Privacy Preserving Time-Series Forecasting of User Health Data Streams

Sana Imtiaz*, Sonia-Florina Horchidan*, Zainab Abbas*, Muhammad Arsalan†,
Hassan Nazeer Chaudhry‡, Vladimir Vlassov*
*KTH Royal Institute of Technology, Stockholm, Sweden
†Otto-von-Guericke Universität Magdeburg, Magdeburg, Germany
‡DEIB, Politecnico di Milano, Milan, Italy
{sanaim,sfhor,zainabab,vladv}@kth.se, muhammad.arsalan@ovgu.de, hassannazeer.chaudhry@polimi.it

***Abstract*—Privacy preservation plays a vital role in health care applications as the requirements for privacy preservation are very strict in this domain. With the rapid increase in the amount, quality and detail of health data being gathered with smart devices, new mechanisms are required that can cope with the challenges of large scale and real-time processing requirements. Federated learning (FL) is one of the conventional approaches that facilitate the training of AI models without access to the raw data. However, recent studies have shown that FL alone does not guarantee sufficient privacy. Differential privacy (DP) is a well-known approach for privacy guarantees, however, because of the noise addition, DP needs to make a trade-off between privacy and accuracy. In this work, we design and implement an end-to-end pipeline using DP and FL for the first time in the context of health data streams. We propose a clustering mechanism to leverage the similarities between users to improve the prediction accuracy as well as significantly reduce the model training time. Depending on the dataset and features, our predictions are no more than 0.025% far off the ground-truth value with respect to the range of value. Moreover, our clustering mechanism brings a significant reduction in the training time, with up to 49% reduction in prediction accuracy error in the best case, as compared to training a single model on the entire dataset. Our proposed privacy preserving mechanism at best introduces a decrease of $\approx$ 2% in the prediction accuracy of the trained models. Furthermore, our proposed clustering mechanism reduces the prediction error even in highly noisy settings by as much as 38% as compared to using a single federated private model.**

***Index Terms*—Federated Learning, Differential Privacy, Streaming k-means, Generative Adversarial Networks**

I. Introduction

Advancements in cloud computing and the Internet of Things (IoT) paradigms inspired the creation of highly interconnected heterogeneous computing environments that are capable of generating and processing tremendous volumes of data. Consequently, the past decades witnessed a notable growth in the adoption of wearable smart devices, which further enabled the development of applications and services employing the collected data. For instance, numerous health applications are adopting cloud platforms for real-time health monitoring, achieving fitness goals, disease diagnostics, and medical data analysis leading to personalized medicine. Modern health trackers such as Fitbit are equipped with multiple sensors capable of recording complex health metrics like the caloric burn, sleep quality, or the wearer's activity level [1]. Besides, health-tracking mobile applications, such as MyFitnessPal, are freely accessible to users and can be paired with wearable devices to provide an intricate glimpse into user health [2]. Typically, health and fitness applications require processing enormous amounts of personal user data. In some cases, this data is handled by an untrusted third party for data analytics or machine learning (ML) based services. In light of this discussion, it can be stipulated that personal health information is not absolutely private. Moreover, the General Data Protection Regulation (GDPR) demands to enforce privacy preservation policies, particularly in the health care domain, as this data could be potentially maltreated.

Real-time or low-latency systems and services, such as wearable smart device services, continuously process an enormous influx of data streams. In the traditional health care domain, data anonymization and de-identification are usually employed as the privacy preservation practices. This is owing to their relatively lower complexity, although they lack effective privacy guarantees [3], [4]. An alternative approach utilizes synthetic but representative datasets for improved privacy guarantees [5], [6]. On the contrary, privacy preservation solutions based on cryptography, blockchains, and private compute units are often compute-intensive, and hence, not suitable for distributed low-latency environments [4]. On the other hand, decentralized privacy-preserving ML-based solutions like federated learning (FL) facilitate collaborative training of models without exposing raw data and can be acclimated to real-time environments [7]. Moreover, solutions based on differential privacy (DP) can be integrated into ML systems to provide strong privacy guarantees. However, these solutions may introduce computational and performance overheads in the system, such as accuracy loss and degraded quality of service. These overheads become critical in time-series forecasting, particularly in the health care domain, as a small error in the forecast may lead to dire consequences. Moreover, devices may experience connectivity issues in distributed environments, so the system necessitates catering to asynchronous learning and service provisioning. All these issues make privacy-preserving forecasting of health data streams a challenging problem. The investigation of privacy preservation trade-off for low-latency environments is a significant research problem, particularly

978-1-7281-6251-5/20/$31.00 ©2020 IEEE

in the context of health data streams, as they demand strong privacy guarantees.

This work proposes a novel technique for privacy preservation on real-time predictions. In particular, an end-to-end pipeline for time-series forecasting is implemented. Furthermore, the impact of applying several privacy preservation solutions on the application performance in terms of prediction accuracy and model training time is studied.

Our contributions can be summarized as follows:

- Design and implementation of an end-to-end pipeline for time-series forecasting of health data streams in a federated learning environment.
- Design and implementation of a clustering mechanism using streaming *k*-means algorithm and pattern matching.
- Integration of state-of-the-art privacy preservation solutions in the designed pipeline and evaluating their impact on the time-series forecasting.
- Collection and refinement of a real-world dataset from geographically distributed users.
- Creation of a privacy-preserving smart health care dataset employing Generative Adversarial Networks (GANs).

II. Background

A. Time-Series Forecasting

Time-series data: Time-series are sequences of observations ordered by some temporal information. Time-series analysis is done to extract meaningful trends in data. It has applications in numerous fields such as recommender systems, personalized shopping, or targeted advertising. Time-series forecasting is an active field of research since it plays a fundamental role in decision-making. For example, data collected from various traffic sensors can help predict traffic conditions to help the drivers avoid traffic jam in future [8].

In this work, we focus on forecasting and discuss how time-series clustering can improve the predictions. More specifically, we will exploit the similarities between time-series to build better models. We use a raw-data-based approach [9] for time-series clustering in our work as it is best suited to the requirements of our use case, as this approach discovers groups of similar time-series directly from raw data.

Distributed k-means algorithm: The k-means algorithm works by finding *k* centroids corresponding to *k* clusters such that the distance between each point and its closest center gets minimized. We will focus on the scalable variant of k-means for big datasets that parallelize the algorithm by distributing the computation to multiple workers and provide a good approximation of the solution. [10] tackles the problem by proposing a parallel k-means algorithm based on the Single Program Multiple Data (SPMD) model, using message-passing. Our implementation adapts the SPMD message passing model [10] to work on the streaming data.

Pattern matching: In time-series clustering, the pattern matching process is employed to discover groups of series that exhibit similar patterns. In the case of symbols sequences, the simplest method is to compare each symbol of the series at a given time in pairs. Techniques such as Hamming and Levenshtein distance are mostly used for measuring distance. We cluster the users' time-series to capture similar trends in user diets using Hamming distance.

