Practical Challenges in Differentially-Private Federated Survival Analysis of Medical Data

Shadi Rahimian SHADI.RAHIMIAN@CISPA.SAARLAND
CISPA Helmholtz Center for Information Security, Germany

Raouf Kerkouche RAOUF.KERKOUCHE@CISPA.DE
CISPA Helmholtz Center for Information Security, Germany

Ina Kurth INA.KURTH@DKFZ-HEIDELBERG.DE
DKFZ German Cancer Research Center, Germany

Mario Fritz FRITZ@CISPA.DE
CISPA Helmholtz Center for Information Security, Germany

Abstract

Survival analysis or time-to-event analysis aims to model and predict the time it takes for an event of interest to happen in a population or an individual. In the medical context this event might be the time of dying, metastasis, recurrence of cancer, etc. Recently, the use of neural networks that are specifically designed for survival analysis has become more popular and an attractive alternative to more traditional methods. In this paper, we take advantage of the inherent properties of neural networks to federate the process of training of these models. This is crucial in the medical domain since data is scarce and collaboration of multiple health centers is essential to make a conclusive decision about the properties of a treatment or a disease. To ensure the privacy of the datasets, it is common to utilize differential privacy on top of federated learning. Differential privacy acts by introducing random noise to different stages of training, thus making it harder for an adversary to extract details about the data. However, in the realistic setting of small medical datasets and only a few data centers, this noise makes it harder for the models to converge. To address this problem, we propose DPFed-post which adds a post-processing stage to the private federated learning scheme. This extra step helps to regulate the magnitude of the noisy average parameter update and easier convergence of the model. For our experiments, we choose 3 real-world datasets in the realistic setting when each health center has only a few hundred records, and we show that DPFed-post successfully increases the performance of the models by an average of up to 17% compared to the standard differentially private federated learning scheme.

Data and Code Availability The pre-processed data for our experiments can be obtained through the **pycox** package [1]. We also submit our code as supplemental material alongside this paper.

1. Introduction

Survival analysis or event history analysis is an old branch of statistics dating back to the 17th century (Camilleri, 2019). The goal of survival analysis is to predict the time an event of interest occurs. This event can be the time of death of a patient(e.g. Goldhirsch et al., 1989; Brenner, 2002), the time of default for a bank customer(e.g. Baesens et al., 2005; Dirick et al., 2017; Stepanova and Thomas, 2002; Laitinen, 2005), time of unsubscribing from an online service(e.g. Mishachandar and Kumar, 2018; Lu, 2002; Lee et al., 2019), time until a mechanical system fails(e.g. Hanson, 2004; Styc and Lagacherie, 2016) and so on. The assumption is that for each population, there is a mapping from the observed features of the individuals and their time of event. Survival analysis tools are used to learn this mapping from past data that have experienced the event and generalize to the whole population.

As for any predictive model, it is important to have enough data points to achieve generalization. This is especially of great significance in the case of medical data with many outliers and complicated patterns. However, data is sparse and usually the centers that

1. https://github.com/havakv/pycox

collect these data (e.g. hospitals or banks) are reluctant or not allowed to share their data with each other due to privacy and security reasons.

This calls for federated learning in the case of survival analysis. Federated learning McMahan et al. (2017) grants the data owners the possibility of keeping their data locally, but participate in training of a global model, jointly, with other data owners. This is achievable by choosing a trusted central server and a machine learning model that is trained locally by each participant. The role of the central server is to iteratively collect these trained models, update the global model based on the updated local models and share the updated global model with participants. Despite the obvious benefits of learning survival models in a federated setting, there has been no study on the practical effects of federation on survival analysis tools.

Although federated learning provides a scheme in which no direct sharing of data is required, there is still the risk of information leakage through parameters/output of the model. It has been shown that an adversary can carry out successful reconstruction attacks (Zhu et al., 2019; Geiping et al., 2020), membership inference attacks (Nasr et al., 2019; Melis et al., 2019) and feature leakage attacks (Melis et al., 2019).

One solution for protecting the privacy of these models and their training data is to utilize differential privacy (DP) (Dwork and Roth, 2014). DP offers privacy through a noise addition mechanism and is based on rigorous theoretical guarantees against data information leakage. In federated learning, DP can be directly applied on the clients side such that any record in any dataset of any client is protected (record-level DP e.g. in Truex et al., 2020; Kerkouche et al., 2021c) or it can be applied on the updates that are shared by each client with the central server such that the client's dataset as a whole is protected. (client-level DP e.g. in Geyer et al., 2017). The latter is preferable as it provides tighter privacy gurantees.

In this paper, we focus on real-world challenges of DP federated learning of survival models for medical data, where data and also the number of clients that participate in the federated learning is limited. Here, the noise of client-level DP - which scales proportional to the client sampling probability - is so large that it prevents the model from converging. To tackle this problem, we take advantage of the post-processing property of differential privacy and make an additional step of clipping the noisy average update of clients by a reasonable value. This method can be viewed as using a regularization for the learning rate of the global model.

Our contributions are:

- We evaluate the effect of federation of survival models on real-world datasets
- We discuss what the challenges would be in federation of these models for the realistic setting when a few clients each possessing only a few hundred data points wish to collaborate.
- We successfully apply a post-processing step for client-level differentially-private federated learning and observe that the performances of the model are consistently improved over different datasets and models when this trick is used.

2. Background

In this section, we will give an overview of all the necessary concepts to understand our approach. We first describe survival analysis in a more formal way and introduce some traditional methods that are used for time-to-event prediction. We explain why in the medical setting it is important that multiple parties collaborate and share their data such that a richer, more generalizable model can be learned.

We then introduce federated learning which is a well-motivated and popular solution to collaborative training of models. By using federated learning, different parties each holding their own dataset can jointly train a unified model.

Naturally, collaborative training of a model is prone to privacy breaches. So at last we give a brief overview of differential privacy as a privacy-preserving method that can be deployed on top of federated learning.

2.1. Survival Analysis

Survival analysis is the collection of the tools for analysing and predicting the time duration until a specific event happens. For example, in the case of clinical data, this event can be the time of death for the patients.

The *survival function* $S(t)$ and the *cumulative incidence function* (CIF) $F(t)$ can be defined as follows:

$$S(t) = \mathrm{P}(T > t) = 1 - F(t) \tag{1}$$

where $\mathrm{P}(T > t)$ indicates the probability that the event of interest T happens after time t. So the survival function is interpreted as the probability of the event happening after time t and the complementary CIF is the probability of the event happening before or at time t. We can also define the *hazard rate* $h(t)$:

$$h(t) = \lim_{\Delta t \to 0} \frac{1}{\Delta t} \mathrm{P}(t \leq T < t + \Delta t | T \geq t) \tag{2}$$

Thus the survival function can be calculated from the hazard rate by:

$$S(t) = \exp[-\int_0^t h(s)ds] = \exp[-H(t)], \tag{3}$$

where $H(t)$ is called the cumulative hazard. We can observe that having one of these functions is adequate to calculate the others, but each offers a unique perspective into the survival status of the data.

For real-world data, the event we are interested in, e.g. death or recovery, may not be observed for all data points. This can happen when, for example, the patient does not participate in the follow-ups, or they die due to a cause unrelated to the original study, or simply when they have not yet experienced the event of interest at the last follow-up of the study. These data points are said to be *right-censored*. So the time of event is chosen as $T = \min\{T^*, C\}$, where T^* is the true time of event for the individual and C is the time of censoring. Survival data usually comes in the form of:

$$\mathcal{X} = \{\mathbf{x}_i, y_i\}_{i=1}^N = \{\mathbf{x}_i, T_i, E_i\}_{i=1}^N \tag{4}$$

for N individuals. Here, $\mathbf{x}_i$ and y_i are the vector of covariates or features and the label for data point i, respectively. The features vector can, for example, contain information about the age, gender, medical history, tumor size, etc. for medical data. y_i can further be decomposed into the tuple of $\{T_i, E_i\}$ where T_i is the time of event for the event type E_i for individual i. It is customary to set $E = 0$ for individuals that are right-censored.

Given this set $\mathcal{X}$, one way to calculate the survival functions is by *Cox Proportional Hazards* (Cox, 1972) model:

$$h(t|\mathbf{x}) = h_0(t)g(\mathbf{x}) \qquad \text{where} \qquad g(\mathbf{x}) = \beta^T \mathbf{x} \tag{5}$$

The main assumption of the Cox proportional hazards model is that the features $\mathbf{x}$ are independent and a linear combination of them and a non-parametric *baseline hazard* $h_0(t)$ are sufficient to model the data. To fit this model, first the coefficients β are found by maximizing the partial log likelihood of the model:

$$L = \Pi_i \left(\frac{\exp[g(\mathbf{x}_i)]}{\sum_{j \in R_i} \exp[g(\mathbf{x}_j)]} \right)^{E_i} \tag{6}$$

where R_i is the risk set of all individuals at risk at time T_i.

