm has a stable error with distance growing, which stays under 240 s per omit-table because the total error is 2100–2300 s. However, LSTMNeural has a large estimation error because of its low learning efficiency.

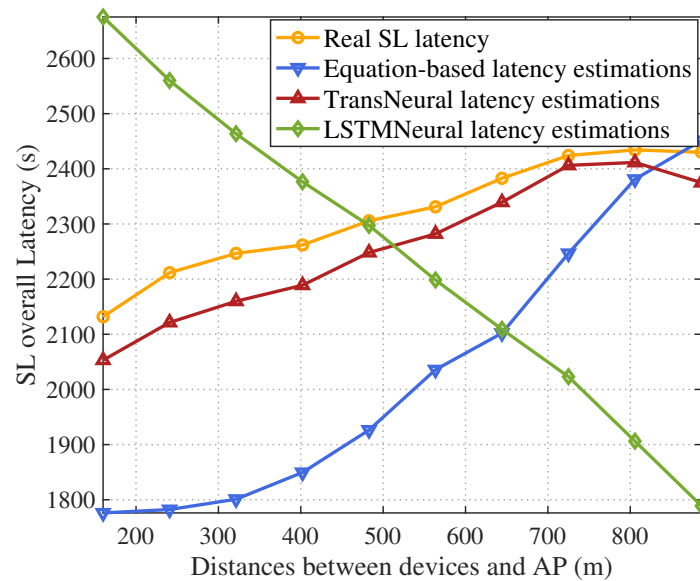


Figure 4. SL training latency vs. distances between devices and AP.

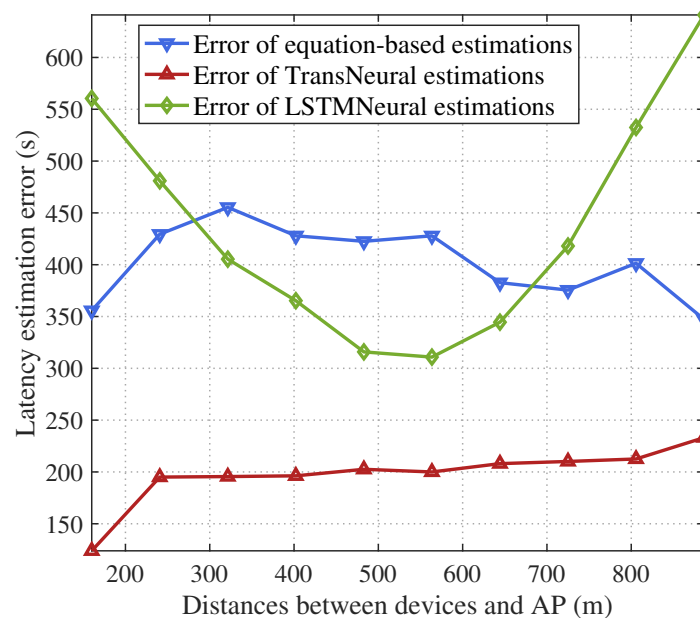


Figure 5. Estimation error of SL training latency vs. distances between devices and AP.

Figures 6 and 7 give the total SL latency and estimation error with a growing allocated computing resources on devices. Because the user device needs to use their allocated computing resources to compute forward and backward propagation of lower layers in SL target models, the computing latency will decrease with the increasing allocated computing resources, which leads the total latency decrease. However, because of user misreport information, the real allocated computing resources may be smaller than the amount of allocated resource in network strategies, which can not be modeled in traditional equation-based latency estimation methods, leading to estimation errors. In comparison, the proposed TransNeural algorithm and LSTMNeural algorithm can automatically learn these misreport behaviors from historical data, thus better estimate the real latency than traditional methods. Moreover, the TransNeural outperforms LSTMNeural because of high learning efficiency.