B. Neural Networks for Time-Series Data

Long Short-Term Memory networks (LSTMs) [11] are well suited for learning order dependence in sequence prediction or classification problems including text [12] and speech recognition, anomaly detection, and time-series data forecasting [8].

The main idea of LSTMs is making the network decide which information is relevant and which information can be forgotten. Three special gates control which information is kept or forgotten at each step: the forget gate which decides which information should be thrown away, the input gate determines what new input information is useful and should be added to the state, and the output gate builds the output.

C. Federated Learning

Federated learning is a novel approach in distributed machine learning (ML) with two highly appealing characteristics: the gains in privacy and performance [13], [14].

FL mechanism learns from all participants' data without actually seeing it. The actors of the FL algorithm are the *clients* and a *central coordinator* (often called *server*). Each client holds a local dataset which contains only their data. The server shares a central model with all the clients. Then, each client improves the current model using information from their local data and sends the update back to the server. The server aggregates the updates from multiple users and an improved central model is created and shared with the clients. The process repeats as the clients' devices collect more data.

FL caters to a variety of features suited to distributed ML on mobile client devices, such as catering to non-independent and identically distributed data, unbalanced and massively distributed datasets, and high capability to function in scenarios with limited communication.

D. Differential Privacy

Differential privacy comes in many flavours. However, the most popular mathematical tool used to express it is as it follows [15]: given a randomized algorithm A, the set of all datasets D and D' that differ on at most one row (i.e. the data of one individual), and any subset $S \subseteq range(A)$,

$$Pr[A(D) \in S] \leq e^{\varepsilon} Pr[A(D') \in S] \tag{1}$$

The ε parameter is used to quantify the loss of privacy. As can be noted, absolute privacy is obtained when ε equals 0. Achieving higher levels of accuracy involves adding more noise to the data which leads to a decrease in the accuracy of the algorithm. Thus, decreasing the parameter ε means increasing the accuracy. A trade-off must be found between keeping the information private and achieving meaningful and accurate results. When an algorithm requires multiple additive noise mechanisms, the privacy guarantee follows from the basic composition theorem [16], [17] or from advanced composition theorems and their extensions [18]–[20].

III. User Clustering Using Stream Processing

Our end-to-end health forecasting pipeline is implemented in Apache Flink [21]. It has two main components: (1) the clustering mechanism, which includes the streaming k-means subsystem and the pattern matching subsystem, and (2) the FL system with privacy preservation mechanism.

Our proposed pipeline consumes a stream of time-series diet and health logs from several users and attempts to predict the next logs by leveraging the history of each individual and the similarities between users. Due to the volume and velocity of the ingested data, the system has to be highly scalable. Besides, the streaming nature of the problem imposes strict requirements in terms of latency, as the system should be able to provide real-time predictions. And most importantly, the proposed pipeline has to ensure a strict level of privacy preservation because we are dealing with sensitive data.

First, we cluster the users and use this information to build separate federated models for each group. In this way, we ensure that the prediction caters to individuals with unique dietary patterns and lifestyles, thus offering personalised forecasts. We introduce our two-step clustering mechanism, that is able to group multidimensional time-series data based on common characteristics.

Clustering mechanism: The users' meal logs are first clustered using the streaming k-means clustering algorithm. Each meal log of a day consists of breakfast, lunch and dinner. We assign the meals in the stream into three groups/clusters that represent breakfast, lunch and dinner. At the beginning of k-means clustering, first unique meals that appear in the stream are chosen as centroids for each meal group. We have three centroids to represent our three meal groups, i.e. breakfast, lunch and dinner. Each user's meal is mapped to the closest centroid value during processing. Once user meals are mapped to clusters for a period of 7 days, pattern matching is done on the clustered data. In pattern matching, the centroid IDs are considered as numbers of sequences. For example, for simplicity we have a pattern for each user for one day, user 1 has a pattern (1,2,1) and user 2 has a pattern (1,3,1). Here the numbers indicate the centroid ID of the group/cluster to which the meals (breakfast, dinner, lunch) belonged to. For pattern matching, Hamming distance is computed over the given sequences of centroid IDs or patterns. In the end, we get groups of users with closest meal patterns. These groups are then used to train a separate FL model for each group. This approach whilst simple and intuitive, comes with a drawback in terms of implementation, as it has to be performed by a central entity. However, this approach is not only efficient in terms of improvements in training performance but also beneficial in terms of privacy preservation. This is because the central coordinator stores patterns, which are essentially sequences of centroid IDs, and not raw data, and can not be exploited to extract a specific user from the system.

We now overview the privacy preserving techniques used in the proposed time-series forecasting pipeline.

IV. Privacy Preservation Techniques in Time-series Forecasting

Since stream processing requires low latency and real-time response, we select the techniques that ensure strong privacy guarantees with low performance overhead in terms of model training time and with a minimal loss in prediction accuracy.

A. Categorical Data Anonymization

We study the impact of sensitive Individually Identifiable Data (IID) on user clustering by using k-modes for categorical data clustering [22] and k-means for numerical data clustering. k-modes was used as the expert clustering mechanism to validate the correctness of our streaming k-means approach. We observed that our proposed approach can cluster similar users using non-private data. As this work strongly advocates user privacy, we removed all the IID from users' data except gender and location (country). The latter were retained only for the synthetic data generation with GANs, as both attributes have a major impact on diet patterns; and to observe the correctness of the clustering patterns. However, our proposed pipeline does not make use of any IID in any mechanism.

B. Differential Privacy for Federated Learning

TensorFlow-Federated (TFF) is used for our implementation [23], [24]. We employ two well-established mechanisms for DP: Gaussian and Laplacian mechanisms [7]. We use the former mechanism in the noisy learning approach and the latter mechanism in the data noising approach.

Noisy learning: Gaussian noise addition to the output of a function f of sensitivity S_f on database D is defined by:

$$M(D) \triangleq f(D) + N(0, S_f^2\sigma^2), \tag{2}$$

where $N(0, S_f^2\sigma^2)$ is the normal distribution with mean 0 and standard deviation $S_f\sigma$ [7].

TensorFlow Privacy mainly uses a differentially private version of stochastic gradient descent (DP-SGD) to modify the learned gradients. Models trained with DP-SGD provide provable DP guarantees for their input data. It uses two additional hyperparameters with the stochastic gradient descent optimizer: the `clip` and the `noise_multiplier`. The former is used to clip each gradient computed on each training point in a mini-batch. Then, random noise from a Gaussian distribution is sampled and added to the clipped gradients to make it statistically impossible to know whether or not a particular data point was included in the training dataset. A differentially private query `DPQuery` is responsible for clipping gradients computed by the optimizer, accumulating them, and returning their noisy average to the optimizer [25].

Noisy data: For this approach, we use the Laplacian differential privacy by adding noise directly to the aggregated data records. Traditionally, for Laplace mechanism, random noise is drawn from a Laplacian distribution with mean 0 and variance S_f/ε to achieve ε-differential privacy [26]. In this work, all the data points in an aggregated data record are individually noised as we pick random noise samples for each point from a $Lap(0, 1/\varepsilon)$ distribution.

