Split Learning on Segmented Healthcare Data

Ling Hu, Tongqing Zhou, Zhihuang Liu, Fang Liu, and Zhiping Cai, *Member, IEEE*

***Abstract*—Sequential data learning is vital to harnessing the encompassed rich knowledge for diverse downstream tasks, particularly in healthcare (e.g., disease prediction). Considering data sensitiveness, privacy-preserving learning methods, based on federated learning (FL) and split learning (SL), have been widely investigated. Yet, this work identifies, for the first time, existing methods overlook that sequential data are generated by different patients at different times and stored in different hospitals, failing to learn the sequential correlations between different temporal segments. To fill this void, a novel distributed learning framework STSL is proposed by training a model on the segments in order. Considering that patients have different visit sequences, STSL first implements privacy-preserving visit ordering based on a secure multi-party computation mechanism. Then batch scheduling participates patients with similar visit (sub-)sequences into the same training batch, facilitating subsequent split learning on batches. The scheduling process is formulated as an NP-hard optimization problem on balancing learning loss and efficiency and a greedy-based solution is presented. Theoretical analysis proves the privacy preservation property of STSL. Experimental results on real-world eICU data show its superior performance compared with FL and SL ($5\% \sim 28\%$ better accuracy) and effectiveness (a remarkable 75% reduction in communication costs).**

***Index Terms*—Sequential data analysis, distributed machine learning, split learning, data privacy, healthcare data.**

I. Introduction

THE processing and analysis of healthcare data have always been an essential focus in medical informatics towards assisting treatment and enhancing the quality of care delivery. The healthcare data (e.g., blood albumin) is featured as time series data, so-called sequential data, in nature because the health condition of patients changes over time and the treatments may often be long-term [1]. Nurtured by such data, a considerable body of research has explored building deep learning models for predicting disease/mortality risk [2], analyzing treatment [3], discovering new drugs [4], etc.

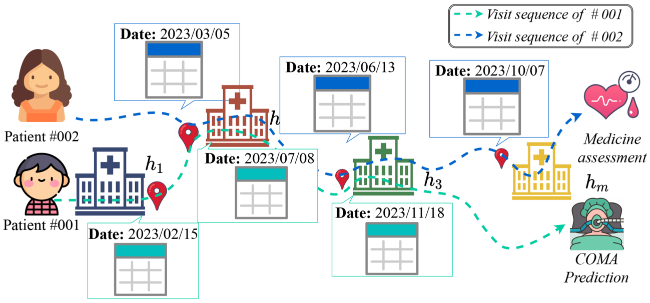


Fig. 1. An illustrative example for segmented healthcare data.

The assurance of patient privacy is the premise of exploring healthcare data [5] and lots of efforts have been devoted to this issue [6]. From the aspect of data management, there is a drive to establish cross-nation cohorts for large-scale healthcare database [7], but the process of collecting ethical consent is not trivial. As to technical advances, we have witnessed the proposal of federated learning (FL) [4] and Split Learning (SL) [8], [9] in recent years, both dedicated to privacy-preserving distributed data learning. In essence, FL has a model trained at different clients with private data and aggregates the locally trained models in a global server, while SL partitions the model into consecutive chunks and has a client and server collaboratively train the chunks. Both paradigms have been widely adopted in healthcare informatics [1], owing to their advantages of retaining sensitive patient data in the trust zone (i.e., the hospital that stores it).

However, we note that existing FL and SL designs overlook the spatial characteristics of sequential healthcare data by inherently assuming that the comprehensive data sequence of a patient is stored in one institute/hospital. In practice, patients would often visit several different (specialized) hospitals depending on their disease progression and the hierarchical medical services they use. As reported, in 2022, Chinese residents visit healthcare facilities six times on average [10]. In China alone, the number of cross-province visits reached 110.5 million in 2022 [11]. As a result, the temporal data sequence of a patient is spatially distributed in several hospitals, which is denoted as *segmented healthcare data*. For instance, in Fig. 1, each patient has a visit sequence of 3 different hospitals for his/her diseases with clinical records of 3 segments. Traditional FL and SL are ill-suited to such segmented data: *FL trains data in each hospital independently, thus missing the temporal knowledge of disease progression across hospitals. SL, although seems to suit distributed segments with distributed model chunks, partition models in static order and number of chunks, which*

Received 31 August 2024; revised 9 February 2025; accepted 17 March 2025. Date of publication 31 March 2025; date of current version 4 September 2025. This work was supported in part by the National Key Research and Development Program of China under Grant 2022YFF1203001, in part by the National Natural Science Foundation of China under Grant 62172155, Grant 62472434, and Grant 62102425, and in part by the Science and Technology Innovation Program of Hunan Province under Grant 2022RC3061 and Grant 2023RC3027. Recommended for acceptance by N. Cao. *(Ling Hu and Tongqing Zhou contributed equally to this work.) (Corresponding authors: Zhihuang Liu; Zhiping Cai.)*

Ling Hu, Tongqing Zhou, Zhihuang Liu, and Zhiping Cai are with the College of Computer Science and Technology, National University of Defense Technology, Changsha, Hunan 410073, China (e-mail: linghu50@nudt.edu.cn; zhoutongqing@nudt.edu.cn; lzhliu@nudt.edu.cn; zpcai@nudt.edu.cn).

Fang Liu is with the School of Design, Hunan University, Changsha, Hunan 410073, China (e-mail: fangl@hnu.edu.cn).

Digital Object Identifier 10.1109/TBDATA.2025.3556639

couldn't cater to various visit sequences of real-world clinical practices.

This work first investigates the privacy-preserving learning problem on segmented healthcare data. Considering the data split situation, we propose to adapt SL to support the distributed learning process and design a novel framework `STSL` (i.e., Spatio-temporal Split Learning). At its core, `STSL` performs: 1) visit ordering to identify the sequence of distributed data segments of all users/patients; 2) spatially groups patients into training batches according to visit sequence (i.e., patients in a batch have the same sequence); and 3) temporally partitions models to hospitals for split learning. Yet, the construction of `STSL` is hindered by two challenges:

C1 *How to protect patients' privacy from leaking to other entities?* The healthcare records and visit time of patients are sensitive information and should be protected according to privacy and ethical regulations [5]. Using SL essentially assures data privacy, while its prerequisite ordering step may incur the leakage of the visit time.

C2 *How to jointly accommodate learning efficiency and utility w.r.t. training batch construction?* Intuitively, fewer batches incur smaller communication for distributing and integrating sub-models but would exclude more data segments for aligning the inner-batch visit sequence. As with the example in Fig. 1, we have to discard the data of patient #001 in hospital h_1 and that of patient #002 in hospital h_m, when scheduling them into one batch for one-shot learning. However, this would cause the assessment model to miss the prediction knowledge by a large margin (abnormal indicators appear in hospital h_m). Proper balance must be attained in batch scheduling.

To relieve C1, `STSL` designs a privacy-preserving visit ordering mechanism, which guarantees that patients' visit time information is retained in its original hospital and constructs a healthcare tree for efficient retrieval. Meanwhile, `STSL` formulates the batch scheduling process as an optimization problem on communication costs and data loss. We prove the NP-hardness of this problem and propose a greedy-based solution to selectively schedule data segments into batches. Furthermore, this is the first instance where temporally and spatially distributed healthcare data has been considered in privacy-preserving data analysis. As a result, there may not be any suitable state-of-the-art to compare it with. To address this, we have created a segmented healthcare dataset by synthesizing real-world ICU data and compared the performance of `STSL` with conventional FL and modified SL. The main contributions of this work are as follows:

- This is the first work that investigates privacy-preserving learning on spatio-temporal distributed sequential data (i.e., segmented data), particularly dedicated to the healthcare context.
- A novel framework `STSL` is presented for protecting privacy with secure ordering and distributed learning and attaining split learning on segmented data with batch scheduling.
- We formulate the scheduling in `STSL` as a penalty-aware optimization problem and use selective data discarding and combinatorial optimization to address it.
- A formal proof of privacy preservation is provided, and experimental results demonstrate the superior accuracy of `STSL` compared to the FL and SL baselines, thereby illustrating the effectiveness of extracting knowledge from distributed data segments.

II. Related Work

Nowadays, numerous deep learning models have been proposed for analyzing and predicting healthcare data. This section begins by highlighting facts on healthcare data. It then proceeds to discuss various privacy-preserving distributed learning frameworks that currently exist, showcasing their workflows and applicable scenarios. Finally, it illustrates methods on sequence data profiling, providing insights into their effectiveness for healthcare applications.

A. Sequential Healthcare Data Learning

Patients usually pay visits to hospitals multiple times for healthcare services. Meanwhile, with the increased mobility of the population, people tend not to be confined to a single hospital for medical treatment. As a result, healthcare data is characterized by multiple origins [12], massive volumes [13], and sequential nature [14]. Recently, sequential healthcare data has been collected by countries worldwide, with the United States establishing eICU database [15], Australia initiating "My Health Record" project [6], China establishing ICU database [16], and so on.

Sequential healthcare data learning, combining sequential healthcare data and machine learning, can offer assistance in clinical decision-making [12] (e.g., disease prediction by profiling patients with their sequential data). For structured electronic healthcare records (EHRs), [17] proposed a data-driven approach, which encoded longitudinal temporal information, to identify Parkinson's disease. Timeline [3] employed an attention mechanism to learn the weights of patients' multiple visits and formulated a time-aware disease progression function to simulate the decay of healthcare information over time. HiTANet [18] integrated both local and global attention mechanisms and incorporated time vectors as part of the input to consider the monotonicity of time decay. For unstructured healthcare data, CGNet [2] proposed to integrate graph knowledge for feature reconstruction in the detection of pneumonia, and VecoCare [19] employed joint representation learning on structured EHRs and unstructured clinical notes. As illustrated in Table I, though these approaches enhance model effectiveness in medical tasks, they primarily rely on gathering data for centralized learning, which neglects the pivotal privacy concerns in healthcare services.

B. Privacy-Preserving Distributed Learning

Privacy concerns are often raised with centralized data storage, which may not comply with regulations such as GDPR [5]. Therefore, researchers prefer distributed frameworks that preserve privacy, such as federated learning [20] and split learning [21] for sequential data learning in healthcare. Federated learning (FL) is a distributed learning framework that enables

TABLE I
COMPARISONS ON MAINSTREAM LEARNING FRAMEWORKS AND OUR STSL FRAMEWORK

Framework	Design Considerations and Capabilities				Examples
	Data Privacy	Model Privacy	Resource-constrained Clients	Segmented Data Exploration	

Centralized Learning	○	○	N/A	○	Timeline [3];
					HiTANet [18];
					VecoCare [19].

Federated Learning	●	○	○	○	MELLODDY [4].

Split Learning	●	●	●	○	SL-1DCNN [8];
					MS-SL [9];
					LSTMSPLIT [26];
					FedSL [27].

STSL (Ours)	●	●	●	●	STSL

●Fully Support ○Not Support

individual clients to train a complete machine learning network with local data. This process involves parallel training of local models for a certain number of epochs, followed by the aggregation of these updates at the server to form a global model. Once the global model has been created, it is shared among all clients for further training in the next round. This iterative process continues until the algorithm converges, resulting in a robust and accurate model. Moreover, many efforts have been made to enhance the performance of FL [22], [23] and ensure privacy [24]. Nowadays, some works [25] achieve privacy-preserving training on biomedical data with the help of FL.

In addition to safeguarding data privacy, split learning (SL) is an innovative framework designed to alleviate the computation costs on clients. By dividing the deep learning network w into the server-side deep layers (w_s) and the client-side shallow layers (w_c), split learning ensures model parameter privacy and enables clients to offload computationally intensive tasks to the server, thereby reducing computation costs on resource-constrained clients. To connect w_C and w_S, the activation (so-called smashed data) of the split layer, known as the cut layer, in the client-side shallow layers is sent to the server. In return, the server conducts forward propagation with these smashed data, calculates gradients, conducts backward propagation, and sends the gradients back to the client.