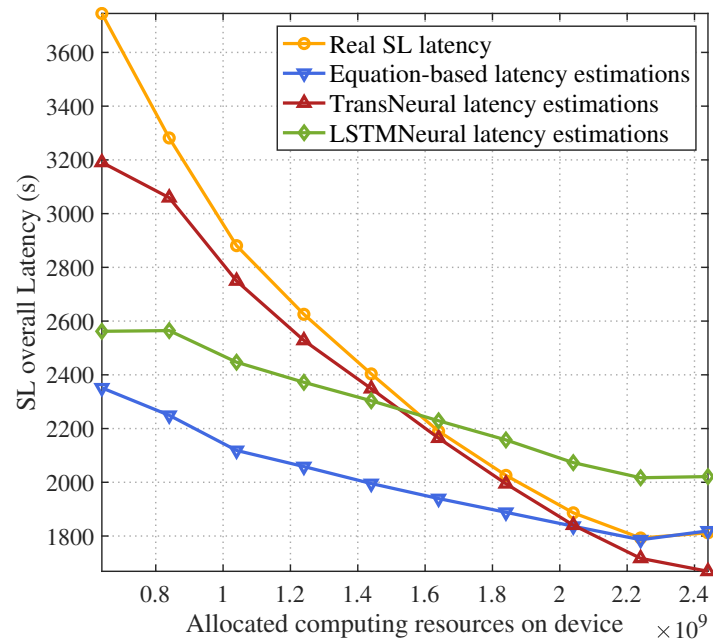


Figure 6. SL training latency vs. computing resources on devices.

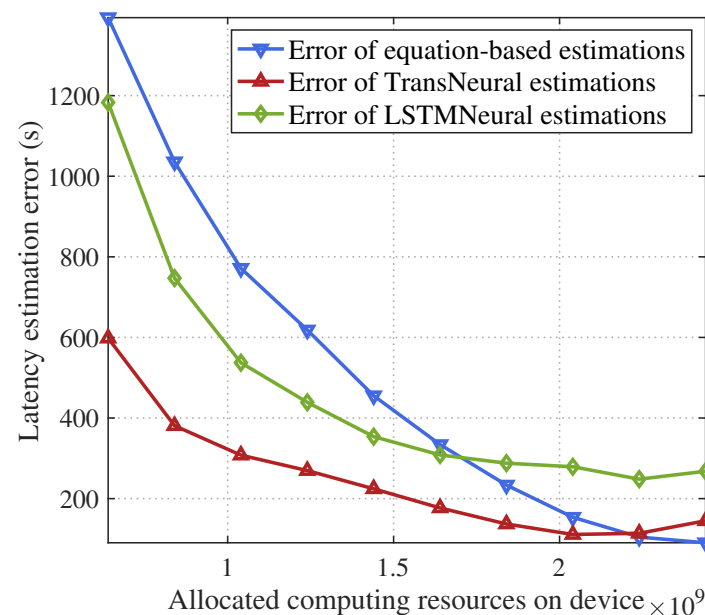


Figure 7. Estimation error of SL training latency vs. computing resources on devices.

Figure 8 gives the natural logarithm of the training divergence with growing average size of mini-batch used for training on user devices. The divergence drops as the size of mini-batch grows. Since the equation-based algorithm cannot learn the data correlations between device and takes the average correlation value to estimate divergence, its estimation has a large error compared with two other algorithms. In comparison, the proposed TransNeural algorithm can accurately predict the divergence under various size of mini-batch, thanks to its high learning ability. In addition, the figure indicates that the LSTMNeural algorithm has a higher estimation error than the proposed TransNeural algorithm. This is because that the LSTM algorithm cannot leverage dataset to learn data correlations between different devices efficiently compared with the transformer algorithm, as discussed earlier.