In the second step, the baseline hazard is estimated for the parameters found in the first step.

The *proportionality* in the name of this model comes from the fact that for two data points $\mathbf{x}_i$ and $\mathbf{x}_j$ the hazard ratio remains constant over time:

$$\frac{h(t|\mathbf{x}_i)}{h(t|\mathbf{x}_j)} = \frac{h_0(t)g(\mathbf{x}_i)}{h_0(t)g(\mathbf{x}_j)} = \frac{g(\mathbf{x}_i)}{g(\mathbf{x}_j)} \tag{7}$$

To summarize, the main goal of survival analysis is to model the relationship between the vector of features of the data points and the time that an event of interest happens. This model is fitted on a set of data points that we assume is representative of the statistical properties of a population. The more datapoints are used to fit the model, the more generalizable the model will be, as each datapoint adds more information about the properties of the population.

Unfortunately, the collection of data is expensive and time consuming for the health centers, and the number of patients they receive is also limited. This means that by using only the data of one health center, we would have a high bias and the fitted model would not be well-generalizable. As a solution for this problem, we look at the federated learning for survival models, in the next section. Federated learning allows multiple data-holders to jointly learn a model over the union of their data.

2.2. Federated Learning

One of the major issues for modeling survival data is the lack of enough training data. Oftentimes many separate data owners have access to limited amounts of data and due to reasons, such as privacy and security, are unwilling to simply share their data with each other to learn a richer model. This is where federated learning McMahan et al. (2017); Shokri and Shmatikov (2015) becomes important. Federated learning is the concept of distributing the learning process of machine learning models among several data owners, without the need to access their local data.

In this paper we use deep neural networks designed for survival analysis as the model that we wish to federate. In general to train a deep neural network, an objective function is optimized over the parameters of the model, $\mathbf{w}$, for a training dataset which contains n data points:

$$\min_{\mathbf{w}} f(\mathbf{w}) \text{ where } f(\mathbf{w}) = \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(\mathbf{x}_i, y_i; \mathbf{w}) \tag{8}$$

where $\mathcal{L}(\mathbf{x}_i, y_i; \mathbf{w})$ is the loss function for i-th data point with label y_i, and the training is done iteratively over the dataset. Therefore by defining the appropriate loss function for survival analysis tasks, we can train the network to capture the behavior of the stochastic processes from the training set.

In federated learning a central server holds a global model which is also shared with all the clients. Clients train this model locally and only send their updated model parameters to the central server. The server updates the parameters of the global model with a weighted average of the local parameters and re-shares the updated global model with the clients. To reduce the cost of communication and computation, usually at each round, only a fraction of clients, $C = K/N$, are chosen randomly to train their local model. Here C is the sampling probability of clients, K is the number of clients chosen out of a total of N clients at each round:

$$\mathbf{w}^r \leftarrow \mathbf{w}^{r-1} + \frac{\sum_{k=1}^{K}(\mathbf{w}_k^r - \mathbf{w}^{r-1})}{K} \tag{9}$$

where $\mathbf{w}_k^r$ is the model parameters for client k after the r-th round of communication with the central server. These steps are done for a total number of T_{cl} communication rounds. The local parameters $\mathbf{w}_k^r$ are computed by each client using the standard training algorithms, such as stochastic gradient decent (Robbins and Monro, 1951; Kiefer and Wolfowitz, 1952). Here, it is assumed that each client possesses n_k training data and total number of data points can be calculated as $n = \sum_{k=1}^{K} n_k$.

Although Federated Learning improves privacy by design, model parameters can leak information about the training data. Indeed, Zhu et al. (2019); Zhao et al. (2020); Geiping et al. (2020) presented some attacks that allow an adversary to reconstruct some training data samples of some entities. Nasr et al. (2019); Melis et al. (2019) define a membership attack that allows to infer if a particular record is included in the data of a specific entity. Similarly, Melis et al. (2019) define an attack which aims at inferring if a group of people with a specific property, like for example skin color or ethnicity, is included in the dataset of a particular participating entity. A solution to prevent these attacks and provide theoretical guarantees in to use a privacy model called Differential Privacy (Dwork and Roth, 2014).

2.3. Differential Privacy

Differential privacy (DP) allows an entity to privately release information about a dataset: a function of a record dataset is perturbed, so that any information which can differentiate this record from the rest of the dataset is bounded Dwork and Roth (2014).

Definition 1 (Privacy loss) *Let $\mathcal{A}$ be a privacy mechanism which assigns a value Range($\mathcal{A}$) to a dataset D. The privacy loss of $\mathcal{A}$ with datasets D and D' at output $O \in$ Range($\mathcal{A}$) is a random variable $\mathcal{P}(\mathcal{A}, D, D', O) = \log \frac{\Pr[\mathcal{A}(D)=O]}{\Pr[\mathcal{A}(D')=O]}$ where the probability is taken on the randomness of $\mathcal{A}$.*

Definition 2 ((ϵ, δ)-Differential Privacy) *A privacy mechanism $\mathcal{A}$ guarantees (ε, δ)-differential privacy if for any database D and D', differing on at most one record, $\Pr_{O\sim\mathcal{A}(D)}[\mathcal{P}(\mathcal{A}, D, D', O) > \varepsilon] \leq \delta$ Dwork and Roth (2014).*

This guarantees that an adversary, who has access to the output of $\mathcal{A}$, can draw almost the same conclusions up to ε (with probability larger than $1 - \delta$) about any record no matter if it is included in the input of $\mathcal{A}$ or not.

Moments Accountant. Differential privacy maintains composition; the privacy guarantee of the k-fold adaptive composition of $\mathcal{A}_{1:k} = \mathcal{A}_1, \ldots, \mathcal{A}_k$ can be computed using the moments accountant method Abadi et al. (2016). In particular, it follows from Markov's inequality that $\Pr[\mathcal{P}(\mathcal{A}, D, D', O) \geq \varepsilon] \leq \mathbb{E}[\exp(\lambda\mathcal{P}(\mathcal{A}, D, D', O))]/\exp(\lambda\varepsilon)$ for any output $O \in$ Range($\mathcal{A}$) and $\lambda > 0$. $\mathcal{A}$ is (ε, δ)-DP with $\delta = \min_\lambda \exp(\alpha_\mathcal{A}(\lambda) - \lambda\varepsilon)$, where $\alpha_\mathcal{A}(\lambda) = \max_{D,D'} \log \mathbb{E}_{O\sim\mathcal{A}(D)}[\exp(\lambda\mathcal{P}(\mathcal{A}, D, D', O))]$ is the log of the moment generating function of the privacy loss. The privacy guarantee of the composite mechanism $\mathcal{A}_{1:k}$ can be calculated using that $\alpha_{\mathcal{A}_{1:k}}(\lambda) \leq \sum_{i=1}^{k} \alpha_{\mathcal{A}_i}(\lambda)$ Abadi et al. (2016).

Gaussian Mechanism. A fundamental concept of all DP sanitization techniques is the *global sensitivity* of a function (Dwork and Roth, 2014).

Definition 3 (Global L_p-sensitivity) *For any function $f : \mathcal{D} \to \mathbb{R}^n$, the L_p-sensitivity of f is $\Delta_p f = \max_{D,D'} ||f(D) - f(D')||_p$, for all D, D' differing in at most one record, where $||\cdot||_p$ denotes the L_p-norm.*

The Gaussian Mechanism (Dwork and Roth, 2014) consists of adding Gaussian noise to the true output of a function. In particular, for any function $f : \mathcal{D} \to \mathbb{R}^n$, the Gaussian mechanism is defined as adding i.i.d Gaussian noise with variance $(\Delta_2 f \cdot \sigma)^2$ and zero mean to each coordinate value of $f(D)$. Recall that the pdf of the Gaussian distribution with mean μ and variance ξ^2 is $\mathsf{pdf}_{\mathcal{G}(\mu,\xi)}(x) = \frac{1}{\sqrt{2\pi}\xi} \exp\left(-\frac{(x-\mu)^2}{2\xi^2}\right)$.

In fact, the Gaussian mechanism draws vector values from a multivariate spherical (or isotropic) Gaussian distribution which is described by random variable $\mathcal{G}(f(D), \Delta_2 f \cdot \sigma \mathbf{I}_n)$, where n is omitted if its unambiguous in the given context.