Although FL and SL provide privacy protection, they assume that clients' datasets are independent and lack correlations, implying that a patient's comprehensive data sequence is stored within a single hospital.

C. Privacy-Preserving Sequence Profiling

As shown in Table I, some attempt to utilize the federated learning framework for sequential data learning. The most notable work is MELLODDY [4], which has successfully implemented industry-scale federated learning to support drug discovery, yielding promising outcomes. However, MELLODDY focuses on situations where the comprehensive data sequence of a patient is stored in one institute/hospital. The others adopt the split learning framework. SL-1DCNN [8] investigated the effectiveness of 1D CNN with split learning for sequential data learning. Nevertheless, SL-1DCNN collected all data on one client. Furthermore, the smashed data would violate privacy preservation as it allows for inference of the original data. While LSTMSPLIT [26] and MS-SL [9] are oriented towards distributed data to explore the effectiveness of split learning, the complete sequence of an object also remains centralized on one client.

Due to the multi-origin nature of healthcare data, FedSL [27] attempted to explore distributed sequential data. However, their research is limited to the first half of data in a certain client, and the second half in the server. This fixed setup lacks flexibility in real-world applications as healthcare data is often disorganized and cannot be regularly divided into two segments in the same order. Additionally, no single hospital can own the second-half data of all objects like the server. Hence, the above methods are still not applicable for segmented healthcare data learning where each patient visits multiple hospitals.

III. FRAMEWORK AND PROBLEM ANALYSIS

To fill in the technique void for exploring segmented healthcare data (e.g., In Fig. 1, each of the two patients has a visit sequence of 3 different hospitals for his/her diseases with clinical records of 3 segments.), this work presents a novel framework STSL, short for spatio-temporal split learning. Whereas "spatial" means to conduct consecutive learning on the data of different patient groups (i.e., *split in the patient dimension*), each with the same visit sequence, "temporal" means to divide the model into portions according to patients' sequential visits to different hospitals (i.e., *split in the hospital dimension*). This section describes the overall design of STSL and the analysis of its crux problems.

A. Framework Design of STSL

The architecture of our STSL framework is shown in Fig. 2 with an example scenario of n patients visiting m hospitals.

Definition 1. (Sequential data): Sequential data refers to a series of data points arranged in a specific order, typically based on the progression of time. Each data point in the sequence is related to the preceding data point.

Definition 2. (Segmented healthcare data): Each patient u_i has a visit trajectory/sequence $\mathbb{S}_i$ to several hospitals, where the j^{th} hospital in $\mathbb{S}_i$ records the healthcare data d_i^j of patient u_i during his visit to it. We denote data d_i^j as a segmentation of the healthcare data of patient u_i and the set of these pieces of segmentation as $\mathcal{D}_i$.

Segmented healthcare data is short for segmented healthcare sequential data, as it represents a specific type of sequential data, distributed both temporally and spatially. In the example of Fig. 2, the visit sequence of u_1 is $h_1 \rightarrow h_2 \rightarrow h_3$, which renders three pieces of segmentation held in different hospitals, presented as ①, ②, and ③. Given the segmented healthcare data input, STSL intuitively attempts to split the model to be learned into partitions that match the split of healthcare data. This process works in three phases: ordering each patient's data, making a schedule for the training batches, and performing split learning on the set batches consecutively.

1) Patient visit ordering: Building a sequential model requires samples in the correct order, such that the tuned model could predict the future event given the current observed data sequence,

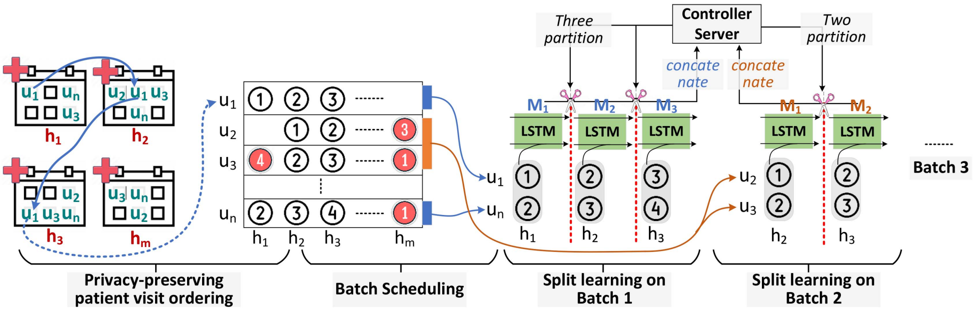


Fig. 2. Architecture of the proposed STSL framework. It takes segmented healthcare records of n patients distributed in m hospitals as input, estimates the visit order of each patient without gathering their data, splits the segmented data horizontally, and partitions the model vertically for batch-based split learning.

e.g., a sentence in NLP. Hence, in the context of this work, the first phase manages to figure out the visit sequence of each patient among all the relevant hospitals. Ideally, this would yield a patient-hospital matrix with each element the order number that a specific patient pays a visit to the corresponding hospital (e.g., element (2,2) is ① as u_2 visits h_2 first during its seek of medical service).

2) Batch scheduling: The visit sequence and the number of hospitals visited by different patients are hardly the same. As a result, it is impossible to perform a one-shot training with one split of the model exactly matching the segmentation distributions of all patient data. For example, if putting the first part of the model in h_1, then the data of u_2 and u_3 would be excluded as h_1 is never visited or not visited as the first by them. As a remedy, in the second phase, STSL assigned patients into K different groups (i.e.,$\mathcal{U}_k$) with ones in the same group sharing the same or similar visit sequence. Particularly, we denote the training schedule of the k^{th} group as batch $\mathbb{B}_k$, which defines the selected segments of each patient in $\mathcal{U}_k$, the corresponding visit sequence $\mathbb{S}_k$ of the left segments and the training order k. For alignment purposes, some segmentation may be discarded to include more data in a batch (e.g., ① and ④ of u_3 in Fig. 2).

3) Split learning on batches: The batch-scheduling result is then used to organize model split and learning. It performs training on each batch with the tuned parameter of the former batch passed to the next one. Specifically, for the k^{th} batch that involves data on $|\mathbb{S}_k|$ hospitals, the controller server partitions the target model $\mathcal{M}$ (w.l.o.g., LSTM in this work) into exactly $|\mathbb{S}_k|$ sub-models and offloads the j^{th} sub-model $\mathcal{M}_k^j$ to the j^{th} hospital in this batch.

Based on the partition, forward propagation is conducted in a way that the 1^{st} hospital in $\mathbb{S}_k$ calculates the outputs (i.e., hidden layer activation) of its sub-model with the batch data it holds and transmits these intermediate results to the 2^{nd} hospital through the network. The 2^{nd} hospital then calculates its sub-model outputs based on the outputs of the former sub-model and its local segmented data. The propagation and activation transmission ends with a prediction at the last hospital in $\mathbb{S}_k$, which estimates a loss between the ground truth and the prediction. Next, the last hospital initials back-propagation with gradient calculation and transmission, optimizing the sub-models in each hospital with gradient descent. For example, in batch $\mathbb{B}_1$ of Fig. 2, the model is partitioned into $\mathcal{M}_1^1$, $\mathcal{M}_1^2$, and $\mathcal{M}_1^3$ and offloaded to hospitals h_1, h_2, and h_3, respectively. Wherein, h_2 tunes $\mathcal{M}_1^2$ with the output of $\mathcal{M}_1^1$, gradient from h_3, and its local data, i.e., ② of u_1 and ③ of u_n.

TABLE II
FREQUENTLY USED NOTATIONS

Notation	Description
$\mathbb{B}_k$	The k^{th} split learning batch in schedule, specifying a group of patients, their selected data, and visit sequence.
$\mathbb{S}_k$	An array of hospitals, whose sequence denotes the visit order of patients in batch $\mathbb{B}_k$.
$\mathcal{M}_k^j$	The j^{th} partition of the target model $\mathcal{M}$ in batch $\mathbb{B}_k$.
$\mathcal{H},\ h_i$	$h_i \in \mathcal{H}$ is a hospital participating in training with local patients' healthcare records.
$\mathcal{U},\ \mathcal{U}_k$	$\mathcal{U}_k \subset \mathcal{U}$ is the group of patients assigned to batch $\mathbb{B}_k$ with the same visit sequence $\mathbb{S}_k$.
$\mathcal{D}_i,\ d_i^j$	$\mathcal{D}_i$ is the healthcare data of $u_i \in \mathcal{U}$ and $d_i^j \in \mathcal{D}_i$ is his/her j^{th} data segment.
$\mathbb{X}_i,\ x_i^j,\ z_i$	$x_i^j \in \mathbb{X}_i$ indicates the involvement of segment d_i^j of patient u_i in its assigned batch with index z_i.
$h_i^{(j)},\ n_i^{(j)}$	$h_i^{(j)}$ represents the hospital that collects the j^{th} data segment of u_i which consists of $n_i^{(j)}$ medical records.

Note: In the healthcare context, users who generate data refer to patients who visit various hospitals, so u_i, instead of the symbol "p", is used to represent a patient and also avoid confusion with the probability.

After several rounds of forward and back-propagation, the controller server will stitch the tuned sub-models of batch $\mathbb{B}_k$ together into a complete model $\mathcal{M}_k$ and partition $\mathcal{M}_k$ into $|\mathbb{S}_{\mathbb{B}_{k+1}}|$ sub-models for the learning of the following batch $\mathbb{B}_{k+1}$. Finally, the controller server releases the concatenated model of the last batch as the joint-learned model, which is believed to absorb knowledge from segmented healthcare data distributed in different hospitals.

B. Key Problems in STSL

Two key problems arise in launching STSL. **Q1:** *How to protect patients' privacy, including both their healthcare records and visit time to each hospital, from leaking to other patients or other hospitals?* This also settles our privacy assumptions, together with an honest-but-curious threat model. **Q2:** *How to construct proper batches considering both efficiency and learning performance?* We analyze concrete privacy protection requirements in **Q1** in Section III-B1 and propose two metrics for analyzing **Q2** in Section III-B2. For ease of understanding, we list some key notations in Table II.

1) Privacy Protection Requirements: The increasing prominence of privacy concerns has become a significant obstacle to data sharing in healthcare, whether to protect patient privacy or maintain competitive advantage. In the context of STSL, three types of entities are involved, i.e., hospitals, patients, and the controller server. The data of patient u_i is in the form of $[u_i, [< t_i^1, d_i^1, h_i^{(1)} >, .., < t_i^j, d_i^j, h_i^{(j)} >, \ldots]]$, including his/her healthcare records d_i^j with start timestamp t_i^j stored in hospital $h_i^{(j)}$, which also indicates his/her visit sequence $\mathbb{S}_i = \{h_i^{(1)}, \ldots, h_i^{(j)}, \ldots\}$.

Threat model: We assume all the entities are honest-but-curious, which means they would perform the learning or scheduling tasks in STSL and may deduce data of others.

Privacy Boundary: Every visit record is sensitive and should not be disclosed to entities other than the patient himself/herself and the hospital that stores it. On one hand, one's name, visit time to each hospital (t_i^j), and healthcare data (d_i^j) in each hospital are private. On the other hand, for sequential data learning, a patient's visit sequence ($\mathbb{S}_i$ inherently telling whether he/she has visited some hospitals) is necessary and believed to be secure to reveal to the controller server. These set the privacy boundary of this work.

STSL uses split learning that trains sub-models on local data, shares only the activation (or gradient) parameters to downstream (or upstream) hospitals in its batch, and uploads sub-model parameters to the controller server. By retaining segmented data in the hospital that generates it, the privacy of patient data is protected. The tricky part remains to attain the visit sequence of each patient without disclosing the specific visit time to each hospital, which will be addressed in Section IV-A.