V. Proposed End-to-end Private Learning Pipeline

This section presents the proposed pipeline and describes the interaction of each subsystem in the noisy learning and noisy data approaches for privacy preservation.

A. System Overview

First, raw data points are aggregated as individual user data records. As explained in Section II-C, central coordinator uses updates from the clients to improve a global model. The global model is described using a set of hyperparameters. We use Adam optimizer for both client and server, and Standard Federated Averaging algorithm [14] as the aggregation method. Moreover, a random sample of 10% users is used in each round of FL as this fraction achieves a good trade-off between model convergence and computational efficiency [14].

The records are forwarded to the clustering mechanism, where we use streaming k-means algorithm with pattern matching to find similar users as explained in Section III. Afterwards, the central coordinator in FL stores these cluster patterns, which are essentially the sequences of centroid IDs, and not the aggregated raw data records.

Based on the clustered patterns, the server maintains k federated models, where k is the number of groups. Figure 1 depicts the process in one communication round for a randomly sampled client. $client_p$ sends its updates (step 1). Then, the coordinator looks up $client_p$ and finds that this client belongs to cluster $k-1$ (step 2). The coordinator finds the model that corresponds to cluster $k-1$ ($model_{k-1}$) and updates it using the information received from $client_p$ (step 3). In the final step, the coordinator shares the updated model with the client.

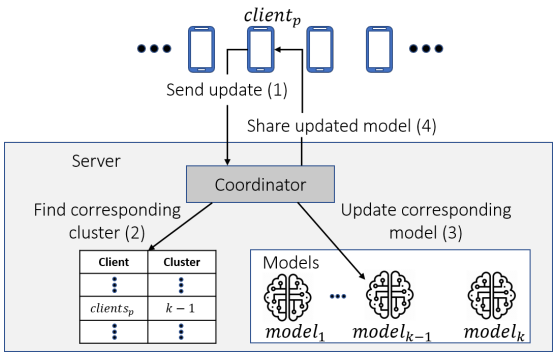


Fig. 1: An overview of the communication round in the FL process with clustering, assuming grouping into k clusters

B. Client Confidentiality with DP

We now explain the proposed pipeline in noisy learning and noisy data settings. Both approaches achieve the goal of noising the updates that are sent to the coordinator so that no sensitive information can be leaked by exchanging these updates.

Noisy learning: The weights are noised and sent to the federated aggregator with the mechanisms provided in TensorFlow Privacy. As depicted in Figure 2, each client adds noise to the true update, in both baseline and clustered scenarios and the amount of added noise is controlled by the *clip* and *noise multiplier* parameters, as explained in Section IV-B. The standard deviation of the added noise is computed by multiplying these two parameters. Moreover, aggregated user records are also sent separately to the clustering mechanism, as can be seen in Figure 2.

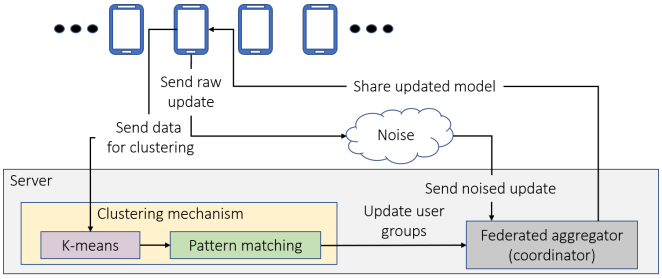


Fig. 2: Noisy learning: clustered FL using streaming k-means. Baseline model is traditional FL setup (not shown separately)

Noisy data: In this setup, the FL process remains very similar to the standard process, with the small modification that the clients noise their local datasets before trying to improve the federated model. Thus, the weights of each model sent as updates to the coordinator implicitly contain noise. Figures 3 and 4 show a high level image of the process. It should also be noted that the clustering mechanism receives noised aggregated data records, meaning that the quality of clustering will be affected. It should be noted that the prediction is performed locally and on clean data.

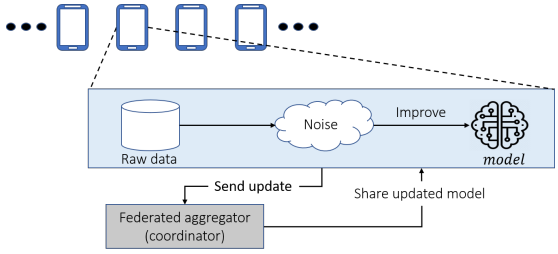


Fig. 3: Noisy data: a) Baseline FL

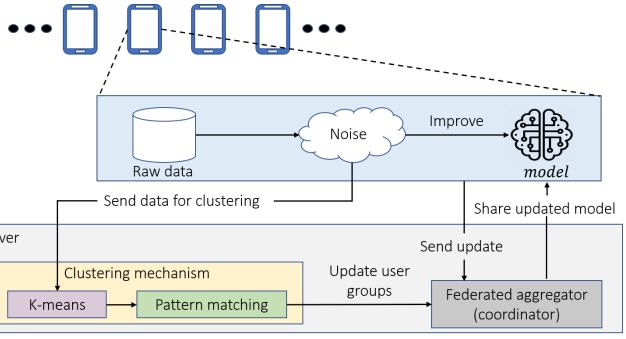


Fig. 4: Noisy data: b) Clustered FL using streaming k-means

Researchers state that an epsilon higher than 1 does not give good privacy protection in general. However, Apple MacOS's

DP has an epsilon as large as 6 and Google's version of DP claims to achieve an epsilon value of 2 in certain scenarios [27]. We will use these values as a guideline in our study and strive to achieve a compromise between the achieved level of privacy and the performance of the system.

VI. Experiments and Results

This section presents the impact of introducing our clustering mechanism and DP techniques on the system performance in terms of model training time and prediction accuracy.

A. Datasets

We used 3 datasets for our evaluation: MyFitnessPal [28], collected Fitbit dataset and Fitbit-GAN dataset. An overview of the datasets in terms of scale is shown in Table I.

TABLE I: Datasets used for evaluation

Dataset	# of users	# of days	# of raw records	Size
MyFitnessPal	9.9K	207	1.9M	2.1GB
Fitbit	25	60	≈17M	3.2GB
Fitbit-GAN	630	60	≈435M	≈ 83GB

MyFitnessPal: MyFitnessPal [2] contains records from 9900 users who logged their foods for almost 207 days. Each entry contains an anonymized user ID, logging date, name of the food, and respective macronutrients' breakdown: carbohydrates, protein, and fat. The dataset was analyzed to drop the users with missing or inconsistent logs. Only the users who logged at least one meal per day over a consistent period of time are included. Afterward, all the meals were aggregated into 3 categories: breakfast, lunch, and dinner. The snacks were summed up with their corresponding meal category (for example, the morning snacks were added to breakfast). After preprocessing, the dataset contained 89 users who concurrently recorded their meals for 151 days.