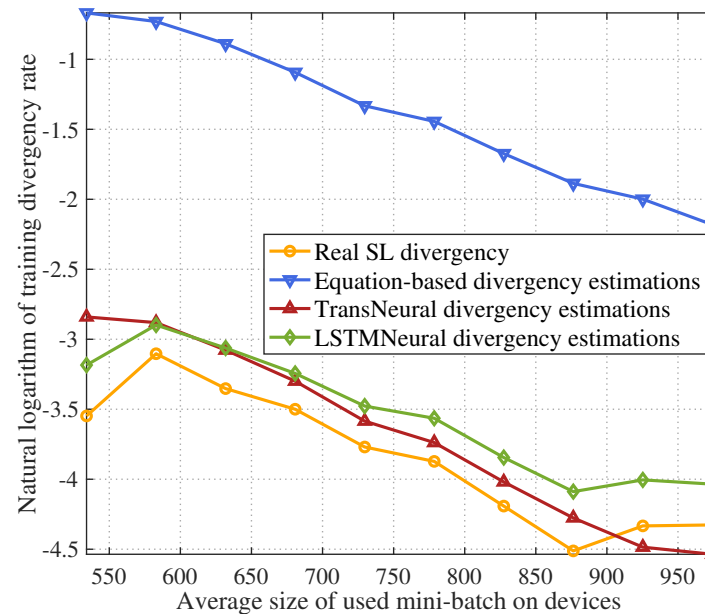


Figure 8. SL training divergence rate vs. average size of mini-batch used for training on devices.

Figure 9 gives the natural logarithm of training divergence with growing average distances between selected devices and AP. As distance grows, outage probability in wireless transmission process increases. It greatly influences the SL training divergence considering the dense wireless transmission process exists in SL training, which is also proved in the figure. In traditional equation-based algorithm, inaccurate CSI leads to a high divergence estimation error compared with the other two algorithm, which could eliminate error because of integrated neural network.

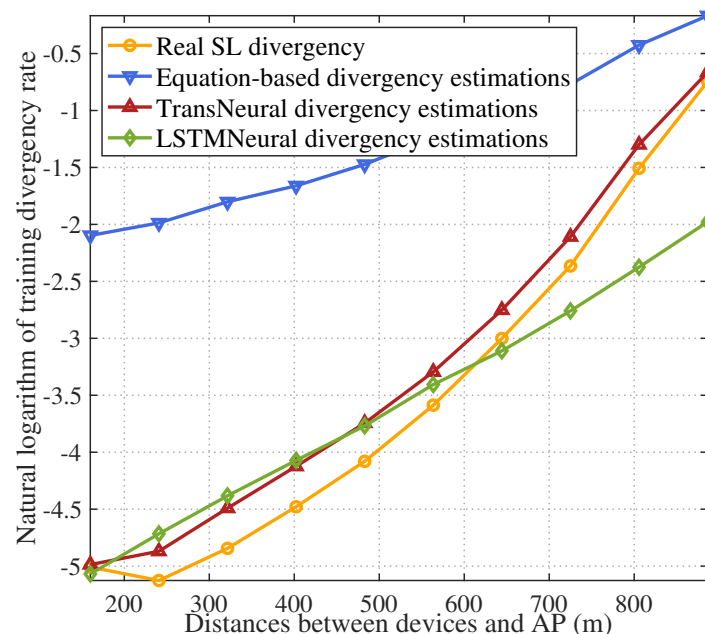


Figure 9. SL training divergence rate vs. distances between devices and AP.

Figure 10 gives the natural logarithm of training divergence with growing average training iterations among participating devices. In general, the divergence decreases as the training iteration increasing. The equation-based algorithm has a higher estimating error compared with two other algorithms because of lacking accurate data correlation parameters. Moreover, the proposed TransNeural algorithm outperforms the LSTMNeural algorithm because of high learning efficiency on given dataset.