2) Communication Costs Versus Data Loss: In traditional split learning, the transmission of sub-models among different hospitals incurs communication costs, while STSL's learning in batches aggravates the cost by K time. Intuitively, with fewer batches, like $K = 1$, communications costs would be largely reduced. However, this would inevitably discard more data, leading to a degradation of learning performance. Next, we formulate these two aspects in this part.

Formulation of communication costs: We focus on merely the communication cost instead of computational, as the latter is reduced with fewer training tasks at each hospital in the context of STSL compared to traditional split learning. The communication traffic of STSL can be categorized into three aspects: private communication for patient visit ordering, model communication of offloading and uploading sub-model between the controller server and hospitals, and parameter communication during activation forward-propagation and gradient back-propagation. The first aspect is not impacted by the scheduling process and has a fixed cost, so we focus on the latter two. The volume of model communication is decided by the number of partitions and the multiplex of sub-models between neighbor batches, which could be formulated as:

$$C_M\left(\mathbb{B}_k\right) = \sum_{j=1}^{|\mathbb{S}_k|} 2 \cdot |\mathcal{M}_j^k| \cdot I\left(\mathcal{M}_j^k, \mathcal{M}_j^{k-1}\right), \tag{1}$$

where k is the training order of the batch, $|\mathcal{M}_j^k|$ represents the size of sub-model $\mathcal{M}_j^k$ and $I(\mathcal{M}_j^k, \mathcal{M}_j^{k-1}) \in \{0, 1\}$ indicates whether a hospital receives the same sub-model as the former batch (i.e., $I(\cdot, \cdot) = 0$ if $\mathcal{M}_j^k$ and $\mathcal{M}_j^{k-1}$ are the same and partitioned to the same hospital, otherwise, $I(\cdot, \cdot) = 1$).

Parameter communication is closely related to the number of samples in each batch and the number of segments:

$$C_D\left(\mathbb{B}_k\right) = \left(|\mathbb{S}_k| - 1\right) \cdot |\mathcal{U}_k| \cdot \left(w_f + w_b\right), \tag{2}$$

where w_f and w_b represent the size of activation in forward-propagation and the size of gradients in back-propagation, which are static for a specific model $\mathcal{M}$. $|\mathcal{U}_k|$ and $|\mathbb{S}_k|$ are the number of patients and selected hospitals in batch $\mathbb{B}_k$.

With these, we formally calculate the communication costs of a batch scheduling $\mathbb{B}$ as $C_{total}(\mathbb{B}) = \sum_{k=1}^{K}(C_M(\mathbb{B}_k) + C_D(\mathbb{B}_k))$.

Formulation of data loss: If STSL uses all the healthcare data for training, it may get batches with only one patient, adding up communication costs with limited gain on learning performance. Hence, some data are excluded in constructing batches for efficiency. The controller server prudently assigns $x_i^j \in \{0, 1\}$ to each segment d_i^j of every patient u_i, where $x_i^j = 1$ means the segment is included in the z_i^{th} batch, otherwise excluded. Such discarding of data, on the other hand, degrades learning performance as providing fewer samples and the knowledge of some valuable samples may be ignored.

To form such loss prior to model evaluation, the decreased amount of data is a straightforward measure. Meanwhile, we note that the healthcare data generated during patients' recent visits to a hospital is more valuable than those recorded months or years ago, considering that doctors often take account of recent diagnoses. To mimic such diagnostic behaviors, we introduce a value weight v_i^j to each data segment d_i^j with later data having larger v_i^j. Hence, we have the overall value $o(\mathcal{D}_i)$ and the remaining value $r(\mathcal{D}_i|\mathbb{B})$ of $\mathcal{D}_i$ after scheduling as:

$$o\left(\mathcal{D}_i\right) = \sum_{j=1}^{|\mathcal{D}_i|} \left(|d_i^j| \cdot v_i^j\right), \quad r\left(\mathcal{D}_i|\mathbb{B}\right) = \sum_{j=1}^{|\mathcal{D}_i|} \left(|d_i^j| \cdot x_i^j \cdot v_i^j\right), \tag{3}$$

where $\sum_{j=1}^{|\mathcal{D}_i|} v_i^j = 1$ and v_i^j is a time-dependent weight.

Furthermore, the impact of discarding data segments of a patient is exponentially increased rather than linearly. Considering this, we use a multi-tier loss measuring strategy by setting a penalty factor β_l for each tier of data loss. The higher tier corresponds to larger $o(\mathcal{D}_i) - r(\mathcal{D}_i|\mathbb{B})$ and is set with a larger penalty factor. Therefore, given scheduling that discards data with $x_i^j = 0$, the overall data loss is calculated as:

$$DP\left(\mathcal{D}_i|\mathbb{B}\right) = \begin{cases} \beta_1\left(o\left(\mathcal{D}_i\right) - r\left(\mathcal{D}_i|\mathbb{B}\right)\right), \\ \quad \text{if } r\left(\mathcal{D}_i|\mathbb{B}\right) \geq \eta_1 \cdot o\left(\mathcal{D}_i\right) \\ \beta_1\left(1-\eta_1\right) o\left(\mathcal{D}_i\right) + \beta_2\left(\eta_1 o\left(\mathcal{D}_i\right) - r\left(\mathcal{D}_i|\mathbb{B}\right)\right), \\ \quad \text{if } \eta_1 > r\left(\mathcal{D}_i|\mathbb{B}\right)/o\left(\mathcal{D}_i\right) \geq \eta_2 \\ \quad \cdots \end{cases} \tag{4}$$

where $\eta = [\eta_1, \eta_2, \ldots, \eta_t]$ ($\eta_i \in [0, 1]$, $\eta_i > \eta_j$ if $i < j$) and $\beta = [\beta_1, \beta_2, \ldots, \beta_t]$ ($\beta_i \in \mathbb{R}^+$). By integrating the data loss of all patients, we have $DP(\mathbb{B}) = \sum_{i=1}^{n} DP(\mathcal{D}_i|\mathbb{B})$.

STSL executes split learning batch-by-batch, reducing communication costs by judiciously discarding data. The crucial issue is searching for a balance between communication costs and data loss. We will address this issue in Section IV-B.

IV. Key Components

There is a controller server (hereinafter referred to as the server) in the proposed STSL framework, responsible for privacy-preserving visit ordering and batch scheduling, including *Data Selection* and *Combinatorial Optimization*. Subsequentially, STSL can realize split learning on batches.

A. Privacy-Preserving Visit Ordering

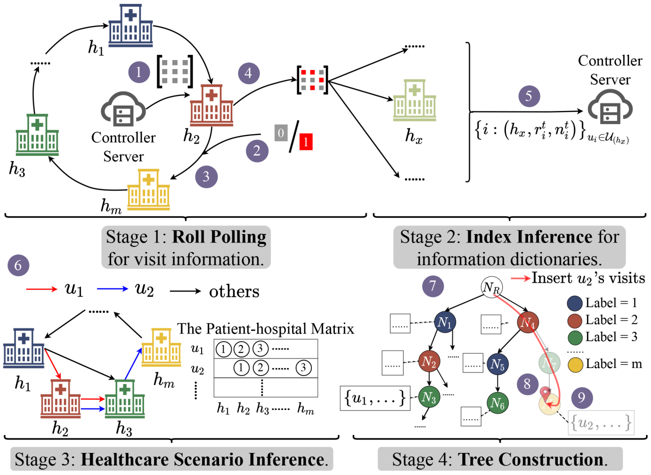
1) Preliminaries: While visit time is privacy, the systematic organization of EHRs can reveal the progression of a disease and assess the effectiveness of a diagnosis [3], [17], [18]. Hence, the first critical challenge is privacy-preserving visit ordering. This involves arranging patient visits in chronological order privately, amounting to a secure multi-party multi-dimensional ordering challenge.

Confidential ordering approaches, Shamir secret sharing [28] and homomorphic encryption [29], cannot be extended to multi-party ordering. [30] examines multi-party scenarios but restricts analysis to single-data comparisons. Obfuscated circuits [31] have also been proposed to securely compute any function in circuit form. However, this method faces limitations due to complex computations. [32] comprehensively addresses scenarios where multiple participants have repeated elements, but their work is focused on one-dimensional data. Actually, visit ordering is often complex, involving multiple parties and dimensions.

2) Calculation Principle: To obtain patients' visit sequences and the associated number of medical records [1] without revealing their specific visit time, privacy-preserving visit ordering is proposed. As illustrated in Fig. 3, this ordering involves the following four stages.

Stage 1: Roll Polling aims to securely collect patient visit information from participating hospitals.

Step 1 The server sets a candidate hospital set $\mathcal{H}^* = \mathcal{H}$ and initialize an all-0 binary matrix $Y_0 = [0]^{|\mathcal{U}| \times T}$ based on the participating patient set $\mathcal{U}$ and the time granularity T [2]. The value at (i, j) of this matrix indicates whether patient u_i visited a hospital at time t_j, with value 1 for positive. During ordering, this matrix is first randomly inverted at a hospital, passed to other hospitals for collecting their visit information in turn, and finally returned to the initial hospital to gain the ordering, thus named a "polling matrix".



h_1
h_2
h_3
h_m
h_x
Controller Server
Controller Server
1
2
3
4
5
0
1
$\{i : (h_x, r_i^t, n_i^t)\}_{u_i \in \mathcal{U}_{(h_x)}}$
Stage 1: **Roll Polling** for visit information.
Stage 2: **Index Inference** for information dictionaries.
6
u_1
u_2
others
The Patient-hospital Matrix
u_1 ① ② ③
u_2 ① ② ③
h_1 h_2 h_3 h_m
Stage 3: **Healthcare Scenario Inference**.
7
8
9
Insert u_2's visits
N_R
N_1
N_2
N_3
N_4
N_5
N_6
N_7
$\{u_1, \ldots\}$
$\{u_2, \ldots\}$
Label = 1
Label = 2
Label = 3
Label = m
Stage 4: **Tree Construction**.

Fig. 3. The flow diagram of privacy-preserving visit ordering.

Step 2 The server randomly selects a hospital $h_a \in \mathcal{H}^*$ and passes the polling matrix Y_0 to h_a. Then h_a is requested to randomly invert the value of $y_{i,t}$ (i.e., let $y_{i,t} = 1$) with a probability for each $i \in [1, |\mathcal{U}|]$ and $t \in [1, T]$ and record corresponding elements in a recorder $R = \{(i_1,\ t_1), (i_2,\ t_2), \ldots\}$.

Step 3 Next, h_a modifies the polling matrix to Y_a, where for each $u_i \notin \mathcal{U}_{(h_a)}$, $y'_{i,t} = y_{i,t}$, and for each $u_i \in \mathcal{U}_{(h_a)}$, $y'_{i,t} = \neg y_{i,t}$, if u_i visit h_a at t ($y'_{i,t} = y_{i,t}$, else). After removing h_a from $\mathcal{H}^*$, the server randomly selects a hospital $h_x \in \mathcal{H}^*$ and informs h_a to pass Y_a to h_x. Then h_x modifies the polling matrix to Y_x according to the regulations above.

Ordering Continuously performs Step 3 until $\mathcal{H}^*$ becomes empty. Eventually, the polling matrix Y^* will encompass the visit time information of all patients.

Step 4 The server informs the last hospital to send Y^* to the first chosen hospital h_a, where some elements in Y^* are inverted again to revert to their original values with the help of the recorder R:

$$y_{i,t} = \begin{cases} \neg y^*_{i,t}, & \text{if } (i,\ t) \in R \\ y^*_{i,t}, & \text{else} \end{cases}. \tag{5}$$

The combination of Step 2 and Step 4 enhances privacy protection and prevents the second hospital from inferring specific information about the first hospital by Y_a. For instance, the server initializes a polling matrix Y_0 and passes it to a randomly selected hospital h_2 in Fig. 3. Subsequently, h_2 randomly sets $y_{1,3} = 1$, $y_{2,2} = 1$ and so on, while recording $R = \{(1, 3), (2, 2), \ldots\}$. Then h_2 modifies Y_2 based on local data and passes it to the next hospital h_m randomly selected by the server. As a result, h_m cannot discern whether an element with a value of 1in the matrix means a patient visit or a random invert nor can they ascertain if the original value of an element with a value of 0 is 1 or not, thus preventing visit inference, thereby ensuring privacy protection.