3. Approach

The most successful research in privacy-preserving federeted learning has been studied on huge datasets, con-

taining up to millions of data records and hundreds of clients (see e.g. Kairouz et al., 2019; Augenstein et al., 2019; McMahan et al., 2017; Truex et al., 2019). However, in the realistic setting of medical application, the data is very scarce and the number of clients that are willing to collaborate, i.e. health centers, is also limited. This poses problems such as instability of results as well as sensitivity of the model to the DP noise when a privacy-preserving solution is needed.

In this section, we first define the privacy model and the layer we choose to add differential privacy to, then discuss the problems that this noise would cause in the case of small datasets. We propose our post-processing technique which does not change the DP guarantees, however, acts as a regularization on the learning rate thus stabilizing the convergence of the model during the federated training.

Finally, we present our experiments using 4 different survival deep neural networks and compare the results we obtain by federation, differentially private federation as well as applying the post-processing step.

Algorithm 1

`Differentially Private Federated Learning with post processing (DPFed-post).`

N total clients, local mini-batch size B, local epochs E, communication rounds T_{cl}, learning rate η, sensitivity S and post-processing parameter P.

Initialize $\mathbf{w}_0$ and send the model to clients
for $r = 1, ...T_{cl}$
  Select K clients randomly
  for each selected client $k = 1, ..., K$
    $\mathbf{w}_k^r \leftarrow \text{ClientUpdate}(k, \mathbf{w}^{r-1})$
    $\Delta\mathbf{w}_k^r \leftarrow \mathbf{w}_k^r - \mathbf{w}^{r-1}$
    $\Delta\hat{\mathbf{w}}_k^r \leftarrow \Delta\mathbf{w}_k^r / \max\left(1, \frac{||\Delta\mathbf{w}_k^r||_2}{S}\right)$
  $\Delta\mathbf{w}^r \leftarrow \frac{\sum_{k=1}^{K} \Delta\hat{\mathbf{w}}_k^r + \mathcal{G}(0, S\sigma\mathbf{I})}{K}$
  $\Delta\hat{\mathbf{w}}^r \leftarrow \Delta\mathbf{w}^r / \max\left(1, \frac{||\Delta\mathbf{w}^r||_2}{P}\right)$
  $\mathbf{w}^r \leftarrow \mathbf{w}^{r-1} + \Delta\hat{\mathbf{w}}^r$
ClientUpdate$(k, \mathbf{w})$
  for client k
  for $i = 1, ..., E$
    for local batches b
      $\mathbf{w} \leftarrow \mathbf{w} - \eta\nabla l(b; \mathbf{w})$
  return $\mathbf{w}$ to server

3.1. Privacy Model

We consider an adversary, or a set of colluding adversaries, who can access any update vector sent by the server at each round of the protocol. A plausible adversary is a participating entity, i.e. a malicious client, that wants to infer the training data used by other participants. We assume that the server is trusted. The adversary is *passive* (i.e., honest-but-curious), that is, it follows the learning protocol faithfully.

Different privacy requirements can be considered depending on what information the adversary aims to infer. In general, private information can be inferred about:

- any record (user or patient) in any dataset of any client (*record-level privacy*),
- any client/party (*client-level privacy*).

To illustrate the above requirements, suppose that several hospitals build a common model to predict the risk of death for their patients. A hospital certainly does not want other hospitals to learn the status of any of their patients (record privacy) and perhaps not even the average status of all their patients (client privacy).

Record-level privacy is a standard requirement used in the privacy literature, but it is weaker than client-level privacy. Indeed, client-level privacy requires to hide any information which is unique to a client including perhaps all its training data.

We aim at developing a solution that provides *client-level privacy*. For example, in the scenario of collaborating hospitals, we aim at protecting any information that is unique to each single hospital's training data. The adversary should not be able to infer from the received model or its updates whether any client's data is involved in the federated run (up to ε and δ). We believe that this adversarial model is reasonable in many practical applications when the confidential information spans over multiple samples in the training data of a single client (e.g., the presence of a group a samples, such as people from a certain race). Differential Privacy guarantees plausible deniability not only to any groups of samples of a client but also to any client in the federated run. Therefore, any negative privacy impact on a party (or its training samples) cannot be attributed to their involvement in the protocol run.

3.2. DPFed-post: Differentially Private Federated Learning with Post-Processing

3.2.1. Challenge

Although standard differentially-private federated learning (DPFed) generates a model which protects the dataset of each participant with a theoretical privacy guarantee (bounded by ϵ), the utility of the model is generally very bad, due to the large amount of noise required by DP for a reasonable ϵ budget. Indeed, this noise prevents generally the convergence of the model or highly decreases its performances.

Recent works show that it is possible to improve the utility of a differentially private model by: reducing the sensitivity of the model's update (S) and/or increasing the value to noise level (Kerkouche et al., 2021a,b) or

even by local adaptation Yu et al. (2020). Note that, the sensitivity S is not the only DP parameter which governs the noise. Indeed, reducing the sampling probability $C = K/N$ is also one manner to reduce the noise for a given ϵ budget. However, in certain scenarios as ours when we have only few participants (small N) with small datasets, the convergence of the model before using DP can be possible only with a large sampling probability (C is set to 0.5 in our case). Therefore, noise might be very large in such cases, e.g. the noise multiplier σ is set to $\sigma = 2, 3$ which result in $\epsilon = 5.4, 8.9$, respectively.

3.2.2. Our solution

To tackle the well-known "utility-privacy tradeoff" under challenging settings, we propose a straightforward approach called **DPFed-post**. Our scheme outperforms the standard differentially private federated learning scheme (**DPFed**) by adding one normalization step.

Indeed, in both **DPFed-post** and **DPFed**, at each round r the server sends the global model $\mathbf{w}^{r-1}$ to K clients selected randomly, each selected client trains and sends back its model to the server. The server clips the update of each client $\Delta\mathbf{w}_k^r$ to have a L_2-norm at most S. After that, the server sums up the clipped updates $\Delta\hat{\mathbf{w}}_k^r$, adds the Gaussian Noise and average it to obtain the noisy update $\Delta\hat{\mathbf{w}}^r$.

The difference between **DPFed-post** and **DPFed** starts from this step. Instead of directly updating the global model with the averaged noisy update as it is the case in **DPFed**, in our scheme **DPFed-post**, $\Delta\mathbf{w}^r$ is normalized to obtain $\Delta\hat{\mathbf{w}}^r$ with L_2-norm at most P. Finally, the normalized noisy update $\Delta\hat{\mathbf{w}}^r$ is used to update the new global model $\mathbf{w}^r$ (See Alg. 1 for more details).

This normalization acts exactly as decreasing the learning rate. Indeed, the large additive noise generally prevents the convergence to a good local minimum. Therefore, this normalization will slow down the learning process in order to reach better performances compared to the standard scheme **DPFed**.

3.2.3. Privacy analysis:

The adversary can only access the noisy model which is sufficiently perturbed to ensure client-level differential privacy; any client-specific information that could be inferred from the model is tracked and quantified by the moments accountant, described in Section 2.3, as follows.

Let $\eta_0(x|\xi) = \mathsf{pdf}_{\mathcal{G}(0,\xi)}(x)$ and $\eta_1(x|\xi) = (1-C)\mathsf{pdf}_{\mathcal{G}(0,\xi)}(x) + C\mathsf{pdf}_{\mathcal{G}(1,\xi)}(x)$ where C is the sampling probability of a single client in a single round. Let $\alpha(\lambda|C) = \log\max(E_1(\lambda,\xi,C), E_2(\lambda,\xi,C))$ where $E_1(\lambda,\xi,C) = \int_{\mathbb{R}} \eta_0(x|\xi,C)\cdot\left(\frac{\eta_0(x|\xi,C)}{\eta_1(x|\xi,C)}\right)^{\lambda} dx$ and $E_2(\lambda,\xi,C) = \int_{\mathbb{R}} \eta_1(x|\xi,C)\cdot\left(\frac{\eta_1(x|\xi,C)}{\eta_0(x|\xi,C)}\right)^{\lambda} dx$.

Theorem 4 (Privacy of DPFed-post) *DPFed-post is* $(\min_\lambda(T_{\mathsf{cl}}\cdot\alpha(\lambda|C) - \log\delta)/\lambda, \delta)$*-DP.*

Given a fixed value of δ, ε is computed numerically as in Abadi et al. (2016); Mironov et al. (2019).