Fitbit: Fitbit Charge 2 HR devices were used for this study to observe 25 subjects distributed across Belgium and Sweden. 12 devices were used for dataset collection, with 2 continual participants (male and female), and 10 users in circulation. The users were asked to record a minimum of 60 days of observations. We collected more than $17M$ measurements related to users' meal logs, heart rate (HR), calories burned, steps taken, activity and sleep. Nutritionix API [29] was used to impute the missing nutritional breakdown for meals. The measurements were aggregated into 3 records per day for each user. Each aggregated record contained the nutritional breakdown for a meal (breakfast/lunch/dinner), calories burned during the mealtime, resting HR from the previous day, and activity records for that day. The complete spectrum of data ranges is shown in Table II.

Since a huge amount of IID is collected by the Fitbit platform, the number of participants is relatively small as it requires fully informed consent for data disclosure. The biggest advantage of experimenting on this dataset is the private information available for each user. However, as discussed in IV-A, all the IID were removed for actual processing.

TABLE II: Recorded features for Fitbit dataset

Features		Unit	Granularity		Range
			Recorded	Aggregated	
Macro-nutrients	Fat	gm	food	meal	0.03 – 25
	Carbs	gm	food	meal	0.01 – 105
	Protein	gm	food	meal	0.04 – 55
Calories	Burned	kcal	food	meal	416 – 1435
Heart rate	Resting	bpm	7.5 s	day	49.5 – 83.4
Activity	Light	mins	day	day	2 – 481
	Moderate	mins	day	day	0 – 211
	High	mins	day	day	0 – 253
	Sedentary	mins	day	day	600 – 998

Fitbit-GAN: We employed conventional Generative Adversarial Networks [30] to generate augmented Fitbit data for training our models. We refer to this dataset as Fitbit-GAN.

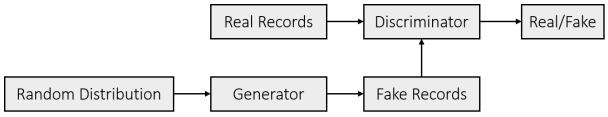


Fig. 5: GAN model for data augmentation.

The proposed GAN approach is showed in Fig. 5. The generator creates a data record from a random distribution. Next, the generated record is fed to the discriminator, alongside aggregated records taken from the actual data. The two models are trained together in an adversarial zero-sum game i.e. with time the discriminator is updated to become stronger in discrimination (real or fake) and the generator is updated to fool the discriminator based on how well or not the fake records are created. This process continues until the discriminator model is fooled about half the time, meaning the generator model is generating plausible examples.

A sample of the synthetically generated data record of two days for a user using GAN is shown in Table III. It can be seen that the GAN is able to learn the macronutrients breakdown for meals, calories burned, resting HR and daily activities.

TABLE III: Synthetic data using GAN

Meal	Fat	Carbs	Protein	Calories Burned	Resting HR	Active Minutes			
						Lightly	Moderately	Very	Sedentary
Breakfast	2.97	35.04	10.56	515.27	64.21	170	22	10	768
Lunch	11.97	47.83	25.64	655.85 8	64.21	170	22	10	768
Dinner	12.73	46.46	21.06	679.38	64.21	170	22	10	768
Breakfast	3.30	28.92	10.76	505.71	64.67	171	23	12	723
Lunch	13.92	49.80	14.40	665.38	64.67	171	23	12	723
Dinner	13.61	54.84	15.91	869.65	64.67	171	23	12	723

B. Results - Federated Learning without Privacy

In all the experiments, we predict the health data streams for the next day (3 meals' breakdown, calories burned, HR and activities) based on the previous day (3 meals' breakdown, calories burned, resting HR and activities). We discuss our findings and highlight the contributions they bring in the context of these research questions:

- How accurately can we predict the dietary and health-related behaviour of a user?
- Does grouping similar users bring any benefit in terms of accuracy and/or model training time?
- What impact do privacy preservation methods have on the accuracy of the forecasting?

1) Choosing the right model: First, we compared the most common statistical models for multivariate time-series forecasting such as Vector Autoregression (VAR), Vector Autoregression Moving-Average (VARMA), and Vector Autoregression Moving-Average with Exogenous Regressors (VARMAX); with popular NN architectures to choose the best configuration for our experiments. For NN architectures, we experimented with FNNs, LSTMs and GRUs with a sample of 10% from MyFitnessPal dataset. Our results indicate that the NN architectures are better candidates for our problem.

Next, we performed a grid search for the following hyperparameters for the NN architectures: learning rate (η), batch size (b), and the number of neurons on each layer l. Table IV shows the results of the best three models obtained, tested for a different number of epochs e and rounds r in a federated learning (FL) process. We chose the LSTM-2 architecture, i.e. LSTM with two hidden layers with 5 epochs per round for our following experiments as it offers the best prediction accuracy.

TABLE IV: Accuracy of the FL models using a grid search

Best models	Parameters	Federated 40e - 5r		Federated 20e - 10r		Federated 10e - 20r		Federated 5e - 40r	
		MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
FNN-2	$\eta = 0.001$ $b = 32$ $l = 10$	14.351	17.94	9.238	10.717	4.77	5.525	1.33	1.603
LSTM-2	$\eta = 0.001$ $b = 32$ $l = 200$	11.278	14.398	6.391	7.965	2.35	2.69	0.655	0.853
GRU-2	$\eta = 0.001$ $b = 32$ $l = 200$	10.226	12.703	16.11	18.473	4.104	5.064	1.289	1.621

2) MyFitnessPal dataset: We now present and analyze our results for the MyFitnessPal dataset.

Baseline FL model: We analyze the accuracy of the model trained on the whole dataset and refer to it as the baseline FL model in this case. Table V shows the achieved performance by training the model with parameters discovered using the grid search in terms of Mean Absolute Error (MAE) and Root Mean Square Error (RMSE). As can be noted, our model can forecast the next time-series with an MAE equal to 0.246, meaning that each predicted value will be at most 0.246 gms higher or lower than the ground truth. Given that the macronutrients in this dataset have a minimum value of 0.5 and a maximum of 609, we believe our system is capable of making highly accurate predictions.

Clustered FL models: Next, we analyze the performance of training a separate model for each cluster of users. Our clustering method identified 4 user clusters in this dataset. Table V shows the prediction accuracy of the FL models. It is important to remember that all the models in this comparison were trained using the same hyperparameters and number of epochs as the baseline FL model. Our results show that most of the clustered models are able to achieve high accuracy, though not as high as the baseline model. In particular, cluster 3 shows the highest increase in observed error. However, cluster 3 contains only 8 users and the least amount of data. So, we believe the data might not be enough for the model to learn from in this case.

We now analyze the training time needed for each model. Even though the clustered models did not outperform the baseline model, it is noteworthy that we are still able to achieve

TABLE V: Comparison between the baseline model and the clustered FL models. Dataset: MyFitnessPal.