Stage 2: Index Inference requires each $h_x \in \mathcal{H}$ to infer sequential order for each local patient $u_i \in \mathcal{U}_{(h_x)}$ based on the

[1] A patient visit can last for a while and produce multiple medical records.

[2] For instance, with monthly intervals, $T = 12$ and the start timestamp $t_1^1 = [2023/02/15]$ of the first segment d_1^1 of patient u_1 is regarded as $t_1^1 = 2$. The finer the granularity, the greater the value of T.

polling matrix Y^* and calculate the corresponding numbers of medical records.

Step 5 The server informs h_a to send reverted Y^* to each hospital $h_x \in \mathcal{H}$, then h_x calculates the sequential order r_i^t in corresponding visit time t for patient $u_i \in \mathcal{U}_{(h_x)}$:

$$r_i^t = \sum_{k=1}^{k=t} y_{i,k}^*,\ \left(y_{i,t}^* = 1\right). \tag{6}$$

Next, h_x calculates the number of medical records n_i^t for patient u_i during period t and constructs an information item $\{i : (h_x, r_i^t, n_i^t)\}$. For example, hospital h_2 constructs an information item $\{1 : (h_2, 2, 18)\}$, meaning that patient u_1's 2^{nd} visit happened in hospital h_2 (i.e., $h_1^{(2)} = h_2$) and generated 18 records (i.e., $n_1^{(2)} = 18$). Finally, h_x sends its information dictionary $\{i : (h_x, r_i^t, n_i^t)\}_{u_i \in \mathcal{U}_{(h_x)}}$ to server.

Stage 3: Healthcare Scenario Inference is primarily concerned with deducing the complete healthcare scenario, leveraging the information dictionaries provided by $\mathcal{H}$.

Step 6 For each $u_i \in \mathcal{U}$, the server sorts the corresponding information items $\{i : (h_x, r_i^t, n_i^t)\}_{h_x \in \mathcal{H}}$ received from $\mathcal{H}$ based on the value of r_i^t, ordering s healthcare segments as $[(h_i^{(1)}, n_i^{(1)}), (h_i^{(2)}, n_i^{(2)}), \ldots, (h_i^{(s)}, n_i^{(s)})]$. Consequently, the visit sequence $\mathbb{S}(u_i; \mathbb{X}_i) = [h_i^{(1)}, h_i^{(2)}, \ldots, h_i^{(s)}]$ and the corresponding numbers of medical records $[n_i^{(1)},\ n_i^{(2)}, \ldots, n_i^{(s)}]$ can be inferred. Taking u_1 as an example, the server sorts information items of u_1 as $\{1 : (h_1, 1, 9), 1 : (h_2, 2, 18), 1 : (h_3, 3, 6)\}$, orders his/her segments as $[(h_1, 9),\ (h2, 18), (h_3, 6)]$, and finally deduces his/her visit sequence $\mathbb{S}(u_1; \mathbb{X}_1) = [h_1,\ h_2,\ h_3]$ and corresponding numbers of medical records $[9, 18, 6]$. Ultimately, the patient-hospital matrix can be deduced, where each element represents the sequential order of a specific patient visiting the corresponding hospital.

It is noteworthy that if patient u_i has consecutive sequential orders within h_x, the server will retain the earliest one and aggregate the medical records of corresponding visits as those of the consolidated instance for u_i. Additionally, the subsequent sequential orders will be decremented accordingly. What's more, if more than one hospitals have the same sequential order for a patient, the server can order them randomly or repeat preceding steps with bigger T for these hospitals to get precise sequential orders.

Stage 4: Tree Construction involves the conversion of the healthcare scenario into a healthcare tree.

As illustrated in Fig. 3, each non-root node (intermediate and leaf nodes, denoted as N_i) of the healthcare tree is labeled with the ID of a specific hospital. Its attributes comprise a patient information table storing details of certain patients, encompassing their IDs and numbers of medical records. Each node can have multiple child nodes which signify patients referred from the parent hospital to the respective child hospital. The retrieval path from the root node to a child node delineates the visit sequence of particular patients. Furthermore, the patient details are archived in the patient information table of the terminal node in its retrieval path.

For instance, the label of N_4 is 2, signifying h_2. The branch $N_R \rightarrow N_1 \rightarrow N_2 \rightarrow N_3$ signifies the visit sequence $h_1 \rightarrow h_2 \rightarrow h_3$, where node N_3 records details of all patients with $\mathbb{S} = [h_1, h_2, h_3]$ (e.g., u_1) in its patient information table. The construction of a healthcare tree is delineated as follows.

Step 7 The server initializes a healthcare tree with a root node.

The server repeats Step 8 and Step 9 for each patient $u_i \in \mathcal{U}$:

Step 8 The server checks whether the branch corresponding to u_i's visit sequence $\mathbb{S}_i = [h_i^{(1)}, h_i^{(2)}, \ldots, h_i^{(s)}]$ exists. If not, it constructs the corresponding branch, i.e., constructs $N_R \rightarrow N(h_i^{(1)}) \rightarrow N(h_i^{(2)}) \rightarrow \ldots \rightarrow N(h_i^{(s)})$. For instance, in Fig. 3, the red line denotes that upon inserting the visits of u_2, the branch $N_R \rightarrow N(h_2) \rightarrow N(h_3) \rightarrow N(h_m)$ does not initially exist. As a result, the server branches from N_4 to create nodes N_7 and N_8, labeled as 3 and m, respectively.

Step 9 The server identifies the terminal node in u_i's retrieval path and stores his/her information in its patient information table. For instance, to insert the visits of u_2, the server locates the last node $N(h_m)$ of $N_R \rightarrow N(h_2) \rightarrow N(h_3) \rightarrow N(h_m)$ (i.e., N_8), and then records u_2's information in N_8's patient information table.

The original healthcare scenario possesses complex graph-theoretic properties, posing non-negligible challenges in terms of scheduling complexity and retrieval efficiency. Hence, the graph-based healthcare scenario is further transformed into a healthcare tree. Ultimately, the server accomplishes the ordering task while keeping sensitive visit time private.

In the processing of *Ordering*, the server has meticulously initialized a polling matrix to systematically collect visit information from each hospital $h_x \in \mathcal{H}$. It is crucial to note that once the polling matrix is disseminated, the server no longer has access to it. This intentional measure serves as a safeguard to prevent the server from obtaining any additional inferences beyond the originally obtained information. Such an approach minimizes the potential for biased or intrusive analysis by the server to ensure privacy preservation.

B. Batch Scheduling for Penalty-Aware Split Learning

1) Problem Formulation: Penalty Function: To yield a tunable balance between communication costs and data loss, we use a weighted summation of them to construct the final penalty function $penalty(\mathbb{B})$. It indicates the overall overhead of a specific batch scheduling.

$$penalty\left(\mathbb{B}\right) = \alpha \cdot DP\left(\mathbb{B}\right) + (1-\alpha) \cdot C_{total}\left(\mathbb{B}\right), \tag{7}$$

where α adjusts the impact of C_{total} and DP , and is set to be 0.5w.l.o.g. If one is less tolerant of missing data (or excessive communication costs) during the training process, it can increase (or decrease) the value of α.

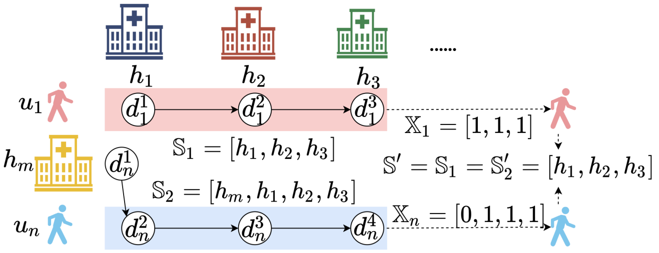

h_1
h_2
h_3
......
u_1
d_1^1
d_1^2
d_1^3
$\mathbb{X}_1 = [1, 1, 1]$
$\mathbb{S}_1 = [h_1, h_2, h_3]$
h_m
d_n^1
$\mathbb{S}' = \mathbb{S}_1 = \mathbb{S}_2' = [h_1, h_2, h_3]$
$\mathbb{S}_2 = [h_m, h_1, h_2, h_3]$
u_n
d_n^2
d_n^3
d_n^4
$\mathbb{X}_n = [0, 1, 1, 1]$

Fig. 4. The flow diagram of *Data Selection*, where u_1 and u_n have visited certain hospitals in the same relative order.

Optimization problem of penalty-aware scheduling: Based on the above analysis, the calculation of batch scheduling is transformed into an optimization problem that finds a proper schedule (controlled by the indicator x_i^j) that minimizes the overall penalty.

$$\min \quad penalty\,(\mathbb{B}) \tag{8}$$

$$\text{s.t.} \quad \mathcal{U}_{k_1} \cap \mathcal{U}_{k_2} = \oslash \;\; (k_1 \neq k_2) \tag{9}$$

$$u_i \in \mathcal{U}_{z_i} \subset \mathcal{U} \tag{10}$$

$$\forall u_i \in \mathcal{U}_k, \; \textstyle\sum_j x_i^j = |\mathbb{S}_k|, \; \mathbb{S}(u_i; x_i) = \mathbb{S}_k \tag{11}$$

$$\nexists u_i \in \mathcal{U}, \; (\mathbb{S}(u_i; x_i) = \mathbb{S}_k) \; \wedge \; (u_i \notin \mathcal{U}_k) \tag{12}$$

$$var \quad x_i^j \in \{0, 1\}\,, z_i \in [1, K]. \tag{13}$$

Wherein, $\mathbb{S}(u_i; \mathbb{X}_i)$ is the selected hospital sequence of u_i, i.e., if $x_i^j = 1$ (or $= 0$) then $h_i^{(j)} \in \mathbb{S}(u_i; \mathbb{X}_i)$ (or $\notin \mathbb{S}(u_i; \mathbb{X}_i)$). Here, $h_i^{(j)}$ represents the hospital that records the j^{th} data segment of u_i. Specifically, constraint in (9) states that the healthcare data of one patient is only assigned to at most one batch. Constraint in (10) denotes the index of the batch that u_i is assigned to. Constraint in (11) requires that the visit sequence of all the patients in the same batch $\mathbb{B}_k$ is the same as $\mathbb{S}_k$. Constraint in (12) guarantees the completeness of the scheduling by adding all the patients that have the same visit sequence after data discarding to the same batch.

Solution analysis: Communication efficiency is crucial for accelerating distributed deep neural network [33]. However, due to the significant variability of patient visit sequences, training a model using unprocessed data would result in frequent transitions among clients, introducing significant communication costs. Fortunately, there are similarities in visit sequences. For example, patients from different prefecture-level cities may be referred to the same higher-level hospital after visiting different hospitals in their respective towns. Alternatively, they may initially visit the same hospital for treatment but later be redirected to different hospitals for further diagnosis and treatment due to varying causes of illness. To fully utilize the similarities in visit sequences to minimize penalty, the server performs three essential components: *Data Selection*, *Combinatorial Optimization*, and *Model Partitioning*.

2) Data Selection. Basic Idea: As shown in Fig. 4, the core idea of *Data Selection* is to sacrifice some medical records of certain patients to merge their visit sequences, with the ultimate goal of reducing communication costs while ensuring adequate utility. This problem falls in a binary programming problem concerning $\mathbb{X}$ to minimize $penalty(\mathbb{B})$.