4. Experimental Results

4.1. Neural Networks for Survival Analysis

In this paper, we utilize 4 deep neural networks which are designed for survival analysis. As explained in Section. 2.2, by defining appropriate loss functions for neural networks, they could be used to serve specific purposes:

CoxPH: adopted from Kvamme et al. (2019), also known as DeepSurv (Katzman et al., 2018) replaces $g(\mathbf{x})$ of Equation. 5 with a fully connected deep neural network. The loss for this model is defined as:

$$\text{Loss} = \frac{1}{n}\sum_{i:E_i=1}\log\left(\sum_{j\in R_i}\exp[g(\mathbf{x}_j) - g(\mathbf{x}_i)]\right) \tag{10}$$

CoxCC: adopted from Kvamme et al. (2019), Similar to CoxPH except for the fact that the risk set R_i of the loss function is approximated with a large enough set to make calculations easier.

CoxTime: adopted from Kvamme et al. (2019), similar model to CoxCC, but the proportionality assumption of the Cox hazard model is abondoed:

$$h(t|\mathbf{x}) \quad = \quad h_0(t)\exp[g(t,\mathbf{x})] \tag{11}$$

so that $g(t,\mathbf{x})$ uses the time t as another covariate.

DeepHit: adopted from Kvamme et al. (2019); Lee et al. (2018a), this model is fundamentally different from the previous 3 models and makes no assumption about the underlying relation between the covariates. It has a 2-part loss function; the first part measures the negative log-likelihood and the second part is a ranking loss which investigates each possible pair of data and tries to sort them with respect to their time of event. The output of all the previous models were continuous over time. However, the output of DeepHit is discretized and we need to define time-bins over the experiment span of our datasets.

4.2. Metrics

In this section we explain 3 different performance metrics, each offering a unique perspective into measuring how well the models predict on the data (for detailed explanation cf. Kvamme et al. (2019)).

4.2.1. Integrated Brier Score

Brier score is a performance measure for binary labels and takes into account both the discrimination as well as the

calibration of the model's predictions. To make our labels - which are times of events - binary, we can pick a fixed time t and binarize based on whether or not the event happens before or after this time t. So the generalized Brier score Graf et al. (1999) can be calculated as:

$$\text{BS}(t) = \frac{1}{N}\sum_{i=1}^{N}\Big[\frac{\hat{S}(t|\mathbf{x}_i)^2\mathbb{1}(T_i \le t, E_i = 1)}{\hat{G}(T_i)} + \frac{(1-\hat{S}(t|\mathbf{x}_i))^2\mathbb{1}(T_i > t)}{\hat{G}(t)}\Big] \tag{12}$$

where N is the number of data points and $\hat{G}(t)$ is the Kaplan-Meier estimate for censoring survival function $\text{P}(T > t)$ which helps to account also for the censored data in this metric. Note that $E_i = 1$ and $E_i = 0$ represent datapoints experiencing the event of interest and censoring, respectively.

To further extend the $\text{BS}(t)$ to all the possible time values t, we can integrate over time:

$$\text{Integrated BS} = \text{IBS} = \frac{1}{t_2 - t_1}\int_{t_1}^{t_2}\text{BS}(s)ds \tag{13}$$

which can be approximated numerically by a time grid over the test set. Note that a **lower value** of the Brier score or integrated Brier score indicates better performance of the model.

4.2.2. Integrated Binomial Log-Likelihood

This is also another binary classification metric that measures discrimination and calibration. In the same fashion as Brier score, we can binarized the labels and define the mean binomial log likelihood as:

$$\text{BLL}(t) = \frac{1}{N}\sum_{i=1}^{N}\Big[\frac{\log(1-\hat{S}(t|\mathbf{x}_i))\mathbb{1}(T_i \le t, E_i = 1)}{\hat{G}(T_i)} + \frac{\log(\hat{S}(t|\mathbf{x}_i))\mathbb{1}(T_i > t)}{\hat{G}(t)}\Big] \tag{14}$$

We can again integrate over different times in the same way as 13:

$$\text{Integrated BLL} = \text{IBLL} = \frac{1}{t_2 - t_1}\int_{t_1}^{t_2}\text{BLL}(s)ds \tag{15}$$

A higher value of the integrated binomial log-likelihood indicates better performance of the model. To avoid reporting negative values, in our experiments we calculate the negative integrated binomial log-likelihood, so here, **lower values are preferred**.

4.2.3. Concordance Index

Concordance index (C-index) Harrell et al. (1982) is one of the most commonly-used metrics in the field of survival analysis. The concordance is only concerned with the ordering of the pairs of data points in the data set, regardless of the true time of the events.
The time-dependent C-index, for a pair of data points i and j can be estimated by:

$$\begin{aligned}\text{C-index} &= \text{P}\{\hat{F}(T_i|\mathbf{x}_i) > \hat{F}(T_i|\mathbf{x}_j)|T_i < T_j, E_i = 1\} \\ &\approx \frac{\sum_{i\neq j} A_{i,j}\mathbb{1}(\hat{F}(T_i|\mathbf{x}_i) > \hat{F}(T_i|\mathbf{x}_j))}{\sum_{i\neq j} A_{i,j}}\end{aligned} \tag{16}$$

where $\hat{F}(t|x)$ is the estimated CIF and $A_{i,j}$ is a sorting function defined as $A_{i,j} = \mathbb{1}(T_i < T_j, E_i = 1)$. The idea is that when comparing two data points, if one experiences the event sooner than the other, at that event time, it should have a lower survival probability than the other data point. A C-index of 0.5 indicates chance-level prediction and **higher values indicate better performance**.

4.3. Datasets

For our experiments, we choose three real-world medical datasets which are collected over multiple years:

Rotterdam and German Breast Cancer Study Group (GBSG): Contains data of 2232 breast cancer patients from the Rotterdam tumor bank Foekens et al. (2000) and the German Breast Cancer Study Group (GBSG) Schumacher et al. (1994). The data is pre-processed similar to Katzman et al. (2018) and contains 7 features and the maximum survival duration is 87 months.

The Molecular Taxonomy of Breast Cancer International Consortium (METABRIC): This dataset contains gene and protein expressions of 1904 individuals Curtis et al. (2012). We use a dataset prepared similar to Katzman et al. (2018). This pre-processed dataset contains 4 gene indicators (MKI67, EGFR, PGR, and ERBB2) and 5 clinical features (hormone treatment indicator, radiotherapy indicator, chemotherapy indicator, ER-positive indicator, age at diagnosis) for each patient and is followed for a maximum of 355 months.

Study to Understand Prognoses Preferences Outcomes and Risks of Treatment (SUPPORT): This dataset consists of 8873 seriously-ill adults Knaus et al. (1995). The dataset has 14 features (age, sex, race, number of comorbidities, presence of diabetes, presence of dementia, presence of cancer, mean arterial blood pressure, heart rate, respiration rate, temperature, white blood cell count, serum's sodium, and serum's creatinine) with a maximum survival time of 67 months. We use a pre-processed version according to Katzman et al. (2018).

Data in practice: For all our experiments, we assume that 10 hospitals/data centers would like to collaborate to

train a model. This is a reasonable estimate of the number of health centers that would like to study a specific disease, as seen e.g. in Tomczak et al. (2015); Murphy et al. (2000); Linge et al. (2016). Furthermore, in practice, each hospital would have access to only a few hundred data (e.g. Mohan et al., 1996; Györffy et al., 2010; Chi et al., 2007). In Table 1, we show for each dataset how many patients were censorded, and given that the data is split between 10 hospital, how many patients each hospital would have.

4.4. Setup

In this section, we give an overview of the technical details of our experiments in both centralized and federated setting. As mentioned in Section 4.1, the output of all the models except DeepHit are continuous. For this reason we define time-bins with duration of ~ 12 months for all of our datasets when deploying DeepHit. We use pycox package [2] to construct all our models.

Centralized: To have a baseline to compare the effect of federation of the models, we first need to study their performance in a centralized setting. Datasets are randomly split into 80% training and 20% test data. To account for the effect of this random splitting, we run each experiment 5 times and report the average value and standard deviation. For all our experiments, the structure of all models are in the form of $[\text{input}, 32, 32, \text{output}]$ fully-connected nodes. We use AdamOptimizer with a learning rate of 10^{-4} to train these models and use an early stopping criterion to achieve the best performance.

Federated: For all of the federated experiments, we again randomly split the data into 80% training and 20% test set. The training set is then randomly split into $N = 10$ equal-sized parts to be assigned to each client. Here, we also run the experiments 5 times with different random splittings and report the average performance and the standard deviation.

The same neural network structure and optimizer as the centralized setting is used by clients. At each round of communication, the selected clients train their model for 50 local epochs. The training is done for a total of $T_{cl} = 50$ communication rounds.