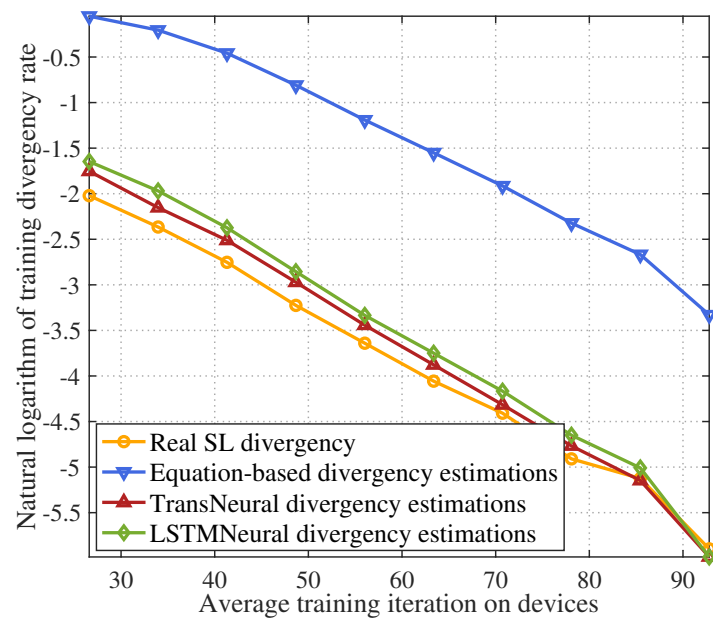


Figure 10. SL training divergency rate vs. average training iterations among devices.

Table 3 compares the estimation accuracy on SL training latency and divergence under different algorithms. As for the SL latency, the proposed TransNeural algorithm increases the estimating accuracy by 9.3% and 6.5% compared with the equation-based algorithm and LSTMNeural algorithm, respectively. As for the SL convergence, the proposed TransNeural algorithm increases estimating accuracy by 22.41% and 2.34% compared with the equation-based algorithm and LSTMNeural algorithm, respectively. The table proves that the proposed algorithm can effectively improve the estimation accuracy for various types of SL metrics.

Table 3. Comparison of deviant ratio for estimated latency and divergence in SL training process.

Latency Estimation (s)		Accuracy
Actual SL latency	2316.0	/
TransNeural algorithm	2114.8	91.3%
LSTMNeural algorithm	1961.3	84.7%
Equation-based algorithm	1897.9	82.0%
Convergence Estimation		Accuracy
Actual SL convergence	97.82%	/
TransNeural algorithm	95.89%	98.03%
LSTMNeural algorithm	93.60%	95.69%
Equation-based algorithm	73.97%	75.62%

Tables 4 and 5 verify the algorithm scalability. In detail, Table 4 shows that the proposed TransNeural algorithm remains a high latency estimation accuracy with growing number of devices. The estimation accuracies under different numbers of devices exhibit fluctuations, since different data distributions may result in different degrees of learning difficulties. Fortunately, they are all higher than 90%, which verifies the scalabilities of the proposed TransNeural algorithm. Similarly, Table 5 shows that the proposed TransNeural algorithm remains a high convergence estimation accuracy with growing number of devices. That is, all of the convergence estimation accuracies under different number of devices are high than 95%, which again proves the well scalability of proposed TransNeural algorithm.

Table 4. Accuracy of latency estimation under different participant scales.

$N = 10, K = 5$	$N = 15, K = 10$	$N = 20, K = 15$	$N = 25, K = 20$	$N = 30, K = 25$
91.3%	93.8%	98.4%	99.5%	97.3%

Table 5. Accuracy of convergence estimation under different participant scales.

$N = 10, K = 5$	$N = 15, K = 10$	$N = 20, K = 15$	$N = 25, K = 20$	$N = 30, K = 25$
98.0%	99.1%	99.7%	96.0%	97.2%

5. Conclusions

The paper closes the gap of studying the pre-validation environment for SL training tasks. It proposes a TransNeural algorithm to estimate the latency and divergency of SL training process under given resource allocation solution. In detail, the TransNeural algorithm integrates the transformer and neural network to collaboratively learn data similarities, complex relationships between SL performance (latency, divergency, etc.) and participants selections, wireless/computing resource allocation, and the reported deviation on wireless channels and available computing resources. Simulation shows the proposed TransNeural algorithm can effectively improve the estimating accuracy by 9.3% on latency and 22.4% on divergency compared with traditional equation-based algorithms.

Author Contributions: Conceptualization, G.L.; methodology, G.L., M.K., Y.Z. and N.L.; software, M.K. and M.Z.; validation, M.K. and Q.Z.; formal analysis, G.L., M.K. and Y.Z.; investigation, Y.Z.; resources, G.L. and Y.Z.; data curation, M.K.; writing—original draft preparation, G.L. and M.K.; writing—review and editing, Y.Z. and M.K.; visualization, M.K.; supervision, G.L. and Y.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded by the National Key R&D Program of China (2022YFB2902100).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data can be found in the article.