In the context of original healthcare scenario, a patient $u_i \in \mathcal{U}$ with a visit sequence denoted as $\mathbb{S}(u_i; \mathbb{X}_i) = [h_i^{(1)}, h_i^{(2)}, \ldots, h_i^{(|\sim|)}]$ is associated with $\mathbb{X}_i = \{1\}^{|\sim|}$. It is crucial to emphasize that $|\mathbb{S}|$ manifests variations in both content and length among different patients. The binary programming problem concerning $\mathbb{X}$ entails the assignment of $x_i^k = 0$, indicating the exclusion of the k^{th} segment d_i^k for u_i, thereby resulting in the elimination of $h_i^{(k)}$ from $\mathbb{S}(u_i; \mathbb{X}_i)$. As illustrated in Fig. 4, $x_n^1 = 0$ signifies that the first segment of u_n has been omitted and the removal of $h_n^{(1)} = h_m$ from $\mathbb{S}(u_n; \mathbb{X}_n)$. By employing appropriate 01 programming techniques to handle $\mathbb{X}_1$ and $\mathbb{X}_n$, the equivalence $\mathbb{S}(u_1; \mathbb{X}_1) = \mathbb{S}(u_n; \mathbb{X}_n)$ can be achieved, whereby u_n discards a portion of his/her healthcare data. Consequently, the original training batches, namely $\mathbb{B}_1 = \{\mathbb{S}_1, \{u_1\}\}$, and $\mathbb{B}_2 = \{\mathbb{S}_2, \{u_n\}\}$, can be consolidated into a unified training batch denoted as $\mathbb{B}' = \{\mathbb{S}', \{u_1, u_2\}\}$. It is noteworthy that integrating their training paths effectively reduces the model communication ($C_{\mathcal{M}}$), and judiciously discarding data contributes to curtailing parameter communication ($C_{\mathcal{D}}$). The efficacy of this programming endeavor is ascertained when the metric $penalty(\mathbb{B})$ exhibits a reduction, i.e., $penalty(\mathbb{B}') < penalty(\mathbb{B}_1) + penalty(\mathbb{B}_2)$. The core objective of *Data Selection* lies in assigning values to $\mathbb{X} = \{\mathbb{X}_1, \mathbb{X}_2, \ldots, \mathbb{X}_{|\mathcal{U}|}\}$, thereby minimizing the overall $penalty(\mathbb{B})$.

Algorithm Design: Data Selection can be formulated as a capacity vehicle routing problem (CVRP) [34] with relaxed constraints, allowing passengers to disembark en route and disregarding issues of overloading. It is noteworthy that CVRP has been proven to be an NP-hard problem [35], and this extended version of CVRP is also NP-hard. Therefore, *Data Selection* proposes employing a greedy strategy to obtain an approximate solution.

Given two batches, $\mathbb{B}_1 = \{\mathbb{S}_1, \mathcal{U}_1\}$ and $\mathbb{B}_2 = \{\mathbb{S}_2, \mathcal{U}_2\}$, the consolidation attempt of *Data Selection* is as follows:

Step 1 A longest common subsequence, denoted as $\mathbb{S}'$, is computed for visit sequences $\mathbb{S}_1$ and $\mathbb{S}_2$.

Step 2 Each $\mathbb{X}_i$ for $u_i \in \mathcal{U}_1 \bigcup \mathcal{U}_2$ is calculated to realize corresponding visit sequence $\mathbb{S}(u_i; \mathbb{X}_i) = \mathbb{S}'$.

Step 3 The change of $penalty(\mathbb{B})$ after and before consolidation is computed:

$$\triangle\, penalty = penalty\,(\mathbb{B}') - penalty\,(\mathbb{B}_1) - penalty\,(\mathbb{B}_2)\,. \tag{14}$$

Data Selection attempts to consolidate each batch with other batches separately and finally selects the consolidation with the maximum decrease in *penalty* (i.e., Min $\triangle penalty$ and $\triangle penalty < 0$). Meanwhile, the healthcare tree would be modified accordingly by deleting or adding nodes. If no consolidation decreases *penalty* (i.e., $\forall \mathbb{B}_i, \mathbb{B}_j \in \mathbb{B}, \triangle penalty \geq 0$), the batch remains unchanged. *Data selection* repeats this process until all batches are stabilized.

3) Combinatorial Optimization: Basic Idea: As Shown in Fig. 5, the core concept of *Combinatorial Optimization* is to adjust the training order of batches to maximize utilization of common parts in visit sequences so that model communication ($C_{\mathcal{M}}$) can be minimized. This part corresponds to assignment

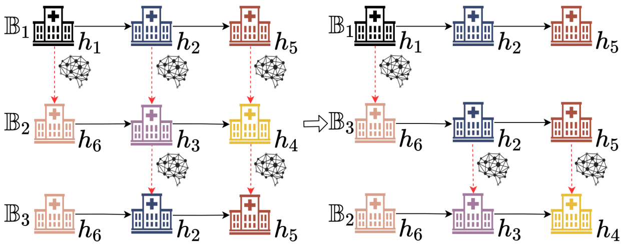


Fig. 5. The flow diagram of *Combinatorial Optimization*.

of $z_i(u_i \in \mathcal{U})$ in (13), which is transformed to pairing $\mathbb{B}_k = \mathbb{B}_{(j)}$ $(k, j \in [1, K])$. Moreover, *Combinatorial Optimization* does not modify the value of $\mathbb{X}$, so it effectively reduces C_{total} without introducing any additional DP. This problem falls in searching for the optimal order of $\mathbb{B}$ to minimize $penalty(\mathbb{B})$.

The initial training order is determined by a depth-first search on the processed healthcare tree. This order only uses the similarities found in the prefix subsequence, without considering the shared characteristics in the middle and suffix subsequences. As illustrated in Fig. 5, adhering to the original training order $\mathbb{B} = [\mathbb{B}_1, \mathbb{B}_2, \mathbb{B}_3]$ necessitates five iterations of model transmission. However, by rearranging the order to $\mathbb{B}' = [\mathbb{B}_1, \mathbb{B}_3, \mathbb{B}_2]$, both prefix and suffix commonalities can be simultaneously exploited, resulting in a reduction to only three iterations of model transmission. Consequently, this modification leads to a decrease in the amount of model communication required. The primary objective of *Combinatorial Optimization* lies in generating an optimized order $\mathbb{B} = [\mathbb{B}_{(1)}, \mathbb{B}_{(2)}, \ldots, \mathbb{B}_{(|\mathbb{B}|)}]$ tailored specifically to the processed healthcare tree, thereby minimizing the overall $penalty(\mathbb{B})$.

Algorithm Design: As the size of $\mathbb{B}$ increases, the solution space for *Combinatorial Optimization* grows exponentially, rendering the task of finding the optimal order that minimizes the $penalty(\mathbb{B})$ an NP-hard problem [36]. To address this challenge, *Combinatorial Optimization* adopts a parallel random-start hill-climbing algorithm to approximate a solution. The algorithm follows the following steps:

Step 1 Randomly select an initial node $\mathbb{B}_i = \{\mathbb{S}_i, \mathcal{U}_i\}$, and set candidate batches $\mathbb{B}^* = \mathbb{B} - \mathbb{B}_i$.

Step 2 Identify a batch $\mathbb{B}_j = \{\mathbb{S}_j, \mathcal{U}_j\} \in \mathbb{B}^*$ as a successor to minimize the distance between $\mathbb{S}_i$ and $\mathbb{S}_j$, and then delete $\mathbb{B}_j$ from $\mathbb{B}^*$. The distance is defined as:

$$D\left(\mathbb{S}_i, \mathbb{S}_j\right) = \sum_{k=1}^{k=|\mathbb{S}_i|} I(\mathcal{M}_i^k, \mathcal{M}_j^k)\left(|\mathcal{M}_i^k| + |\mathcal{M}_j^k|\right) + \sum_{k=|\mathbb{S}_i|+1}^{k=|\mathbb{S}_j|} |\mathcal{M}_j^k|, \tag{15}$$

where $I(\mathcal{M}_i^k, \mathcal{M}_j^k) = 0$, if $\mathbb{S}_i^k = \mathbb{S}_j^k$ and $\mathcal{M}_i^k = \mathcal{M}_j^k$, else 1, assuming with generality that $|\mathbb{S}_i| \leq |\mathbb{S}_j|$ and vice versa.

Step 3 Repeat Step 2 until $\mathbb{B}^*$ is empty.

To mitigate the risk of falling into local optima, *Combinatorial Optimization* repeats these steps for k times and selects the solution with the minimized $penalty(\mathbb{B})$ as the final order.

C. Split Learning on Batches

The third crucial phase for STSL is to flexibly partition a global model into multiple segments and guide the hospitals to train a model collaboratively.

1) Model Partitioning: For each batch $\mathbb{B}_k = \{\mathbb{S}_k, \mathcal{U}_k\}$, the relevant hospitals are required to train a M-layer global model $\mathcal{M} = [w_1,\ w_2,\ \ldots,\ w_M]$ (w.l.o.g., each LSTM unit is a layer for the LSTM model in this work) collaboratively. The server splits the global model $\mathcal{M}$ into multiple segments $\mathcal{M}_k^1, \mathcal{M}_k^2, \ldots, \mathcal{M}_k^{|\mathbb{S}_k|}$ and allocate $\mathcal{M}_k^i$ to corresponding hospital $h_k^{(i)} \in \mathbb{S}_k$. Specifically, when $|\mathbb{S}_k| \leq M$, the server sequentially divides the model into $|\mathbb{S}_k| - 1$ segments with $M // |\mathbb{S}_k|$ layers and assigns the remaining to $\mathcal{M}_k^{|\mathbb{S}_k|}$, as more recent patient visits deserving of more layers. In the case where $|\mathbb{S}_k| > M$, the server rotationally distributes M layers to the first $|\mathbb{S}_k| - 1$ hospitals and assigns the remaining layers in the recent rotation to $\mathcal{M}_k^{|\mathbb{S}_k|}$.

2) Model Training: Under the scheduling of the server, $h_k^{(1)}$ is required to calculate the outputs (i.e., hidden layer activation) of its sub-model $\mathcal{M}_k^1$ with the batch data it holds and transmits these intermediate results to $h_k^{(2)}$. Similarly, $h_k^{(2)}$ calculates outputs of $\mathcal{M}_k^2$ based on the outputs of the former sub-model and its local segmented data. The propagation and activation transmission ends with a prediction at the last hospital $h_k^{(|\mathcal{S}_k|)}$, which estimates a loss between the ground truth and the prediction. Specifically, in this work, STSL focuses on a binary classification and employs binary cross-entropy loss as shown in (16). Next, the last hospital initials back-propagation with gradient calculation and transmission, optimizing the sub-models in each hospital with gradient descent.

$$Loss = -[y \cdot \log(p) + (1 - y) \cdot \log(1 - p)], \tag{16}$$

where y denotes the true label of one record and p represents the predicted probability, i.e., sigmoid output.

After several rounds of forward and back-propagation, the server will stitch the tuned sub-models of batch $\mathbb{B}_k$ together into a complete model $\mathcal{M}_k$ as the updated global model. Concretely, sub-models that are not reassigned are updated directly, while those that are reassigned have their weights w_i updated using scale-based federated averaging, ensuring that information from each hospital is incorporated. Subsequently, the server partitions $\mathcal{M}_k$ into $|\mathbb{S}_{\mathbb{B}_{k+1}}|$ sub-models for the learning of the following batch $\mathbb{B}_{k+1}$. Finally, after processing all batches, the server releases the concatenated model as the joint-learned model, which is expected to integrate knowledge from the distributed healthcare data across hospitals.

V. Formal Proof of Privacy Preservation

STSL guarantees privacy preservation throughout the framework, embodied in visit ordering before training and batch training process. The privacy preservation is grounded on the honest-but-curious assumption, which means each entity would perform the ordering or scheduling tasks in STSL honestly but may deduce the patient data of others. This section analyzes the uncertainties of both the occurrence of a patient's visit at a specific time and the number of participating hospitals to prove the privacy of visit ordering and narrates the privacy property of batch learning equivalent to that of split learning to show the security of the training process.