FL Model		MAE	RMSE	Change in observed error	Training time (sec)
Baseline		0.246	0.319	-	5966
Clustered	Cluster 1 (14 users)	0.712	0.811	+2.89×	1082
	Cluster 2 (43 users)	0.536	0.642	+2.18×	3036
	Cluster 3 (8 users)	1.298	1.521	+5.28×	691
	Cluster 4 (10 users)	0.436	0.548	+1.77×	822

accurate predictions with a drastic decrease in training time. For example, cluster 4 manages to predict the macronutrients with an MAE equal to only 0.436 after training for a period which is around $7\times$ smaller than the baseline model.

Discussion: This set of experiments leads to the following conclusions: (1) we can accurately predict the macro-nutrient breakdown of meals, (2) clustering similar users does not improve the prediction accuracy with small clusters, but (3) we are still able to achieve high performance in terms of significantly less training time.

It is challenging to say why the clustering mechanism failed to improve accuracy. One possible reason could be the small number of users available for each cluster for the clustering mechanism to actually exhibit its benefits. Another plausible cause could be the homogeneity of the dataset. In this case, even if our method managed to discover some groups, the similarity between members of the same group might not be high enough for the model to benefit from. A third cause might reside in the process of choosing an appropriate model for our data. As mentioned before, we ran a grid search to find the parameters that would perform the best on our entire dataset. However, doing so optimizes the model for the entire dataset. Using the same configuration for the clusters might train an overfitted model, which in turn becomes a potential source of performance drop.

3) Fitbit dataset: The Fitbit dataset serves as a real-world example for our specific use-case, as it exhibits gaps in the time-series. This dataset contains 25 users and more private information, such as resting HR and active minutes throughout the day. Similar to the previous dataset, we performed a grid search to find the optimal hyperparameters.

Baseline FL model: We first investigate the performance of our FL procedure on the entire dataset. Table VI shows the accuracy of our baseline FL model. It should be noted that the number of training rounds needed to be adjusted due to the size of the dataset. Training for a longer period led to overfitting. Here we notice an overall accuracy drop as MAE is higher as compared to the MyFitnessPal dataset results. This is probably caused by two factors: a considerably smaller amount of training data and gaps in the time-series. Nonetheless, predicting the next macronutrients intake with a precision of ± 3.27 gms is still remarkable. The same reasoning can be applied to other features.

Clustered FL models: We cluster our users based on their similarities. Our method found 4 clusters of various sizes. We trained a separate FL model for each cluster. Table VII

TABLE VI: Prediction accuracy of the baseline model. Dataset: Fitbit

Predicted	MAE	RMSE	Training time (sec)
Macronutrients	3.27	4.047	245
Calories burned	11.831	15.261	
Resting HR	0.859	1.044	
Active minutes	4.495	5.320	

contains the prediction accuracy of these models. The average change in observed error is in comparison to the model trained on the entire dataset. An increase in average observed error suggests a decrease in the model prediction accuracy.

TABLE VII: Prediction error obtained by training clustered FL models. Dataset: Fitbit

FL Model	Predicted	MAE	RMSE	Average change in observed error	Training time (sec)
Cluster 1 (3 users)	Macronutrients	7.776	9.429	+5.63×	64
	Calories burned	57.544	77.070		
	Resting HR	9.462	12.321		
	Active minutes	19.303	23.290		
Cluster 2 (3 users)	Macronutrients	6.651	7.780	+3.08×	62
	Calories burned	38.816	51.437		
	Resting HR	5.269	6.754		
	Active minutes	12.923	15.459		
Cluster 3 (9 users)	Macronutrients	4.119	4.933	+1.9×	110
	Calories burned	21.791	28.054		
	Resting HR	2.447	3.093		
	Active minutes	7.555	9.120		
Cluster 4 (6 users)	Macronutrients	3.571	4.293	+1.84×	88
	Calories burned	28.165	33.643		
	Resting HR	2.012	2.566		
	Active minutes	7.093	8.612		

Discussion: As expected, the clusters with the less training data show a considerable decrease in accuracy (clusters 1 and 2). Groups that have more training data manage to achieve better predictions. For these reasons, the baseline model outperforms all clustered models in this case. Although this dataset contains very interesting yet sensitive data, we believe that the amount of data in this dataset is a major impediment to drawing relevant conclusions. Hence, we focus on the Fitbit-GAN dataset for further experiments.

4) Fitbit-GAN dataset: The experiments performed on the Fitbit-GAN dataset aim to investigate whether a higher amount of data influences the outcome as compared to our previous experiments. The hyperparameters for the baseline model in this section have been chosen using a grid search. We now evaluate the performance of the FL mechanism.

Baseline FL model: Firstly, we focus on determining how well we can predict the features of our dataset. We perform multi-step forecasting and compute the MAE and RMSE as before. Table VIII shows the performance achieved by the model trained on the entire dataset. Similar to the MyFitnessPal dataset results, we conclude that our model can predict user behaviour very accurately, as our prediction is at most 0.025% as far from the ground truth value with respect to the range of each feature.

Clustered FL models: We cluster the users and train separate models for each group. Again, the baseline model outperforms the clustered models for all features. We note that optimizing the hyperparameters for the entire dataset might lead to overfitting for smaller chunks of the dataset as the data distribution for the groups will be different than the one for the entire dataset. We ran a grid search for optimal parameters for each cluster and found a less complex model configuration to be appropriate for the groups. Moreover, it has been found that the same model configuration can cater to all groups (a small FNN-2 and the same hyperparameters). This will also lead to a major improvement in training time.

TABLE VIII: Prediction accuracy of the baseline model. Dataset: Fitbit-GAN

Predicted	MAE	RMSE	Training time (sec)
Macronutrients	0.806	1.054	9891
Calories burned	11.395	14.460	
Resting HR	0.044	0.058	
Active minutes	2.365	2.983	

Table IX shows a significant increase in prediction accuracy for three out of four clusters as compared to the baseline performance. Moreover, the highest training time, which has been recorded for cluster 4, is 20× faster than the training time of the baseline model.