We start our experiments in a centralized settings. In the second stage, we study the effect of standard federation (StdFed, Algorithm. 2 in Appendix A) with no privacy protocol, on the performance of the model. We then apply client-level differential privacy (DPFed, Algorithm. 3 in Appendix A) to this federated scheme. Finally, we utilize our post-processing solution (DPFed-post, Algorithm. 1) on top of the client-level DP federated learning pipeline.

Centralized to StdFed: The first step of our experiments is to study the effect of federation of deep survival models over 10 hospitals. To pick a suitable client sampling probability C, we ran our experiments for $C = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$ and chose the lowest value that results in a comparable performance of the federated model after 50 rounds and the centralized setting. We pick the lowest acceptable C since later when DP is applied, the lower value of client sampling probability would lead to better privacy guarantees.

StdFed to DPFed: Many hyper-parameters play a role when applying DP on federated learning algorithm. The parameter update clipping threshold S, the Gaussian noise multiplier σ, the client sampling probability C, total number of communication rounds T_{cl}, and δ effect the performance of the model as well as the calculated privacy budget ϵ (see Theorem. 4 and Algorithm 1).

The parameter δ is usually set to the inverse of the size of the population that the privacy mechanism is applied on, here 10 clients. However, $\delta = 10^{-1}$, which represents the probability of DP being broken, is too large for any tight theoretical guarantee. Therefore, we set $\delta = 10^{-3}$ for all our calculations.

We also fix the number of communication rounds to $T_{cl} = 50$ since in the case of StdFed this seems to be enough for all the models and datasets to achieve comparable performance to the centralized setting.

When choosing the client sampling probability C, we face a privacy/utility trade-off: The higher the sampling probability, the better the utility of the final model and the higher the value of the privacy budget ϵ. We choose $C = 0.5$ as it is the lowest value that results in comparable utility between the StdFed and centralized models.

The sensitivity S is chosen during the initialization period in collaboration with all the participants. We pick the median value of the clients updates as our clipping threshold S.

The noise multiplier σ has a significant impact on the performance of the models as well the calculated privacy budget ϵ and there is a trade-off between these two: the tighter the privacy guarantee, the worse the utility of the model. Hence we are looking for a reasonable operation point where there is a balance between the value of ϵ and utility. We pick two different values of $\sigma = 2, 3$ which result in $\epsilon = 5.4, 8.9$, respectively [3].

DPFed to DPFed-post: When applying the post-processing step on the client-level DP (DPFed-post Algorithm 1), the value of parameter P also needs to be fixed. This parameter represents the clipping threshold for the average update vector. To find a suitable P value, we examined $P = \{1S, 2S, 3S, 4S, 5S\}$ where S is the parameter update clipping threshold from the previous step. A choice of $2S$ or $3S$ gave consistent improvement across all investigated models and datsets.

2. https://github.com/havakv/pycox

3. Given a fixed value of δ, ε is computed numerically as in Abadi et al. (2016); Mironov et al. (2019). (see Section 4 for more details.

Table 1: Details of each Datasets

Dataset	# features	#uncensored	#censored	#hospitals	# training data/hospital
GBSG	7	1272(57%)	960(43%)	10	178
METABRIC	9	1103(58%)	801(42%)	10	152
SUPPORT	14	6034(68%)	2839(32%)	10	709

Table 2: GBSG dataset

Metric	Model	Non-Private		$(\epsilon = 5.4, \delta = 10^{-3})$		$(\epsilon = 8.9, \delta = 10^{-3})$	
		Centralized	StdFed	DPFed	$\text{DPFed}_{\text{post}}$	DPFed	$\text{DPFed}_{\text{post}}$
C-index ↑	DeepHit	0.66 ± 0.02	0.67 ± 0.02	0.47 ± 0.03	$\mathbf{0.56 \pm 0.04}$	0.54 ± 0.03	$\mathbf{0.59 \pm 0.03}$
	CoxPH	0.66 ± 0.01	0.67 ± 0.03	0.45 ± 0.70	$\mathbf{0.62 \pm 0.02}$	0.47 ± 0.05	$\mathbf{0.64 \pm 0.03}$
	CoxCC	0.63 ± 0.02	0.68 ± 0.01	0.58 ± 0.05	$\mathbf{0.61 \pm 0.02}$	0.62 ± 0.02	$\mathbf{0.64 \pm 0.03}$
	CoxTime	0.64 ± 0.01	0.67 ± 0.01	0.57 ± 0.07	$\mathbf{0.62 \pm 0.02}$	0.61 ± 0.03	$\mathbf{0.63 \pm 0.02}$
IBS ↓	DeepHit	0.18 ± 0.01	0.21 ± 0.01	0.29 ± 0.05	$\mathbf{0.20 \pm 0.01}$	0.21 ± 0.01	$\mathbf{0.19 \pm 0.01}$
	CoxPH	0.18 ± 0.01	0.18 ± 0.01	0.36 ± 0.08	$\mathbf{0.20 \pm 0.01}$	0.29 ± 0.08	$\mathbf{0.19 \pm 0.01}$
	CoxCC	0.19 ± 0.01	0.18 ± 0.01	0.30 ± 0.06	$\mathbf{0.19 \pm 0.01}$	0.20 ± 0.01	$\mathbf{0.18 \pm 0.01}$
	CoxTime	0.18 ± 0.01	0.18 ± 0.01	0.26 ± 0.05	$\mathbf{0.19 \pm 0.01}$	0.20 ± 0.01	$\mathbf{0.19 \pm 0.01}$
NIBLL ↓	DeepHit	0.54 ± 0.01	0.62 ± 0.03	1.73 ± 0.80	$\mathbf{0.57 \pm 0.01}$	0.63 ± 0.06	$\mathbf{0.56 \pm 0.01}$
	CoxPH	0.53 ± 0.01	0.53 ± 0.01	1.70 ± 0.90	$\mathbf{0.56 \pm 0.02}$	1.51 ± 0.90	$\mathbf{0.55 \pm 0.03}$
	CoxCC	0.56 ± 0.02	0.52 ± 0.01	1.69 ± 0.90	$\mathbf{0.56 \pm 0.01}$	0.59 ± 0.03	$\mathbf{0.54 \pm 0.01}$
	CoxTime	0.54 ± 0.01	0.52 ± 0.01	0.87 ± 0.22	$\mathbf{0.56 \pm 0.02}$	0.57 ± 0.01	$\mathbf{0.55 \pm 0.01}$

4.5. Observations

The results per dataset are shown in Tables .2 (and .4, .5 in Appendix B). The Non-Private column contains the results for centralized setting and federated training of the model using the standard FedAvg Algorithm. 2. The two other columns each compare the results when regular differentially-private federated learning is used to when our post-processing technique is applied, for each privacy regime. The standard deviations are reported over 5 random splits of the data as explained in Section. 4.4. The arrows next to metrics indicate if a lower or a higher value is desired.

The first observation is that the average performance of standard federated training of these survival models is always at a comparable level to the centralized setting. This is an important finding, showing that with federation of small datasets, over only 50 rounds of communication, acceptable utilities can be achieved.

Investigating the effect of post-processing technique: to be able to assess how well our proposed technique works, for each privacy regime we compare the performance of DPFed-post with DPFed. The bold numbers in the tables indicate a better relative utility for our method. We observe that for most of the models and metrics and datasets our technique results in an improved performance. In particular for GBSG dataset, we always see an improvement. From Table. 1 we see that GBSG has the second most number of data points and the least number of features. So this finding might be related to the fact that it is easier for models to converge for this dataset and post-processing always helps in leading the gradient in the right direction.

To be able to measure the change in performance quantitatively, we define

$$\Delta_{A \to B} = \pm \frac{[\text{metric}(B) - \text{metric}(A)]}{\text{metric}(A)} \tag{17}$$

which measures the improvement of strategy B w.r.t strategy A. For IBS and NIBLL a negative value of $[\text{metric}(B) - \text{metric}(A)]$ and for C-index a positive value of this parameter shows improvement. So we also multiply by -1 when calculating for IBS and NIBLL.

The average values of $\Delta_{\text{DPFed} \to \text{DPFed-post}}$ for each ϵ value are shown in Table. 3. We calculate Δ according to Equation. 17 for each dataset and metric and model and report the average over all of them. For both values of privacy budget, we observe an average improvement of performance when the post-processing step is applied: for $\epsilon = 5.4$ post-processing results in 17% improvement and for $\epsilon = 8.9$ the average performance is improved by 12%. Post-processing helps more for the case of $\epsilon = 5.4$ and this is expected since at lower ϵ values more DP noise is applied to the model and the convergence of the model with no regularization of the learning rate (via post-processing) becomes difficult and the final model shows noisy behavior.