A. Visit Ordering

If there is only one hospital, it equals centralized learning without interactions with other hospitals. The security naturally holds. As for two or more participating hospitals, the first chosen hospital h_a receives an all-0 polling matrix, revealing nothing. Hence, we only need to prove that non-starting hospitals couldn't deduce sensitive information from their received polling matrix.

Assume without loss of generality that there are n patients having visited s hospitals, randomly chosen from m hospitals, respectively. The polling matrix is with time granularity T and the first chosen hospital randomly sets $y^0_{i,t} = 1$ with a probability p (Step 2 in Roll Polling). The polling matrix Y^* is accepted by hospital h_j after x hospitals have modified it, and the original matrix without disturbance is Y (i.e., $p = 0$).

Theorem 1: The proposed STSL ensures that the occurrence of a patient's visit at a specific time is uncertain, under the assumption $p = 0.5$. For any hospital h_j, it is impossible to deduce whether a patient u_i has visited previous hospitals in period t (i.e., h_j couldn't infer the value of $y_{i,t}$ from $y^*_{i,t}$ with additional certainty).

Proof: The probability of $y_{i,t} = 1$ is denoted as q (i.e., $P(y_{i,t} = 1) = q$). Then the value of $y^*_{i,t}$ is determined as:

$y^*_{i,t}$ $y_{i,t}$ $y^0_{i,t}$	0 $(P = 1 - q)$	1 $(P = q)$
0 $(P = 1 - p)$	0	1
1 $(P = p)$	1	0

Hospital h_j deduces $y_{i,t} = 1$ from $y^*_{i,t} = 1$ with probability:

$$P\left(y_{i,t} = 1 | y^*_{i,t} = 1\right) = \frac{q\,(1-p)}{q\,(1-p) + p\,(1-q)}. \tag{17}$$

Simply, h_j deduces $y_{i,t} = 1$ from $y^*_{i,t} = 0$ with probability:

$$P\left(y_{i,t} = 1 | y^*_{i,t} = 0\right) = \frac{pq}{pq + (1-p)\,(1-q)}. \tag{18}$$

Obviously, if $P(y_{i,t} = 1 | y^*_{i,t} = 1) = P(y_{i,t} = 1 | y^*_{i,t} = 0)$, hospital h_j couldn't get additional information about $y_{i,t}$ from $y^*_{i,t}$. According to (17) and (18), if $p = 0.5$, the equation is satisfied for any x.

Theorem 2: The proposed STSL guarantees that the number of participating hospitals is uncertain, under the assumption $p = 0.5$. For any hospital h_j, it is impossible to derive how many hospitals have modified the polling matrix(i.e., h_j couldn't deduce the value of x from Y^*).

Proof: In case $p = 0$ and there are n patients having visited s hospitals, randomly chosen from m hospitals, respectively, so there will be nsx/m elements with values of 1in Y. With disturbance added, the number of elements $N(x)$ with values of 1in Y^* will change to:

$$N(x) = \frac{nsx}{m}\,(1-p) + \left(nT - \frac{nsx}{m}\right)p. \tag{19}$$

If $N(x)$ is probabilistically independent of x, we can conclude that h_j couldn't derive the value of x from Y. Solving the linear algebraic equation $N(x) = N(x+1)$, we can get $p = 0.5$.

Given the above theorems, we can conclude that any hospital can't infer additional information during Roll Polling. What's more, once the polling matrix is disseminated, the server no longer has access to it. This intentional measure serves as a safeguard to prevent the server from obtaining any additional inferences. Taking into account that only the sequential orders are processed during the subsequent stages, visit ordering is affirmed to be secure. Indeed, privacy-preserving visit ordering design possesses generosity to the problem of secure computation of multi-dimensional data among multiple parties.

B. Batch Training

The batch training mechanism inherits the fundamental security properties of traditional split learning. During the training process, each group of hospitals processes selected data through their local models, strictly adhering to the privacy-preserving protocol established in conventional split learning. More precisely, the security of batch training is rooted in the principle that only smashed data, comprising activations and gradients, are exchanged between the clients (i.e., hospitals) and the server, rather than raw data or model parameters. This design creates an effective information barrier, as transforming raw data into activations through non-linear operations makes it computationally infeasible to reconstruct the original input. Similarly, the exchange of gradients, rather than full model parameters, provides additional protection for model security.

By maintaining these properties, our framework ensures input data confidentiality, as patient data never leaves the local environment, while also protecting model parameters and intermediate data from exposure. These security characteristics align with the rigorous privacy standards of traditional split learning, making the framework suitable for healthcare applications where data confidentiality is of paramount importance.

C. Security Boundary

While the preceding subsections establish the security of the proposed STSL framework under the honest-but-curious assumption, this security guarantee is contingent upon the absence of collusion among participating clients. The framework becomes vulnerable when multiple clients conspire to share intermediate information, potentially compromising both the confidentiality of patient visit times and the model integrity.

The security vulnerabilities manifest in two primary aspects: First, through the roll polling in visit ordering, if a hospital obtains the polling matrix from the predecessor of its preceding hospital, it can systematically deduce the specific time intervals during which patients visit the preceding hospital. This inference process can be recursively applied to compromise the temporal privacy of all participating hospitals. Second, the collaborative sharing of sub-models among hospitals in one batch would directly violate the model's security, potentially exposing sensitive parameters and learned patterns.

To address these security concerns in more stringent application scenarios, several advanced cryptographic techniques could be incorporated into the framework. For visit ordering protection, homomorphic encryption presents a viable solution to prevent collusion-based inference attacks. Regarding split learning enhancements, two promising directions emerge from existing literature: (1) the incorporation of carefully calibrated noise into smashed data, as demonstrated in [37], and (2) the implementation of secure aggregation protocols, as proposed in [38], to mitigate potential data leakage risks. These advanced security measures, while theoretically promising for enhancing the framework's robustness, extend beyond the current study's scope and are constrained by practical limitations, particularly the page restrictions of this publication. Their comprehensive exploration and implementation remain valuable directions for future research endeavors.

VI. Experimental Evaluation

In this section, we carry out a comprehensive evaluation of STSL[3] based on the following three questions.

- *Exp#1:* How does the STSL-trained model perform?
- *Exp#2:* How much do optimizations affect scheduling?
- *Exp#3:* How much do penalty settings affect scheduling?

A. Settings

Dataset: Due to the limited availability of real spatio-temporal distributed medical datasets, we construct our dataset on the publicly available and highly representative eICU database [15]. This multi-center intensive care unit database includes vital signs, care plans, diagnostic information, and more for 200,859 admissions from 139,367 patients across 208 hospitals in the United States, covering the period from 2014 to 2015. First, we filter 10,070 patients from *eICU* who have multiple inter-hospital visits and extract the 24-hour records from their most recent and second-to-last ICU stays. We then select 13 numerical variables, scaled between -1 and 1, and 7 categorical variables to create a feature vector for each record. Consequently, our dataset consists of 10,070 sequential samples, each comprising 48 records, with each record represented by a 20-element feature vector.

Additionally, the proposed STSL has not been specifically optimized for this dataset, which enhances its scalability to other scenarios and datasets.

Healthcare Scenarios: We utilize the processed *eICU* to synthesize healthcare scenarios with varying configurations. For each healthcare scenario, we assume that there are m hospitals participating in training, with each patient's s segments randomly distributed among s out of the m hospitals. The number of each segment's medical records is also randomized. Fig. 6 visualizes healthcare data distributions for a scenario with $m = 4$ and $s = 3$ under different operations. The original healthcare data distribution is irregular due to varied patient visit sequences. After applying privacy-preserving visit ordering, patients with identical visit sequences are grouped, but the distribution remains scattered. Data selection discards some sequences to concentrate the distribution, while combinatorial optimization refines the sequence order for improved coherence.

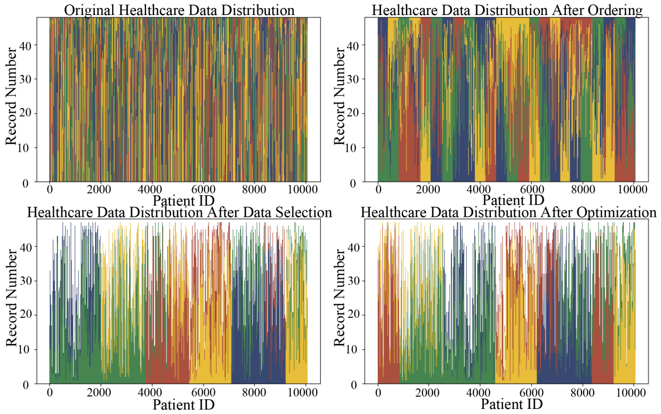


Fig. 6. Visualization of healthcare data distributions for a scenario with $m = 4$ and $s = 3$ under different operations. Each line represents a piece of patient information. The color and length of the lines reveal the hospitals where the segments are stored and the numbers of their medical records.

Training Task and Metrics: We consider the task of predicting patient discharge status, which is a binary classification task where 0 or 1 represents the survival or death of the patient after ICU stays. The discharge status of each patient's last admission is used as the label. Among the 10,070 samples, 1,019 (10.12%) are labeled as "death" (i.e., 1), while 9,051 (89.88%) are labeled as "survival" (i.e., 0).

There are three metrics to evaluate model performance:

- *Accuracy:* Prediction accuracy of discharge status.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}, \tag{20}$$

  wherein TP, FP, TN, and FN denote the proportions of "death" predicted as "death", "survival" predicted as "death", "survival" predicted as "survival", and "death" predicted as "survival", respectively.

- *Precision:* Prediction accuracy of positive samples.

$$Precision = \frac{TP}{TP + FP}. \tag{21}$$

- *Recall:* Prediction coverage of positive samples.

$$Recall = \frac{TP}{TP + FN}. \tag{22}$$

- *F1:* $F1$ score, which is the harmonic mean of precision and recall, providing a balance between the two metrics.

$$F1 = \frac{2 \cdot Precision \cdot Recall}{Precision + recall}. \tag{23}$$

There are three metrics to evaluate optimization impact:

- *Accuracy:* Prediction accuracy of discharge status.
- *CV:* Communication volume for each epoch.

[3] All the scripts could be found in Gitee: https://gitee.com/ahaBCD/STSL.

- DR: Data retention ratio, the ratio of participated data volume to original data volume.

$$DR(\mathcal{D}) = \sum_{\mathcal{D}_i \in \mathcal{D}} \frac{\sum_{j=1}^{j=|\mathcal{D}_i|} \left(|d_i^j| \cdot x_i^j \right)}{\sum_{j=1}^{j=|\mathcal{D}_i|} \left(|d_i^j| \right)}. \tag{24}$$

As for the evaluation of penalty settings impact, we focus on CV and DR.

Model: We implement a model consisting of 3 basic LSTM units for all frameworks for the sake of fairness. Additionally, STSL focuses on the construction of a new framework suitable for distributed segmented data and does not care about the problem of model tuning [39]. Future research can further adjust the training model to obtain better performance.

Baselines: As discussed above, existing methods don't consider segmented healthcare data learning as they inherently assume that the complete sequential data of a patient is stored in one institute. Due to the absence of a suitable state-of-the-art, we compare the model performance of the first explored segmented data learning framework(STSL), with conventional SL and FL (FedAvg [20]). Meanwhile, we test three variants (STSL (w/o S), STSL (w/o D) and STSL (w/o C)) to illustrate the effectiveness of each optimization module.