TABLE IX: Prediction accuracy for training one model per cluster of users. Dataset: Fitbit-GAN

FL Model	Predicted	MAE	RMSE	Average change in observed error	Training time (sec)
Cluster 1 (167 users)	Macronutrients	0.458	0.553	-49%	374
	Calories burned	4.359	5.369		
	Resting HR	0.031	0.037		
	Active minutes	0.829	1.017		
Cluster 2 (63 users)	Macronutrients	0.639	0.788	-17%	199
	Calories burned	6.296	7.792		
	Resting HR	0.04	0.0485		
	Active minutes	2.459	2.822		
Cluster 3 (44 users)	Macronutrients	1.199	1.467	+57%	167
	Calories burned	12.152	14.65		
	Resting HR	0.088	0.107		
	Active minutes	4.101	4.884		
Cluster 4 (213 users)	Macronutrients	0.527	0.63	-47%	478
	Calories burned	5.493	6.8		
	Resting HR	0.02	0.031		
	Active minutes	1.18	1.468		

Discussion: In order to address the impact of the cluster size on the overall performance of the system, it should be noted that the clusters discovered by our mechanism are imbalanced. Our results might give the impression that the accuracy is directly proportional to the size of the dataset. However, this assumption is incorrect. The cluster size influences the results only when the NN does not have enough data to learn from. As long as the group contains enough data, the similarity between the members of each group can be leveraged to improve the prediction accuracy. We can, for example, consider the unclustered version to be a considerably larger cluster. The model trained on this group achieves poorer performance than the ones trained on real but smaller clusters. This proves that the similarity between users is the only factor that determines the increase in accuracy, and not the size of the dataset. It should also be noted that clustering improves the prediction accuracy as well as significantly improving the training time.

C. Results - Differentially Private Federated Learning

The impact of adding privacy preservation methods in the FL mechanism is evaluated by investigating the performance of the system by first noising the learning process and then

noising the data itself. The average results for each configuration are obtained by running each experiment at least 3 times.

1) Noisy learning: In this case, noise is added to the updates and sent to the federated aggregator using TensorFlow Privacy mechanisms.

Baseline private FL model: First, a baseline FL model for all users is trained. The results are presented in Table X. Adding Gaussian noise with a higher standard deviation decreases the prediction accuracy of the model but increases the achieved privacy level. Even for the lowest level of privacy, with $\varepsilon = 10.3$ corresponding to the standard deviation (sd) of 0.225, the model performance is approximately $35\times$ poorer than the non-private baseline model.

TABLE X: Results for noising the learning process to achieve DP in the baseline scenario. $clip$ and $noise$ specify the clip and noise applied to the gradients, with sd standard deviation. sd increases from left to right, suggesting that more noise is added. Contrarily, ε decreases from left to right, suggesting that better privacy levels are achieved as we add more noise.

		$clip = 0.3$ $noise = 0.75$ $(sd = 0.225)$	$clip = 0.5$ $noise = 0.75$ $(sd = 0.375)$	$clip = 0.75$ $noise = 1.2$ $(sd = 0.9)$	$clip = 1$ $noise = 1.3$ $(sd = 1.3)$	$clip = 1.5$ $noise = 2$ $(sd = 3)$
Macronutrients	**MAE**	25.899	24.637	36.308	37.176	77.838
	RMSE	26.835	25.325	37.62	38.457	79.022
Calories burned	**MAE**	678.038	423.555	698.392	818.229	881.288
	RMSE	687.112	437.704	709.221	837.541	900.358
Resting Heart Rate	**MAE**	1.133	2.065	3.089	3.859	4.129
	RMSE	1.208	2.115	3.1581	3.939	4.194
Active minutes	**MAE**	45.942	77.553	104.254	116.546	176.347
	RMSE	49.342	80.0	107.647	121.102	180.743
Epsilon (ε)		**10.3**	**10.3**	**4.27**	**3.8**	**2.2**

Clustered private FL models: When applying noise to the clustered scenario, our results showed that the models became unable to learn anything from the data, as they showcased an MAE higher than 1000 in some cases. Therefore, the results are not presented in this work.

Discussion: The main reasons behind the drastic decrease in prediction accuracy is the number of participants in the federated averaging algorithm which plays a major role on the achieved model performance [31]. Since our dataset is very small as compared to the one used by [31], our results are justifiable. Moreover, the fact that clustering the users caused an even greater performance drop in our experiments is also to be expected in this context, as the number of participants decreases even more when the model is trained for each separate cluster. This analysis shows that the noisy learning method is not appropriate for our use-case.

2) Noisy data: In this scenario, we add noise to the data itself. Each participant in the learning process learns from the noised data and tries to improve the FL model. It should also be noted that, when training on clusters of users, the clustering mechanism also receives noised data, hence, the clusters change for each experiment.

Baseline private FL model: We study the baseline scenario with only one private FL model for all the clients. Table XI shows the effect of various levels of data privacy on the model performance. Google can achieve DP with $\varepsilon = 2$ in certain conditions [27], and this is taken as a starting point for our experiments. Laplacian noise with 0 mean and $\frac{1}{\varepsilon}$ variance is added to the data. With $\varepsilon = 2$, we see an average increase in prediction error of only 2%. Moreover, the best trade-off between privacy and accuracy is obtained for $\varepsilon = 0.1$ where ε is ranging from 1 to 0.025. With this setting, we observe an increase in prediction error of around 21%, with good accuracy and a very high level of privacy is observed.

TABLE XI: Results of noising the training data to achieve DP in the baseline scenario.

		$\varepsilon = 2$	$\varepsilon = 1$	$\varepsilon = 0.5$	$\varepsilon = 0.1$	$\varepsilon = 0.025$	$\varepsilon = 0.01$
Macronutrients	**MAE**	0.743	0.715	0.752	0.874	0.984	1.336
	RMSE	0.969	0.943	0.995	1.135	1.378	1.843
Calories burned	**MAE**	11.596	11.937	12.231	14.383	14.891	19.634
	RMSE	14.687	15.451	16.116	18.482	19.903	27.392
Resting Heart Rate	**MAE**	0.048	0.055	0.058	0.067	0.077	0.122
	RMSE	0.065	0.07	0.077	0.088	0.108	0.161
Active minutes	**MAE**	2.483	2.246	2.293	2.344	2.832	3.832
	RMSE	3.012	2.953	2.768	3.159	3.941	5.438
Average increase in observed error		**2%**	**3%**	**7%**	**21%**	**37%**	**94%**

Clustered private FL models: We now focus on the clustering mechanism and its impact in the FL context. Firstly, as expected, clustering users with noised data alters the clusters. It can be noted that adding more noise to the data has two effects: (1) overall, fewer users are assigned to clusters, and (2) the algorithm discovers a higher number of smaller clusters. As the noise increases, the similarity between users decreases. Hence, the users that were previously very similar might still be clustered together, but the overall size of the groups is expected to decrease. The increased number of found clusters can be observed when applying privacy levels of 0.025, where the algorithm finds 5 clusters, instead of 4.

We examine the results obtained with the best ε value we discovered in the baseline case, 0.1. Table XII shows that unlike the noisy learning case, clustering the users can still improve the quality of prediction, even if the data is noised. The clusters show that several users maintain some degree of similarity that can be used to boost the accuracy of the model.

TABLE XII: Results of noising the training data to achieve DP in the clustered FL scenario. $\varepsilon = 0.1$ noise is added. A decrease in the average observed error (compared to baseline model) implies an increase in accuracy.