To study the effect of post-processing on reducing the standard deviations, we calculate the average relative standard deviation rstd = (std/mean) across all datasets and models and all metrics. The change of this value

from DPFed to DPFed-post is shown in the second row of Table. 3. For both values of ϵ, post-processing helps to reduce the average relative standard deviation. We observe that rstd_{Avg} for DPFed is much larger for $\epsilon = 5.4$ compared to $\epsilon = 8.9$. This is expected since, the noise added to the model is larger in the case of $\epsilon = 5.4$ and this results in the final model showing different behaviors after each round of running the experiments. Interestingly, rstd_{Avg} retains a value of 0.05 for both ϵ regimes. This indicates that our method is successful in suppressing large deviations of the model even for higher noise values.

5. Related Work

Cox proportional hazards model Cox (1972) is one of the earliest and most powerful models for survival analysis. However, it makes very restrictive assumptions on the the data, such as linear dependence of the covariates and proportionality of the hazard rate over time. There has been some suggested modifications to this formulations, such as time-dependent variables Andersen and Gill (1982); Fisher and Lin (1999) or assuming a Wiener process Lee and Whitmore (2010).

Faraggi and Simon (1995) were the first to use simple perceptron model for Cox regression. In 2018 Katzman et al. (2018) proposed DeepSurv, a deep neural network for Cox regression, and showed that their model outperforms a simple Cox proportional hazards model. Kvamme et al. (2019) proposed CoxTime, a modification of DeepSurv, with no proportionality assumption on the data. In a slightly different direction, Lee et al. (2018a) used a deep neural network with no assumption on the data to learn the survival functions.

Due to the high sensitivity of the medical data which are used for the survival analysis, hospitals are often reluctant to share and centralize them. Fortunately, Federated Learning allows different clients to train collaboratively a common model without sharing their data. Federated learning first proposed by Google McMahan et al. (2017); Konečnỳ et al. (2016) has recently become very popular as a method of distributed training of machine learning models. It has been used for mobile keyboard prediction Hard et al. (2018); Lim et al. (2020), financial fraud detection Suzumura et al. (2019); Yang et al. (2019) and in healthcare for, e.g. patient similarity learning Lee et al. (2018b) and patient representation learning Liu et al. (2019); Kim et al. (2017).

Although, Federated Learning is more privacy-preserving compared to its centralized counterpart, different attacks show that the adversary can still infer sensitive information by only using the model's parameters/updates. Indeed, Nasr et al. (2019); Melis et al. (2019) define a membership attack that allows to infer if a particular sample is included in the dataset of a specific client. Similarly, Melis et al. (2019) define an attack which aims at inferring if a group of people with a specific property, like for example skin color or ethnicity, is included in the dataset of a particular participating entity. Finally, the strongest attack is called reconstruction attack Zhu et al. (2019); Zhao et al. (2020); Geiping et al. (2020) presented some attacks that allow an adversary to reconstruct some training data samples of some entities. A solution to prevent these attacks and provide theoretical guarantees in to use a privacy model called Differential Privacy (Dwork and Roth, 2014).

Kerkouche et al. (2021c) uses record-level differential privacy to protect the federated training of medical data. In Kerkouche et al. (2021b,a) client-level differential privacy has been used to protect medical data of each hospital in a federated setting. In their work, they are concerned with a binary classifier that predicts if a patient would die. They address the problem of instability of training after adding noise by reducing the sensitivity S and increasing the value-to-noise levels, either by using Compressive Sensing Kerkouche et al. (2021a) or by constraining the model Kerkouche et al. (2021b).

As far as we know, our work is the first which investigates the utility-privacy tradeoff introduced by DP under such challenging but realistic settings for the privacy analysis. Indeed, we consider a scenario with a total of only 10 hospitals and where half of them participate at each round. This means that our sampling probability C is set to 0.5 and the additive noise required by DP is also very large.

6. Conclusion

In this work, we tackle the challenge of using small datasets to train a differentially-private model in a federated setting. This becomes relevant when the collection of data is time-consuming and expensive, and only a few specialized data holders are interested in training a model. We propose adding a post-processing step to the popular client-level differentially private federated learning scheme. Our results indicate that this technique - which we refer to as DPFed-post - consistently improves the performances of the models and reduces the disparity in its behavior.

Institutional Review Board (IRB)

This research is based on publicly available datasets and does not require IRB approval.

Table 3: Quantitative comparison of the performance of DPFed and DPFed-post. The values are averaged over all datasets and all metrics and all models.

	$(\epsilon = 5.4, \delta = 10^{-3})$ DPFed→DPFed-post	$(\epsilon = 8.9, \delta = 10^{-3})$ DPFed → DPFed-post
Δ_{Avg}	0.17	0.12
rstd_{Avg}	0.19 → 0.05	0.07 → 0.05

7. Acknowledgments

This work is partially funded by the Helmholtz Association within the project "Trustworthy Federated Data Analytics" (TFDA) (funding number ZT-I-OO1 4).

References

Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *ACM CCS*, 2016.

Per Kragh Andersen and Richard D Gill. Cox's regression model for counting processes: a large sample study. *The annals of statistics*, pages 1100–1120, 1982.

Sean Augenstein, H Brendan McMahan, Daniel Ramage, Swaroop Ramaswamy, Peter Kairouz, Mingqing Chen, Rajiv Mathews, et al. Generative models for effective ml on private, decentralized datasets. *arXiv preprint arXiv:1911.06679*, 2019.

Bart Baesens, Tony Van Gestel, Maria Stepanova, Dirk Van den Poel, and Jan Vanthienen. Neural network survival analysis for personal loan data. *Journal of the Operational Research Society*, 56(9):1089–1098, 2005.

Hermann Brenner. Long-term survival rates of cancer patients achieved by the end of the 20th century: a period analysis. *The Lancet*, 360(9340):1131–1135, 2002.

Liberato Camilleri. History of survival analysis, 2019. URL https://www.um.edu.mt/library/oar/handle/123456789/55748.

Chih-Lin Chi, W Nick Street, and William H Wolberg. Application of artificial neural network-based survival analysis on two breast cancer datasets. In *AMIA Annual Symposium Proceedings*, volume 2007, page 130. American Medical Informatics Association, 2007.

David R Cox. Regression models and life-tables. *Journal of the Royal Statistical Society: Series B (Methodological)*, 34(2):187–202, 1972.

Christina Curtis, Sohrab P Shah, Suet-Feung Chin, Gulisa Turashvili, Oscar M Rueda, Mark J Dunning, Doug Speed, Andy G Lynch, Shamith Samarajiwa, Yinyin Yuan, et al. The genomic and transcriptomic architecture of 2,000 breast tumours reveals novel subgroups. *Nature*, 486(7403):346–352, 2012.

Lore Dirick, Gerda Claeskens, and Bart Baesens. Time to default in credit scoring using survival analysis: a benchmark study. *Journal of the Operational Research Society*, 68(6):652–665, 2017.

Cynthia Dwork and Aaron Roth. The Algorithmic Foundations of Differential Privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3–4), 2014.

David Faraggi and Richard Simon. A neural network model for survival data. *Statistics in medicine*, 14(1): 73–82, 1995.

Lloyd D Fisher and Danyu Y Lin. Time-dependent covariates in the cox proportional-hazards regression model. *Annual review of public health*, 20(1):145–157, 1999.

John A Foekens, Harry A Peters, Maxime P Look, Henk Portengen, Manfred Schmitt, Michael D Kramer, Nils Brünner, Fritz Jänicke, Marion E Meijer-van Gelder, Sonja C Henzen-Logmans, et al. The urokinase system of plasminogen activation and prognosis in 2780 breast cancer patients. *Cancer research*, 60(3):636–643, 2000.

Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. Inverting gradients-how easy is it to break privacy in federated learning? *Advances in Neural Information Processing Systems*, 33:16937–16947, 2020.

Robin C Geyer, Tassilo Klein, and Moin Nabi. Differentially private federated learning: A client level perspective. *arXiv preprint arXiv:1712.07557*, 2017.

ARRPA Goldhirsch, Richard D Gelber, R John Simes, Paul Glasziou, and Alan S Coates. Costs and benefits of adjuvant therapy in breast cancer: a quality-adjusted survival analysis. *Journal of Clinical Oncology*, 7(1): 36–44, 1989.

Erika Graf, Claudia Schmoor, Willi Sauerbrei, and Martin Schumacher. Assessment and comparison of prognostic classification schemes for survival data. *Statistics in medicine*, 18(17-18):2529–2545, 1999.

Balazs Győrffy, Andras Lanczky, Aron C Eklund, Carsten Denkert, Jan Budczies, Qiyuan Li, and Zoltan Szallasi. An online survival analysis tool to rapidly assess the effect of 22,277 genes on breast cancer prognosis using microarray data of 1,809 patients. *Breast cancer research and treatment*, 123(3):725–731, 2010.