Conflicts of Interest: Author Guangyi Liu was employed by the company China Mobile Research Institute. Author Mancong Kang is pursuing doctor degree in the School of Communications and Information Engineering at Beijing University of Posts and Telecommunications. Author Yanhong Zhu was employed by the company China Mobile Research Institute, and was pursuing doctor degree in the School of Electronics and Information Engineering at Beijing Jiaotong University. Author Qingbi Zheng was employed by the company China Mobile Research Institute and ZGC Institute of Ubiquitous-X Innovation and Application. Author Maosheng Zhu was employed by the company China Mobile (Suzhou) Software Technology Co., Ltd. Author Na Li was employed by the company China Mobile Research Institute and ZGC Institute of Ubiquitous-X Innovation and Application. The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

1. An, K.; Sun, Y.; Lin, Z.; Zhu, Y.; Ni, W.; Al-Dhahir, N.; Wong, K.K.; Niyato, D. Exploiting Multi-Layer Refracting RIS-Assisted Receiver for HAP-SWIPT Networks. *IEEE Trans. Wirel. Commun.* **2024**, *1*. [\[CrossRef\]](#)
2. Lin, Z.; Lin, M.; de Cola, T.; Wang, J.B.; Zhu, W.P.; Cheng, J. Supporting IoT With Rate-Splitting Multiple Access in Satellite and Aerial-Integrated Networks. *IEEE Internet Things J.* **2021**, *8*, 11123–11134. [\[CrossRef\]](#)
3. Wang, Y.; Yang, C.; Lan, S.; Zhu, L.; Zhang, Y. End-Edge-Cloud Collaborative Computing for Deep Learning: A Comprehensive Survey. *IEEE Commun. Surv. Tutor.* **2024**, *1*. [\[CrossRef\]](#)
4. Wen, J.; Zhang, Z.; Lan, Y.; Cui, Z.; Cai, J.; Zhang, W. A survey on federated learning: Challenges and applications. *Int. J. Mach. Learn. Cybern.* **2023**, *14*, 513–535. [\[CrossRef\]](#) [\[PubMed\]](#)
5. Shen, X.; Liu, Y.; Liu, H.; Hong, J.; Duan, B.; Huang, Z.; Mao, Y.; Wu, Y.; Wu, D. A Split-and-Privatize Framework for Large Language Model Fine-Tuning. *arXiv* **2023**, arXiv:2312.15603.
6. Lin, Z.; Qu, G.; Chen, X.; Huang, K. Split Learning in 6G Edge Networks. *IEEE Wirel. Commun.* **2024**, *31*, 170–176. [\[CrossRef\]](#)
7. Satpathy, S.; Khalaf, O.; Kumar Shukla, D.; Chowdhary, M.; Algburi, S. A collective review of Terahertz technology integrated with a newly proposed split learningbased algorithm for healthcare system. *Int. J. Comput. Digit. Syst.* **2024**, *15*, 1–9.
8. Wu, W.; Li, M.; Qu, K.; Zhou, C.; Shen, X.; Zhuang, W.; Li, X.; Shi, W. Split Learning Over Wireless Networks: Parallel Design and Resource Management. *IEEE J. Sel. Areas Commun.* **2023**, *41*, 1051–1066. [\[CrossRef\]](#)
9. Lin, Z.; Zhu, G.; Deng, Y.; Chen, X.; Gao, Y.; Huang, K.; Fang, Y. Efficient Parallel Split Learning over Resource-constrained Wireless Edge Networks. *IEEE Trans. Mob. Comput.* **2024**, 1–16. [\[CrossRef\]](#)