FL Model	Predicted	MAE	RMSE	Average change in observed error	Training time (sec)
Cluster 1 (73 users)	Macronutrients	0.982	1.131	-20%	222
	Calories burned	9.897	12.379		
	Resting HR	0.039	0.049		
	Active minutes	1.869	2.283		
Cluster 2 (37 users)	Macronutrients	1.025	1.209	-6%	150
	Calories burned	11.96	14.961		
	Resting HR	0.052	0.061		
	Active minutes	2.218	2.68		
Cluster 3 (161 users)	Macronutrients	0.593	0.715	-38%	392
	Calories burned	7.56	9.43		
	Resting HR	0.031	0.038		
	Active minutes	1.842	2.208		
Cluster 4 (97 users)	Macronutrients	0.587	0.712	-34%	262
	Calories burned	9.653	11.463		
	Resting HR	0.033	0.04		
	Active minutes	1.861	2.225		

Discussion: Noising the data improves prediction accuracy without any measurable effect on the training time. Adding noise to the data has been popularly used as a regularization technique for deep NN to avoid overfitting and improve accuracy. This could explain the very high accuracy we obtain, and a relatively small performance drop compared to the non-

noised model. Composing the overall ε experienced at the user end is outside the scope of this work. In principle, removing a user's data has no measurable impact on the overall clusters as well as the prediction accuracy of the system, although more data is shared in the clustering scenario. So, we believe that the overall experienced ε or privacy loss at the user end is also very small as the data is highly noised.

VII. Related Work

We discuss similar approaches found in the literature and compare them to our proposed pipeline. To the best of our knowledge, there is no other work that aims to investigate the problem of user diet and health forecasting in a streaming context with privacy guarantees. Thus, in the following subsections, we present works that are comparable to each of the steps chosen in our pipeline.

A. Multivariate Time-Series Clustering Mechanism

Our work is inspired by [32] whose clustering technique is composed out of two logical steps, discretization to univariate time series and clustering the resultant data. We adapted their framework to implement a streaming variant of for the first step – discretization, followed by computing the Hamming similarity for the second step – pattern matching. Since pattern matching is trivial compared to the discretization process, we focus on the related work for the latter.

Discretization of time-series: Our discretization process is very similar to the one proposed by [33]. The goal is to map each multivariate data point of a time-series into a discrete value. According to the authors, the shape of the pattern is the most important trait of a series instead of actual values. In their study, the real-valued time-series was split into windows. Each subsequence is clustered using k-means. Discretized version of the time-series is obtained by using the ID of the cluster with the closest centroid to each subsequence. Lastly, the series of symbols are clustered using a suitable similarity measure.

The symbolization method proposed by [33] is suitable for our case for two main reasons: (1) working with low-dimensional symbols decreases the execution time of the algorithm, which is crucial in low-latency systems such as streaming applications, and (2) working with symbols instead of raw data brings benefit in terms of stronger user privacy.

Our work is also inspired by the work proposed by [34], where the authors aimed to discover similar patterns in the traveling habits of the subjects over streaming trajectories. Their approach partitions the stream into windows applies a clustering algorithm to observe the movements of the individuals in the given time frame and uses the clusters to detect co-movement patterns. The authors cluster the locations with respect to a specific threshold and we follow the same principle with a few tweaks that suit our problem.

B. Approaches for Federated Learning

We focus on studies that aim to solve two of the drawbacks of FL: the shortcomings of training only one federated model that should provide accurate predictions for all the participants, and the need for additional privacy methods so that sensitive data cannot be inferred from the updates sent to the server. We organize our discussion around these two problems.

Clustered federated learning: Clustered Federated Learning framework [35] was introduced to improve the performance of classic federated algorithms by leveraging natural groups that exist in the client population. The concept has been adopted by several studies, such as [36], who applied a clustering technique on patients' data to boost the performance in predicting the hospitalization time and mortality using electronic medical records. Moreover, [37] studied time-series forecasting and showed that training separate global models for different clusters of time-series improves the performance.

This procedure is not very different from the classic FL. The server clusters the clients according to a chosen similarity metric. Let us assume it discovers k groups. Then, the FL algorithm proceeds with the small modification that the server maintains k models instead of one. When an update from a client arrives, the server must first check the cluster label of the client to decide which model the user's update will contribute to. Also, each client has to download the model corresponding to its cluster. We adapted this approach for our work.

Differentially private FL: Federated learning made a significant step towards better privacy protection by offering a training mechanism that can learn from clients' data without having access to the actual data. However, sensitive information can still be inferred from the updates sent to the central server. Zhu et al. [38] proved in their study that it is, in fact, possible to obtain such information from shared gradients.

Other aggregation algorithms only use the weights obtained from training the local model. Studies show that this technique is not safe either and that private information of individuals can still be divulged [39], [40]. Regardless of the parameters chosen to be shared, each new communication round in an FL algorithm can lead to data leaks, which accumulate in time throughout the process. DP can, thus, be used to conceal the contribution of each client during the training process. Similar to the general idea of DP, a trade-off must be found between privacy loss and model performance.

Geyer et al. [31] address the problem of differentially private FL from a client point of view. Their algorithm distorts the updates sent at each communication round by adding Gaussian noise. This method can maintain the privacy of the clients with only a small decrease in performance, given the dataset size is large enough. Similarly, [41] tackle the same problem in the context of sensitive health data. In their work, the authors add noise to the optimization function and prove this approach offers good data privacy while maintaining an adequate model performance. We applied this DP technique and studied its effect on the prediction accuracy of the model.

C. Noisy Data with LSTMs for Forecasting

We also study the effect of another DP method, which involves disturbing the training datasets themselves by adding noise. The study that served as our main inspiration for this DP algorithm has been undertaken by [42], in which the authors

used DP for stock price predictions. Albeit in a non-federated context, this study is closely related to our work because it noises the data directly to achieve better privacy preservation.

VIII. Conclusions and Future work

We provide an online system that can forecast the dietary habits and health data of users of fitness tracking applications and/or wearable devices. To this extent, we have designed and implemented a pipeline capable of accurately predicting user behaviour and that can leverage similarities between individuals to improve model performance while guaranteeing data privacy. Depending on the dataset and features, our predictions are no more than 0.025% far off the ground-truth value with respect to the range of the value. Moreover, our clustering mechanism leverages similarity between the users to improve prediction accuracy while reducing the model training time, with up to 49% error reduction as compared to an FL model trained for the whole dataset. With high privacy guarantees on user data $\varepsilon = 0.1$, we show that the baseline model has a small drop in prediction accuracy and that data noising mechanism benefits from user clustering. Our clustering system manages to sustain the prediction accuracy and, in most cases, improve it, with a reduction of 38% error in prediction accuracy as compared to the baseline noisy data model in the best case. For future work, we consider investigating adaptive k-means and online ML models. We believe adaptive modeling will help in improving the performance of the system.

Acknowledgment

This work is partly funded by the Erasmus Mundus Joint Doctorate program in Distributed Computing (EACEA of the European Commission under FPA 2012-0030). The authors are truly grateful to the anonymous participants who volunteered to provide their data for this study. We are also thankful to Prof. Paris Carbone and Prof. Sonja Buchegger at KTH, and Prof. Ramin Sadre at UCLouvain, Belgium, for their valuable advice, guidance and support during this research project.