- *SL*: In conventional SL, the model is statically divided into two parts, with the first LSTM unit assigned to a client and the remaining two LSTM units allocated to the server. Within each healthcare scenario, n hospitals with a server engage in the training process. During each round of global training, each client sequentially collaborates with the server to train the model, passing its client model to the next client in line. In such a framework, healthcare segments relating to the same patient across different hospitals cannot be amalgamated, and the server-side model only inputs the hidden state sequence without additional input.
- *FL*: In FedAvg, n hospitals with a server participate in the training process. In each global training round, each client h_i downloads the global model, trains it on local data, and then sends the model updates to the server, which performs federated averaging to update the global model. Similarly, healthcare data from the same patient across different hospitals cannot be combined.
- *STSL (w/o S):* A framework similar to STSL without batch scheduling (i.e., *Data Selection* and *Combinatorial Optimization*).
- *STSL (w/o D):* A framework similar to STSL without *Data Selection*.
- *STSL (w/o C):* A framework similar to STSL without *Combinatorial Optimization*.

We utilize Python to implement the above frameworks and conduct the following evaluations on a computer with NVIDIA GeForce RTX 3070 for five healthcare scenarios.

B. Results

(Exp#1) Different Learning Frameworks: The experimental results illustrated in Fig. 7 reveal the superior performance of STSL over both FL and SL. Notably, STSL demonstrates significant performance enhancements, achieving over 10% improvement in $Accuracy$, more than 21% increase in $Precision$ across various scenarios, and consistently surpassing SL by over 22% in $F1$ across all scenarios.

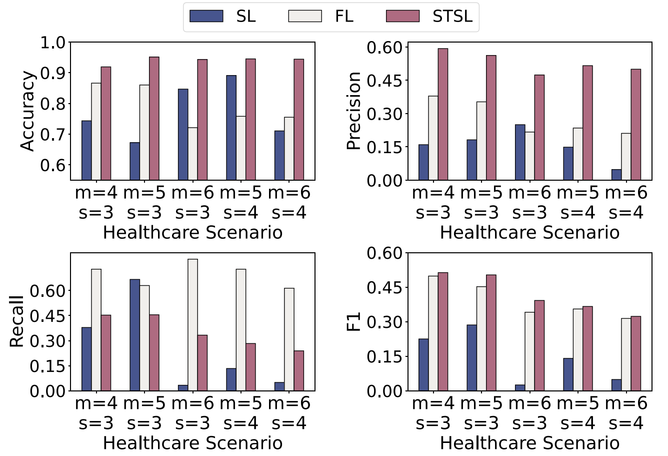


Fig. 7. Comparison of learning frameworks across varying hospital counts (m) and data segments (s).

While FL demonstrates competitive performance in specific metrics, particularly achieving comparable $F1$ scores to STSL and even superior $Recall$ values, a substantial compromise in $Accuracy$ offsets this advantage. This phenomenon can be attributed to FL's tendency to produce biased predictions, frequently classifying samples as positive. In extreme cases, FL generates nearly equiprobable predictions for positive and negative classes, rendering its outputs practically meaningless.

The superior performance of STSL can be primarily attributed to its effective utilization of both temporal and spatial data characteristics through inter-hospital consultation mechanisms, thereby enhancing predictive accuracy. In contrast, FL's limitation to local hospital data and SL's restriction to client-provided smashed data significantly constrain their ability to form comprehensive training sets, particularly given the inherent sparsity of patient visit records. These limitations are especially pronounced in SL, where the server's capacity for data integration is fundamentally restricted.

However, our findings also highlight the sensitivity of model performance to scenario complexity, the number of hospitals (m) and data segments (s), within the STSL framework, suggesting the necessity for extensive model-tuning experiments to maintain optimal performance. While this aspect represents a valuable direction for future research, it extends beyond the current scope of our investigation.

(Exp#2) Ablation results: In experimental settings, we implement the basic LSTM-based model and evaluate each framework[4] with 128 hidden layer nodes by CV and $Accuracy$, as presented in Table III. Additionally, in the real world, the trained models are often larger and more complex. To simulate real-life

[4] To maintain model utility, as well as reducing communication costs, the penalty coefficient β may be different for each scenario.

TABLE III
PERFORMANCE COMPARISON (128 NODES) ACROSS VARYING HOSPITAL COUNTS (m) AND DATA SEGMENTS (s)

Learning Framework	Communication Volume (MB)					*Accuracy*				
	m=4	m=5	m=6	m=5	m=6	m=4	m=5	m=6	m=5	m=6
	s=3	s=3	s=3	s=4	s=4	s=3	s=3	s=3	s=4	s=4
`STSL (w/o S)`	61.57	124.43	229.19	308.86	867.58	0.902	0.873	0.874	0.839	0.758
`STSL (w/o D)`	55.17 *(89.60%)*	99.98 *(80.35%)*	203.00 *(88.57%)*	252.41 *(81.72%)*	721.5 *(83.16%)*	0.902	0.873	0.874	0.839	0.758
`STSL (w/o C)`	23.80 *(38.66%)*	43.01 *(34.56%)*	54.65 *(23.84%)*	83.45 *(27.02%)*	180.61 *(20.82%)*	**0.919**	**0.951**	**0.943**	**0.858**	**0.912**
`STSL`	**17.40** ***(28.26%)***	**24.38** ***(19.60%)***	**31.95** ***(13.94%)***	**48.53** ***(15.71%)***	**82.83** ***(9.55%)***	**0.919**	**0.951**	**0.943**	**0.858**	**0.912**

TABLE IV
PERFORMANCE COMPARISON (1024 NODES) ACROSS VARYING HOSPITAL COUNTS (m) AND DATA SEGMENTS (s)

Learning Framework	Communication Volume (GB)					Data Retention Ratio (%)				
	m=4	m=5	m=6	m=5	m=6	m=4	m=5	m=6	m=5	m=6
	s=3	s=3	s=3	s=4	s=4	s=3	s=3	s=3	s=4	s=4
`STSL (w/o S)`	2.45	5.89	11.63	15.54	46.15	100	100	100	100	100
`STSL (w/o D)`	2.10 *(85.68%)*	4.54 *(77.27%)*	10.20 *(87.67%)*	12.44 *(80.09%)*	38.15 *(82.66%)*	100	100	100	100	100
`STSL (w/o C)`	0.94 *(38.55%)*	1.89 *(32.14%)*	2.76 *(23.69%)*	4.46 *(28.68%)*	7.00 *(15.20%)*	73.88	66.54	66.49	74.07	66.34
`STSL`	**0.63** ***(25.53%)***	**0.87** ***(14.83%)***	**1.19** ***(10.25%)***	**1.84** ***(11.85%)***	**2.96** ***(6.42%)***	73.88	66.54	66.49	74.07	66.34

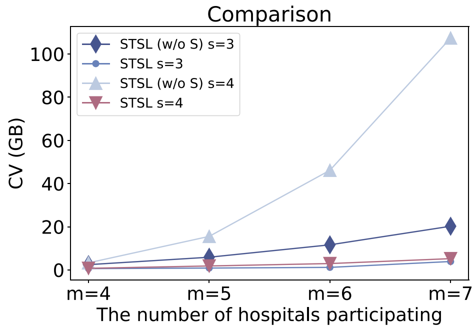


Fig. 8. Communication volume trends across varying hospital counts (m) and data segments (s).

situations, the CVs and DRs of large models with 1024 nodes are also illustrated in Table IV.

It can be observed that both *Data Selection* and *Combinatorial Optimization* operations lead to a decrease in CV. Compared to *Combinatorial Optimization*, which modestly reduces CV (typically by $10\% \sim 20\%$) without compromising data, *Data Selection* sacrifices a portion of data to achieve a significant reduction in CV (usually $60\% \sim 70\%$). With judicious discarding of segments, more samples could be consolidated together, so that $Accuracy$ is improved ($1.7\% \sim 15.4\%$), especially in complicated scenarios with more hospitals participating and patients having more transfer records. Meanwhile, the effect of *Communication Optimization* also becomes increasingly evident in complex scenarios.

As shown in Fig 8, in a conventional learning framework, CV grows exponentially. Whereas, in `STSL` framework, the growth is relatively smooth. Furthermore, in large model environments, the original CV is overwhelmingly high. After scheduling through `STSL`, the CV can be reduced to an acceptable value while retaining a majority of data. This observation shed light on cooperatively training a big model.

TABLE V
PERFORMANCE WITH DIFFERENT PENALTY SETTINGS

β	CV_r/CV_o (%)	DR (%)	Number of Distinct Sequences	
			$len = 3$	$len = 2$
[0.15, 0.45, 1.5, 1.8]	3.56	50.67	1	26
[0.25, 1, 2.5, 3]	6.87	61.72	35	24
[0.4, 1.6, 4, 4.8]	9.48	71.60	81	12

(Exp#3) Effects of the Data Penalty: In a scenario where the model parameters (128 hidden layer nodes) and healthcare scenario ($m = 6$ and $s = 4$) remain static, we conduct comparisons to analyze the impact of different penalty settings on training scheduling. During this experiment, each penalty setting has $\eta = [1, 0.7, 0.6, 0.4]$, only the staircase penalty value β differing in each setting. The comparative results of scheduling are presented in Table V. It can be inferred that as penalty values increase, scheduling tends to retain more segments of patient visits, leading to an improvement in DR (20.93%), accompanied by a slight increase in CV (5.92%), which falls in an acceptable range. As illustrated in Exp#2, fewer batches may bring improvement in $Accuracy$, but more data discarding may result in distorted data which injures model performance. In practice, the values of the data penalty could be further adjusted to achieve even better performance.

VII. DISCUSSION

Practicability and Generalizability: With the increasing mobility of the population and the improved implementation of the hierarchical consultation system, people's healthcare data is often segmented and distributed across multiple hospitals. According to recent findings, Chinese residents made an average of six visits to healthcare facilities in 2022 [10], with 110.5 million of these visits occurring across provinces [11]. The widespread presence of segmented data highlights the immediate promise and broad potential of STSL for learning temporally and spatially distributed data, which is heterogeneous both within individual patient records across hospitals and among visit sequences of different patients.

STSL's ability to handle temporally and spatially distributed data makes it particularly suitable for various domains, such as activity monitoring across different autonomous domains (e.g., anxiety analysis with user data on vehicles and watches), intelligent operation and maintenance in distributed systems (e.g., workload migration across different nodes of supercomputer system), and network measurement tasks (e.g., traffic analysis over multiple routers). These scenarios, which predominantly involve time-series data collected by diverse institutions, can readily adapt to the STSL framework. We also believe that the integration of STSL with the split AI inference framework [40], [41], [42] opens new possibilities for real-time and distributed intelligent systems, enabling efficient data processing and decision-making across complex, multi-domain environments.

Differential Weights: As multiple sequences and hospitals are involved in the learning, an intuitive improvement is assigning different weights to them. On the one hand, one could use the attention mechanism [43] to prioritize hospitals with vaster medical records by assigning more model layers or blocks to them and setting more weights to their sub-models, as these hospitals better record patient health status. On the other hand, hospitals vary in disease-specific expertise, so one could set more model layers or blocks for those hospitals that are known to be excellent for diagnosing specific diseases and more weights for their sub-models.

Model Substitution and Tuning: We state that the basic model (i.e., LSTM) used for splitting learning in this work could be easily replaced by other mainstream sequential data analysis models, such as Transformer and GNN. This flexibility is achieved through customizable model partitioning mechanisms that distribute layers or units across participating clients, coupled with adaptive aggregation schemas tailored to specific model structures. Additionally, STSL enables advanced fine-tuning techniques, such as genetic algorithms [44] and Bayesian optimization [45], for precise parameter adjustment and performance optimization.

VIII. Conclusion

This work focuses on learning sequential data and identifies, in the case of healthcare data, the segmented situation of data that is generated by multiple users now and then and stored in different institutes. STSL has thus been proposed as a framework for exploring the sequential knowledge in the segmented data. In STSL, we provide the privacy-preserving visit ordering method to find out the visit sequences of multiple patients paying visits to multiple hospitals without exposing either each patient's visit time or visit event. Meanwhile, we have also presented the scheduling design that separates all the segmented data into order-aligned batches for consecutive split learning. Theoretical proof demonstrates the security of the framework for learning about patient data and the ordering process for keeping visit information private. Dedicated to the learning of segmented data, both the privacy-preserving design and the batch-based learning paradigm in STSL could shed light on the exploration of such data forms. In the future, we believe extending STSL to streaming data for learning on dynamically generated data is worth investigating.