Daniel Herbert Hanson. *Artificial neural network based survival analysis for hydro-mechanical systems*. The University of Iowa, 2004.

Andrew Hard, Kanishka Rao, Rajiv Mathews, Swaroop Ramaswamy, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. Federated learning for mobile keyboard prediction. *arXiv preprint arXiv:1811.03604*, 2018.

Frank E Harrell, Robert M Califf, David B Pryor, Kerry L Lee, and Robert A Rosati. Evaluating the yield of medical tests. *Jama*, 247(18):2543–2546, 1982.

Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019.

Jared L Katzman, Uri Shaham, Alexander Cloninger, Jonathan Bates, Tingting Jiang, and Yuval Kluger. Deepsurv: personalized treatment recommender system using a cox proportional hazards deep neural network. *BMC medical research methodology*, 18(1):1–12, 2018.

Raouf Kerkouche, Gergely Ács, Claude Castelluccia, and Pierre Genevès. Compression boosts differentially private federated learning. In *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 304–318. IEEE, 2021a.

Raouf Kerkouche, Gergely Ács, Claude Castelluccia, and Pierre Genevès. Constrained differentially private federated learning for low-bandwidth devices. In *Uncertainty in Artificial Intelligence*, pages 1756–1765. PMLR, 2021b.

Raouf Kerkouche, Gergely Acs, Claude Castelluccia, and Pierre Genevès. Privacy-preserving and bandwidth-efficient federated learning: an application to in-hospital mortality prediction. In *Proceedings of the Conference on Health, Inference, and Learning*, pages 25–35, 2021c.

Jack Kiefer and Jacob Wolfowitz. Stochastic estimation of the maximum of a regression function. *The Annals of Mathematical Statistics*, pages 462–466, 1952.

Yejin Kim, Jimeng Sun, Hwanjo Yu, and Xiaoqian Jiang. Federated tensor factorization for computational phenotyping. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 887–895, 2017.

William A Knaus, Frank E Harrell, Joanne Lynn, Lee Goldman, Russell S Phillips, Alfred F Connors, Neal V Dawson, William J Fulkerson, Robert M Califf, Norman Desbiens, et al. The support prognostic model: Objective estimates of survival for seriously ill hospitalized adults. *Annals of internal medicine*, 122(3): 191–203, 1995.

Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

Håvard Kvamme, Ørnulf Borgan, and Ida Scheel. Time-to-event prediction with neural networks and cox regression. *arXiv preprint arXiv:1907.00825*, 2019.

Erkki K Laitinen. Survival analysis and financial distress prediction: Finnish evidence. *Review of Accounting and Finance*, 2005.

Changhee Lee, William Zame, Jinsung Yoon, and Mihaela van der Schaar. Deephit: A deep learning approach to survival analysis with competing risks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018a.

Eunjo Lee, Yoonjae Jang, Du-Mim Yoon, Jihoon Jeon, Seong-il Yang, Sang-Kwang Lee, Dae-Wook Kim, Pei Pei Chen, Anna Guitart, Paul Bertens, África Periáñez, Fabian Hadiji, Marc Müller, Youngjun Joo, Jiyeon Lee, Inchon Hwang, and Kyung-Joong Kim. Game data mining competition on churn prediction and survival analysis using commercial game log data. *IEEE Transactions on Games*, 11(3):215–226, 2019. doi: 10.1109/TG.2018.2888863.

Junghye Lee, Jimeng Sun, Fei Wang, Shuang Wang, Chi-Hyuck Jun, and Xiaoqian Jiang. Privacy-preserving patient similarity learning in a federated environment: development and analysis. *JMIR medical informatics*, 6(2):e20, 2018b.

Mei-Ling Ting Lee and GA Whitmore. Proportional hazards and threshold regression: their theoretical and practical connections. *Lifetime data analysis*, 16(2): 196–214, 2010.

Wei Yang Bryan Lim, Nguyen Cong Luong, Dinh Thai Hoang, Yutao Jiao, Ying-Chang Liang, Qiang Yang, Dusit Niyato, and Chunyan Miao. Federated learning in mobile edge networks: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 22 (3):2031–2063, 2020.

Annett Linge, Fabian Lohaus, Steffen Löck, Alexander Nowak, Volker Gudziol, Chiara Valentini, Cläre von Neubeck, Martin Jütz, Inge Tinhofer, Volker Budach, et al. Hpv status, cancer stem cell marker expression, hypoxia gene signatures and tumour volume identify good prognosis subgroups in patients with hnscc after primary radiochemotherapy: A multicentre retrospective study of the german cancer consortium radiation oncology group (dktk-rog). *Radiotherapy and Oncology*, 121(3):364–373, 2016.

Dianbo Liu, Dmitriy Dligach, and Timothy Miller. Two-stage federated phenotyping and patient representation learning. In *Proceedings of the conference. Association for Computational Linguistics. Meeting*, volume 2019, page 283. NIH Public Access, 2019.

Junxiang Lu. Predicting customer churn in the telecommunications industry—-an application of survival analysis modeling using sas. In *SAS User Group International (SUGI27) Online Proceedings*, volume 114, 2002.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pages 1273–1282. PMLR, 2017.

Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Exploiting unintended feature leakage in collaborative learning. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 691–706. IEEE, 2019.

Ilya Mironov, Kunal Talwar, and Li Zhang. Renyi differential privacy of the sampled gaussian mechanism. *arXiv preprint arXiv:1908.10530*, 2019.

B Mishachandar and K Anil Kumar. Predicting customer churn using targeted proactive retention. *International Journal of Engineering & Technology*, 7(2.27):69, 2018.

Viswanathan Mohan, Gopal Premalatha, Audinarayanan Padma, Suresh T Chari, and Capecomorin S Pitchumoni. Fibrocalculous pancreatic diabetes: long-term survival analysis. *Diabetes Care*, 19(11):1274–1278, 1996.

Patricia Murphy, Barbara Kreling, Erica Kathryn, Marguerite Stevens, Joanne Lynn, and Jennie Dulac. Description of the support intervention. *Journal of the American Geriatrics Society*, 48(S1):S154–S161, 2000.

Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *IEEE Symposium on Security and Privacy*, 2019.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

M Schumacher, G Bastert, H Bojar, K Hübner, M Olschewski, W Sauerbrei, C Schmoor, C Beyerle, RL Neumann, and HF Rauschecker. Randomized 2 x 2 trial evaluating hormonal treatment and the duration of chemotherapy in node-positive breast cancer patients. german breast cancer study group. *Journal of Clinical Oncology*, 12(10):2086–2093, 1994.

Reza Shokri and Vitaly Shmatikov. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pages 1310–1321, 2015.

Maria Stepanova and Lyn Thomas. Survival analysis methods for personal loan data. *Operations Research*, 50(2):277–289, 2002.

Quentin Styc and Philippe Lagacherie. Predicting soil depth using survival analysis models. In *7. Global Workshop on Digital Soil Mapping*, 2016.

Toyotaro Suzumura, Yi Zhou, Natahalie Baracaldo, Guangnan Ye, Keith Houck, Ryo Kawahara, Ali Anwar, Lucia Larise Stavarache, Yuji Watanabe, Pablo Loyola, et al. Towards federated graph learning for collaborative financial crimes detection. *arXiv preprint arXiv:1909.12946*, 2019.

Katarzyna Tomczak, Patrycja Czerwińska, and Maciej Wiznerowicz. The cancer genome atlas (tcga): an immeasurable source of knowledge. *Contemporary oncology*, 19(1A):A68, 2015.

Stacey Truex, Nathalie Baracaldo, Ali Anwar, Thomas Steinke, Heiko Ludwig, Rui Zhang, and Yi Zhou. A hybrid approach to privacy-preserving federated learning. In *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security*, pages 1–11, 2019.

Stacey Truex, Ling Liu, Ka-Ho Chow, Mehmet Emre Gursoy, and Wenqi Wei. Ldp-fed: Federated learning with local differential privacy. In *Proceedings of the Third ACM International Workshop on Edge Systems, Analytics and Networking*, pages 61–66, 2020.

Wensi Yang, Yuhang Zhang, Kejiang Ye, Li Li, and Cheng-Zhong Xu. Ffd: a federated learning based method for credit card fraud detection. In *International Conference on Big Data*, pages 18–32. Springer, 2019.

Tao Yu, Eugene Bagdasaryan, and Vitaly Shmatikov. Salvaging federated learning by local adaptation. *arXiv preprint arXiv:2002.04758*, 2020.

Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. idlg: Improved deep leakage from gradients. *arXiv preprint arXiv:2001.02610*, 2020.

Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Appendix A. Algorithms

Algorithm 2
`Federated Learning (StdFed).`
N total clients, local mini-batch size B, local epochs E, communication rounds T_{cl} and learning rate η.

Initialize $\mathbf{w}_0$ and send the model to clients
for $r = 1, ...T_{cl}$
  Select K clients randomly
  for each selected client $k = 1, ..., K$
    $\mathbf{w}_k^r \leftarrow \text{ClientUpdate}(k, \mathbf{w}^{r-1})$
  $\mathbf{w}^r \leftarrow \mathbf{w}^{r-1} + \frac{\sum_{k=1}^{K}(\mathbf{w}_k^r - \mathbf{w}^{r-1})}{K}$
ClientUpdate$(k, \mathbf{w})$
  for client k
  for $i = 1, ..., E$
    for local batches b
      $\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla l(b; \mathbf{w})$
  return $\mathbf{w}$ to server

Algorithm 3
`Differentially Private Federated Learning with post processing (DPFed).`
N total clients, local mini-batch size B, local epochs E, communication rounds T_{cl}, learning rate η, sensitivity S and post-processing parameter P.

Initialize $\mathbf{w}_0$ and send the model to clients
for $r = 1, ...T_{cl}$
  Select K clients randomly
  for each selected client $k = 1, ..., K$
    $\mathbf{w}_k^r \leftarrow \text{ClientUpdate}(k, \mathbf{w}^{r-1})$
    $\Delta \mathbf{w}_k^r \leftarrow \mathbf{w}_k^r - \mathbf{w}^{r-1}$
    $\Delta \hat{\mathbf{w}}_k^r \leftarrow \Delta \mathbf{w}_k^r / \max\left(1, \frac{||\Delta \mathbf{w}_k^r||_2}{S}\right)$
  $\Delta \mathbf{w}^r \leftarrow \frac{\sum_{k=1}^{K} \Delta \hat{\mathbf{w}}_k^r + \mathcal{G}(0, S\sigma \mathbf{I})}{K}$
  $\mathbf{w}^r \leftarrow \mathbf{w}^{r-1} + \Delta \mathbf{w}^r$
ClientUpdate$(k, \mathbf{w})$
  for client k
  for $i = 1, ..., E$
    for local batches b
      $\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla l(b; \mathbf{w})$
  return $\mathbf{w}$ to server

Appendix B. Results

Results for METABRIC and SUPPORT datasets according to Section. 4.4 and Section. 4.5. The arrows indicate if a lower or a higher value of the metric indicates better utility of the model. To compare our proposed post-processing method with the vanilla DP federated learning (DPFed), we have made numbers bold only when our method offers an advantage.

Table 4: METABRIC dataset

Metric	Model	Non-Private		$(\epsilon = 5.4, \delta = 10^{-3})$		$(\epsilon = 8.9, \delta = 10^{-3})$	
		Centralized	StdFed	DPFed	$\text{DPFed}_{\text{post}}$	DPFed	$\text{DPFed}_{\text{post}}$
C-index ↑	DeepHit	0.68 ± 0.17	0.65 ± 0.02	0.51 ± 0.06	$\mathbf{0.52 \pm 0.03}$	0.53 ± 0.02	$\mathbf{0.57 \pm 0.04}$
	CoxPH	0.64 ± 0.12	0.65 ± 0.01	0.50 ± 0.08	$\mathbf{0.52 \pm 0.03}$	0.60 ± 0.02	$\mathbf{0.63 \pm 0.04}$
	CoxCC	0.62 ± 0.01	0.62 ± 0.02	0.50 ± 0.03	$\mathbf{0.51 \pm 0.07}$	0.53 ± 0.06	$\mathbf{0.55 \pm 0.08}$
	CoxTime	0.63 ± 0.12	0.64 ± 0.01	0.53 ± 0.05	0.51 ± 0.05	0.56 ± 0.01	0.55 ± 0.06
IBS ↓	DeepHit	0.16 ± 0.01	0.17 ± 0.01	0.21 ± 0.01	$\mathbf{0.18 \pm 0.01}$	0.21 ± 0.01	$\mathbf{0.18 \pm 0.01}$
	CoxPH	0.17 ± 0.01	0.18 ± 0.01	0.20 ± 0.01	$\mathbf{0.19 \pm 0.01}$	0.19 ± 0.01	$\mathbf{0.18 \pm 0.01}$
	CoxCC	0.17 ± 0.01	0.17 ± 0.01	0.19 ± 0.01	0.20 ± 0.01	0.19 ± 0.01	$\mathbf{0.18 \pm 0.01}$
	CoxTime	0.17 ± 0.01	0.17 ± 0.01	0.21 ± 0.01	$\mathbf{0.20 \pm 0.01}$	0.20 ± 0.01	$\mathbf{0.19 \pm 0.01}$
NIBLL ↓	DeepHit	0.50 ± 0.02	0.52 ± 0.01	0.67 ± 0.02	$\mathbf{0.54 \pm 0.01}$	0.62 ± 0.02	$\mathbf{0.55 \pm 0.02}$
	CoxPH	0.50 ± 0.02	0.48 ± 0.01	0.57 ± 0.03	0.58 ± 0.03	0.55 ± 0.01	0.55 ± 0.03
	CoxCC	0.51 ± 0.02	0.52 ± 0.01	0.57 ± 0.02	0.57 ± 0.03	0.56 ± 0.01	$\mathbf{0.55 \pm 0.03}$
	CoxTime	0.51 ± 0.01	0.51 ± 0.01	0.60 ± 0.05	$\mathbf{0.58 \pm 0.03}$	0.61 ± 0.02	$\mathbf{0.53 \pm 0.02}$

Table 5: SUPPORT dataset

Metric	Model	Non-Private		$(\epsilon = 5.4, \delta = 10^{-3})$		$(\epsilon = 8.9, \delta = 10^{-3})$	
		Centralized	StdFed	DPFed	$\text{DPFed}_{\text{post}}$	DPFed	$\text{DPFed}_{\text{post}}$
C-index ↑	DeepHit	0.61 ± 0.01	0.59 ± 0.02	0.47 ± 0.02	0.49 ± 0.01	0.49 ± 0.01	$\mathbf{0.51 \pm 0.01}$
	CoxPH	0.61 ± 0.01	0.61 ± 0.01	0.50 ± 0.02	$\mathbf{0.51 \pm 0.02}$	0.51 ± 0.03	$\mathbf{0.53 \pm 0.02}$
	CoxCC	0.58 ± 0.01	0.61 ± 0.02	0.51 ± 0.04	0.50 ± 0.02	0.53 ± 0.03	0.53 ± 0.01
	CoxTime	0.60 ± 0.13	0.59 ± 0.01	0.54 ± 0.01	0.51 ± 0.01	0.49 ± 0.01	$\mathbf{0.53 \pm 0.02}$
IBS ↓	DeepHit	0.19 ± 0.01	0.23 ± 0.02	0.31 ± 0.03	$\mathbf{0.25 \pm 0.02}$	0.26 ± 0.02	$\mathbf{0.25 \pm 0.01}$
	CoxPH	0.19 ± 0.01	0.19 ± 0.01	0.22 ± 0.01	$\mathbf{0.21 \pm 0.01}$	0.21 ± 0.01	0.21 ± 0.01
	CoxCC	0.20 ± 0.01	0.19 ± 0.01	0.22 ± 0.02	0.22 ± 0.01	0.21 ± 0.01	0.21 ± 0.01
	CoxTime	0.20 ± 0.01	0.20 ± 0.01	0.25 ± 0.03	$\mathbf{0.21 \pm 0.01}$	0.23 ± 0.01	$\mathbf{0.21 \pm 0.01}$
NIBLL ↓	DeepHit	0.57 ± 0.01	0.67 ± 0.06	0.97 ± 0.90	$\mathbf{0.70 \pm 0.04}$	0.75 ± 0.06	$\mathbf{0.73 \pm 0.03}$
	CoxPH	0.56 ± 0.01	0.56 ± 0.01	0.64 ± 0.03	$\mathbf{0.61 \pm 0.01}$	0.62 ± 0.03	$\mathbf{0.60 \pm 0.01}$
	CoxCC	0.58 ± 0.01	0.57 ± 0.01	0.63 ± 0.04	0.63 ± 0.01	0.60 ± 0.01	$\mathbf{0.59 \pm 0.01}$
	CoxTime	0.58 ± 0.02	0.58 ± 0.01	0.71 ± 0.08	$\mathbf{0.60 \pm 0.01}$	0.66 ± 0.03	$\mathbf{0.61 \pm 0.02}$