References

[1] Fitbit. [Online]. Available: https://www.fitbit.com/se/home
[2] MyFitnessPal. [Online]. Available: https://www.myfitnesspal.com/
[3] M. K. Kundalwal *et al.*, "An improved privacy preservation technique in health-cloud," *ICT Express*, vol. 5, no. 3, pp. 167–172, 2019.
[4] S. Imtiaz, R. Sadre, and V. Vlassov, "On the case of privacy in the IoT ecosystem: A survey," in *International Conference on Internet of Things (iThings)*. IEEE, 2019, pp. 1015–1024.
[5] M. Young *et al.*, "Beyond open vs. closed: Balancing individual privacy and public accountability in data sharing," in *Proceedings of the ACM FAT*, 2019, pp. 191–200.
[6] J. Jordon, J. Yoon, and M. van der Schaar, "PATE-GAN: Generating synthetic data with differential privacy guarantees," in *ICLR*, 2018.
[7] S. Truex *et al.*, "A hybrid approach to privacy-preserving federated learning," in *Proceedings of the 12th ACM AISec*, 2019, pp. 1–11.
[8] Z. Abbas, J. R. Ivarsson, A. Al-Shishtawy, and V. Vlassov, "Scaling deep learning models for large spatial time-series forecasting," in *IEEE Big Data*, 2019, pp. 1587–1594.
[9] T. W. Liao, "Clustering of time series data - a survey," *Pattern Recognition*, vol. 38, no. 11, pp. 1857–1874, 2005.
[10] I. S. Dhillon and D. S. Modha, "A data-clustering algorithm on distributed memory multiprocessors," in *Large-scale parallel data mining, Workshop at SIGKDD*. Springer, 2002, pp. 245–260.
[11] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
[12] M. Arsalan and A. Santra, "Character recognition in air-writing based on network of radars for human-machine interface," *IEEE Sensors Journal*, vol. 19, no. 19, pp. 8855–8864, 2019.
[13] H. B. McMahan *et al.*, "Federated learning of deep networks using model averaging," *CoRR*, 2016. [Online]. Available: http://arxiv.org/abs/1602.05629
[14] B. McMahan *et al.*, "Communication-efficient learning of deep networks from decentralized data," in *AISTATS*. PMLR, 2017, pp. 1273–1282.
[15] C. Dwork *et al.*, "Calibrating noise to sensitivity in private data analysis," in *TCC*. Springer, 2006, pp. 265–284.
[16] C. Dwork et al., "Our data, ourselves: Privacy via distributed noise generation," in *EUROCRYPT*. Springer, 2006, pp. 486–503.
[17] C. Dwork and J. Lei, "Differential privacy and robust statistics," in *Proceedings of STOC'09*, 2009, pp. 371–380.
[18] M. Bun and T. Steinke, "Concentrated differential privacy: Simplifications, extensions, and lower bounds," in *Theory of Cryptography Conference*. Springer, 2016, pp. 635–658.
[19] C. Dwork, G. N. Rothblum, and S. Vadhan, "Boosting and differential privacy," in *FOCS'10*. IEEE, 2010, pp. 51–60.
[20] P. Kairouz, S. Oh, and P. Viswanath, "The composition theorem for differential privacy," in *ICML*, 2015, pp. 1376–1385.
[21] P. Carbone *et al.*, "Apache flink: Stream and batch processing in a single engine," *IEEE Data Engineering Bulletin*, vol. 38, pp. 28–38, 2015.
[22] Z. Huang, "Extensions to the k-means algorithm for clustering large data sets with categorical values," *Data mining and knowledge discovery*, vol. 2, no. 3, pp. 283–304, 1998.
[23] TensorFlow-Federated. [Online]. Available: https://www.tensorflow.org/federated
[24] K. Bonawitz *et al.*, "Towards federated learning at scale: System design," *arXiv preprint arXiv:1902.01046*, 2019.
[25] H. B. McMahan *et al.*, "A general approach to adding differential privacy to iterative training procedures," *arXiv preprint arXiv:1812.06210*, 2018.
[26] C. Dwork, A. Roth *et al.*, "The algorithmic foundations of differential privacy." *FnT-TCS*, vol. 9, no. 3-4, pp. 211–407, 2014.
[27] Ú. Erlingsson, V. Pihur, and A. Korolova, "Rappor: Randomized aggregatable privacy-preserving ordinal response," in *Proceedings of the 2014 ACM SIGSAC CCS*, 2014, pp. 1054–1067.
[28] I. Weber and P. Achananuparp. Myfitnesspal food diary dataset. [Online]. Available: https://doi.org/10.13140/RG.2.2.14511.64167
[29] Nutritionix API. [Online]. Available: https://developer.nutritionix.com/
[30] I. Goodfellow *et al.*, "Generative adversarial nets," in *NIPS*, 2014, pp. 2672–2680.
[31] R. C. Geyer, T. Klein, and M. Nabi, "Differentially private federated learning: A client level perspective," *NIPS'17. arXiv:1712.07557*, 2017.
[32] T. W. Liao, "A clustering procedure for exploratory mining of vector time series," *Pattern Recognition*, vol. 40, no. 9, pp. 2550–2562, 2007. [Online]. Available: https://doi.org/10.1016/j.patcog.2007.01.005
[33] G. Das, K.-I. Lin, H. Mannila, G. Renganathan, and P. Smyth, "Rule discovery from time series." in *KDD*, vol. 98, no. 1, 1998, pp. 16–22.
[34] L. Chen *et al.*, "Real-time distributed co-movement pattern detection on streaming trajectories," *Proceedings of the VLDB Endowment*, vol. 12, no. 10, pp. 1208–1220, 2019.
[35] F. Sattler, K.-R. Müller, and W. Samek, "Clustered federated learning: Model-agnostic distributed multi-task optimization under privacy constraints," *arXiv preprint arXiv:1910.01991*, 2019.
[36] L. Huang *et al.*, "Patient clustering improves efficiency of federated machine learning to predict mortality and hospital stay time using distributed electronic medical records," *Journal of biomedical informatics*, vol. 99, p. 103291, 2019.
[37] F. Díaz González, "Federated learning for time series forecasting using LSTM networks: Exploiting similarities through clustering," Master's thesis, KTH, EECS, 2019.
[38] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," in *NIPS*, 2019, pp. 14 774–14 784.
[39] C. Ma *et al.*, "On safeguarding privacy and security in the framework of federated learning," *IEEE Network*, 2020.
[40] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proceedings of the 22nd ACM SIGSAC CCS*, 2015, pp. 1310–1321.
[41] O. Choudhury *et al.*, "Differential privacy-enabled federated learning for sensitive health data," *arXiv:1910.02578*, 2019.
[42] X. Li *et al.*, "DP-LSTM: Differential privacy-inspired LSTM for stock prediction using financial news," *arXiv:1912.10806*, 2019.