References

[1] L. Ma et al., "Adacare: Explainable clinical health status representation learning via scale-adaptive feature extraction and recalibration," in *Proc. AAAI Conf. Artif. Intell.*, 2020, pp. 825–832.

[2] X. Yu, S. H. Wang, and Y. D. Zhang, "CGNet: A graph-knowledge embedded convolutional neural network for detection of pneumonia," *Inf. Process. Manage.*, vol. 58, no. 1, 2021, Art. no. 102411.

[3] T. Bai, B. L. Egleston, S. Zhang, and S. Vucetic, "Interpretable representation learning for healthcare via capturing disease progression through time," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, pp. 43–51, 2018.

[4] M. Oldenhof et al., "Industry-scale orchestrated federated learning for drug discovery," in *Proc. AAAI Conf. Artif. Intell.*, 2023, pp. 15576–15584.

[5] E. Parliament, "General data protection regulation," EU, 2016. [Online]. Available: https://gdpr-info.eu/art-4-gdpr/

[6] P. C. et al., "Privacy concerns of the Australian my health record: Implications for other large-scale opt-out personal health records," *Inf. Process. Manage.*, vol. 57, no. 6, 2020, Art. no. 102364.

[7] H. Du et al., "Red meat, poultry and fish consumption and risk of diabetes: A 9 year prospective cohort study of the China kadoorie biobank," *Diabetologia*, vol. 63, pp. 767–779, 2020.

[8] S. Abuadbba et al., "Can we use split learning on 1D CNN models for privacy preserving training?," in *Proc. 15th ACM Asia Conf. Comput. Commun. Secur.*, 2020, pp. 305–318, 2020.

[9] Y. J. Ha, G. Lee, M. Yoo, S. Jung, S. Yoo, and J. Kim, "Feasibility study of multi-site split learning for privacy-preserving medical systems under data imbalance constraints in COVID-19, X-ray, and cholesterol dataset," *Sci. Rep.*, vol. 12, no. 1, pp. 1–11, 2022.

[10] N. H. Commission, "Statistical bulletin on the development of health care in China 2022," 2022. [Online]. Available: http://www.nhc.gov.cn/guihuaxxs/s3586s/202310/5d9a6423f2b74587ac9ca41ab0a75f66.shtml

[11] N. H. S. Administration, "Statistical bulletin on the development of national healthcare security in 2022," 2022. [Online]. Available: http://www.nhsa.gov.cn/art/2023/7/10/art_7_10995.html

[12] CHMIA, "Expert consensus on the application of critical care Big Data in China," *Chin. Med. J.*, vol. 103, no. 6, pp. 404–424, 2023.

[13] L. N. Sanchez-Pinto, Y. Luo, and M. M. Churpek, "Big data and data science in critical care," *Chest*, vol. 154, no. 5, pp. 1239–1248, 2018.

[14] P. Le Roux et al., "Consensus summary statement of the international multidisciplinary consensus conference on multimodality monitoring in neurocritical care: A statement for healthcare professionals from the neurocritical care society and the European society of intensive care medicine," *Neurocritical Care*, vol. 21, pp. 1–26, 2014.

[15] T. J. Pollard, A. E. Johnson, J. D. Raffa, L. A. Celi, R. G. Mark, and O. Badawi, "The eicu collaborative research database, a freely available multi-center database for critical care research," *Sci. Data*, vol. 5, no. 1, pp. 1–13, 2018.

[16] P. Xu et al., "Critical care database comprising patients with infection," *Front. Public Health*, vol. 10, 2022, Art. no. 852410.

[17] X. Zhang et al., "Data-driven subtyping of Parkinson's disease using longitudinal clinical records: A cohort study," *Sci. Rep.*, vol. 9, no. 1, pp. 1–12, 2019.

[18] J. Luo, M. Ye, C. Xiao, and F. Ma, "HiTANet: Hierarchical time-aware attention networks for risk prediction on electronic health records," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2020, pp. 647–656.

[19] Y. Xu et al., "VecoCare: Visit sequences-clinical notes joint learning for diagnosis prediction in healthcare data," in *Proc. Int. Joint Conf. Artif. Intell.*, 2023, pp. 4921–4929.

[20] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, “Communication-efficient learning of deep networks from decentralized data,” in *Proc. Artif. Intell. Statist.*, PMLR, 2017, pp. 1273–1282.

[21] P. Vepakomma, O. Gupta, T. Swedish, and R. Raskar, “Split learning for health: Distributed deep learning without sharing raw patient data,” 2018, *arXiv: 1812.00564*.

[22] Q. Li, B. He, and D. Song, “Adversarial collaborative learning on Non-IID features,” in *Proc. Mach. Learn. Res.*, 2024, pp. 20210–20220.

[23] Y. Diao, Q. Li, and B. He, “Towards addressing label skews in one-shot federated learning,” in *Proc. 11th Int. Conf. Learn. Representations*, 2023, pp. 1321–1331.

[24] Y. Miao, R. Xie, X. Li, Z. Liu, K.-K. R. Choo, and R. H. Deng, “Efficient and secure federated learning against backdoor attacks,” *IEEE Trans. Dependable Secure Comput.*, vol. 21, no. 5, pp. 4619–4636, Sep./Oct. 2024.

[25] Y. El Zein, M. Lemay, and K. Huguenin, “PrivaTree: Collaborative privacy-preserving training of decision trees on biomedical data,” *IEEE/ACM Trans. Comput. Biol. Bioinf.*, vol. 21, no. 1, pp. 1–13, Jan./Feb. 2024.

[26] L. Jiang, Y. Wang, W. Zheng, C. Jin, Z. Li, and S. G. Teo, “LSTMSPLIT: Effective split learning based LSTM on sequential time-series data,” 2022, *arXiv:2203.04305*.

[27] A. Abedi and S. S. Khan, “FedSL: Federated split learning on distributed sequential data in recurrent neural networks,” *Multimedia Tools Appl.*, vol. 83, pp. 28891—28911, 2023.

[28] Z. Marszałek, “Parallel fast sort algorithm for secure multiparty computation,” *J. Universal Comput. Sci.*, vol. 24, no. 4, pp. 488–514, 2018.

[29] H. Dai, H. Ren, Z. Chen, G. Yang, and X. Yi, “Privacy-preserving sorting algorithms based on logistic map for clouds,” *Secur. Commun. Netw.*, vol. 2018, pp. 126–137, 2018.

[30] W. Ning, G. Haomin, Z. Tong, and I. C. Company, “A practical and efficient secure multi-party sort protocol,” *Comput. Appl. Softw.*, vol. 14, no. 5, pp. 458–474, 2018.

[31] A. Wigderson, M. Or, and S. Goldwasser, “Completeness theorems for noncryptographic fault-tolerant distributed computations,” in *Proc. 20th Annu. Symp. Theory Comput.*, 1988, pp. 351–371.

[32] S. Li, R. Du, Y. Yang, and W. Qiong, “Secure multiparty multi-data ranking,” *Chin. J. Comput.*, vol. 43, no. 8, pp. 1448–1462, 2020.

[33] F. Dai, Y. Chen, Z. Huang, and H. Zhang, “WRHT: Efficient all-reduce for distributed DNN training in optical interconnect systems,” in *Proc. 52nd Int. Conf. Parallel Process.*, 2023, pp. 556–565.

[34] R. Fukasawa, J. Lysgaard, M. P. De Aragão, M. Reis, E. Uchoa, and R. F. Werneck, “Robust branch-and-cut-and-price for the capacitated vehicle routing problem,” *Lecture Notes Comput. Sci. (Including Subseries Lecture Notes Artif. Intell. Lecture Notes Bioinf.)*, vol. 3064, pp. 1–15, 2004.

[35] L. Duan et al., “Efficiently solving the practical vehicle routing problem: A novel joint learning approach,” in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2020, pp. 3054–3063.

[36] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*. Chelmsford, MA, USA: Courier Corporation, 1998.

[37] O. Li et al., “Label leakage and protection in two-party split learning,” 2021, *arXiv:2102.08504*.

[38] T. Stevens, C. Skalka, C. Vincent, J. Ring, S. Clark, and J. Near, “Efficient differentially private secure aggregation for federated learning via hardness of learning with errors,” in *Proc. 31st USENIX Secur. Symp.*, 2022, pp. 1379–1395.

[39] S. Merity, N. S. Keskar, and R. Socher, “Regularizing and optimizing LSTM language models,” 2017, *arXiv: 1708.02182*.

[40] D. Wen et al., “Task-oriented sensing, computation, and communication integration for multi-device edge AI,” *IEEE Trans. Wireless Commun.*, vol. 23, no. 3, pp. 2486–2502, Mar. 2024.

[41] D. Wen et al., “Over-the-air federated edge learning with integrated sensing, communication, and computation,” in *Proc. IEEE/CIC Int. Conf. Commun. China*, 2024, pp. 862–867.

[42] D. Wen, X. Li, Y. Zhou, Y. Shi, S. Wu, and C. Jiang, “Integrated sensing-communication-computation for edge artificial intelligence,” *IEEE Internet Things Mag.*, vol. 7, no. 4, pp. 14–20, Jul. 2024.

[43] L. Wang, L. Wong, Z. You, and D. Huang, “AMDECDA: Attention mechanism combined with data ensemble strategy for predicting CircRNA-disease association,” *IEEE Trans. Big Data*, vol. 10, no. 4, pp. 320–329, Aug. 2024.

[44] N. Gorgolis, I. Hatzilygeroudis, Z. Istenes, and L.-G. Gyenne, “Hyperparameter optimization of LSTM network models through genetic algorithm,” in *Proc. 10th Int. Conf. Inf., Intell., Syst. Appl.*, 2019, pp. 1–4.

[45] B. Alizadeh, A. G. Bafti, H. Kamangir, Y. Zhang, D. B. Wright, and K. J. Franz, “A novel attention-based LSTM cell post-processor coupled with Bayesian optimization for streamflow prediction,” *J. Hydrol.*, vol. 601, 2021, Art. no. 126526.

Ling Hu received the BE and MS degrees from the College of Computer Science and Technology, National University of Defense Technology (NUDT), in 2022 and 2024, respectively. She is currently working toward the PhD degree with the College of Computer Science and Technology, National University of Defense Technology. Her research interests include visual analysis, data privacy, and artificial intelligence.

Tongqing Zhou received the BS, MS, and PhD degrees from the College of Computer Science and Technology from the National University of Defense Technology (NUDT), Changsha, in 2012, 2014, and 2018, respectively. He is currently an assistant pesearcher with the College of Computer Science and Technology, NUDT. His main research interests include network measurement, crowdsensing, and data privacy. He is the recipient of Outstanding PhD Dissertation Award and Outstanding Postdoc, both of Hunan Province, China.

Zhihuang Liu received the BE and MS degrees from the College of Computer and Data Science, Fuzhou University, in 2020 and 2023, respectively. He is currently working toward the PhD degree with the College of Computer Science and Technology, National University of Defense Technology (NUDT). His research interests include blockchain, IoT security, and machine learning.

Fang Liu received the BS and PhD degrees from the College of Computer Science and Technology, National University of Defense Technology, Changsha, China, in 1999 and 2005, respectively. She is a full professor with the School of Design, Hunan University. Her main research interests include edge computing, data storage and management, and intelligent design.

Zhiping Cai (Member, IEEE) received the BEng, MASc, and PhD degrees from the College of Computer Science and Technology, National University of Defense Technology (NUDT), China, in 1996, 2002, and 2005, respectively. He is a full professor with the College of Computer Science and Technology, NUDT. His current research interests include artificial intelligence, network security, and Big Data. He is a senior member of the CCF.