10. Khan, L.U.; Saad, W.; Niyato, D.; Han, Z.; Hong, C.S. Digital-Twin-Enabled 6G: Vision, Architectural Trends, and Future Directions. *IEEE Commun. Mag.* **2022**, *60*, 74–80. [[CrossRef](#)]
11. Kuruvatti, N.P.; Habibi, M.A.; Partani, S.; Han, B.; Fellan, A.; Schotten, H.D. Empowering 6G Communication Systems With Digital Twin Technology: A Comprehensive Survey. *IEEE Access* **2022**, *10*, 112158–112186. [[CrossRef](#)]
12. Almasan, P.; Galmés, M.F.; Paillisse, J.; Suárez-Varela, J.; Perino, D.; López, D.R.; Perales, A.A.P.; Harvey, P.; Ciavaglia, L.; Wong, L.; et al. Digital Twin Network: Opportunities and Challenges. *arXiv* **2022**, arXiv:2201.01144.
13. Lai, J.; Chen, Z.; Zhu, J.; Ma, W.; Gan, L.; Xie, S.; Li, G. Deep learning based traffic prediction method for digital twin network. *Cogn. Comput.* **2023**, *15*, 1748–1766. [[CrossRef](#)] [[PubMed](#)]
14. He, J.; Xiang, T.; Wang, Y.; Ruan, H.; Zhang, X. A Reinforcement Learning Handover Parameter Adaptation Method Based on LSTM-Aided Digital Twin for UDN. *Sensors* **2023**, *23*, 2191. [[CrossRef](#)] [[PubMed](#)]
15. Ferriol-Galmés, M.; Suárez-Varela, J.; Paillissé, J.; Shi, X.; Xiao, S.; Cheng, X.; Barlet-Ros, P.; Cabellos-Aparicio, A. Building a Digital Twin for network optimization using Graph Neural Networks. *Comput. Netw.* **2022**, *217*, 109329. [[CrossRef](#)]
16. Wang, H.; Wu, Y.; Min, G.; Miao, W. A Graph Neural Network-Based Digital Twin for Network Slicing Management. *IEEE Trans. Ind. Inform.* **2022**, *18*, 1367–1376. [[CrossRef](#)]
17. Tam, D.S.H.; Liu, Y.; Xu, H.; Xie, S.; Lau, W.C. PERT-GNN: Latency Prediction for Microservice-based Cloud-Native Applications via Graph Neural Networks. In Proceedings of the KDD '23: 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, New York, NY, USA, 6–10 August 2023; pp. 2155–2165. [[CrossRef](#)]
18. Wang, H.; Xiao, P.; Li, X. Channel Parameter Estimation of mmWave MIMO System in Urban Traffic Scene: A Training Channel-Based Method. *IEEE Trans. Intell. Transp. Syst.* **2024**, *25*, 754–762. [[CrossRef](#)]
19. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. In Proceedings of the 31st International Conference on Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; Volume 30.
20. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444. [[CrossRef](#)]
21. Popovski, P.; Trillingsgaard, K.F.; Simeone, O.; Durisi, G. 5G wireless network slicing for eMBB, URLLC, and mMTC: A communication-theoretic view. *IEEE Access* **2018**, *6*, 55765–55779. [[CrossRef](#)]
22. Hou, Z.; She, C.; Li, Y.; Zhuo, L.; Vucetic, B. Prediction and Communication Co-Design for Ultra-Reliable and Low-Latency Communications. *IEEE Trans. Wirel. Commun.* **2020**, *19*, 1196–1209. [[CrossRef](#)]
23. Schiessl, S.; Gross, J.; Skoglund, M.; Caire, G. Delay Performance of the Multiuser MISO Downlink under Imperfect CSI and Finite Length Coding. *IEEE J. Sel. Areas Commun.* **2019**, *37*, 765–779. [[CrossRef](#)]
24. Kang, M.; Li, X.; Ji, H.; Zhang, H. Digital twin-based framework for wireless multimodal interactions over long distance. *Int. J. Commun. Syst.* **2023**, *36*, e5603. [[CrossRef](#)]
25. 3GPP 5G. *Physical Layer Procedures for Data (Release 16)*; Technical Report, 3GPP TS 36.214; ETSI: Sophia Antipolis, France, 2020.

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.