2022 IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom) | 978-1-6654-9425-0/22/$31.00 ©2022 IEEE | DOI: 10.1109/TrustCom56396.2022.00080

Energy-Efficient Privacy-Preserving Time-Series Forecasting on User Health Data Streams

Muhammad Arsalan*, Davide Di Matteo†, Sana Imtiaz†‡, Zainab Abbas†‡, Vladimir Vlassov†, Vadim Issakov*
*Technische Universität Braunschweig, Braunschweig, Germany
† KTH Royal Institute of Technology, Stockholm, Sweden
‡ KRY International AB, Stockholm, Sweden
{muhammad.arsalan,v.issakov}@tu-braunschweig.de, {davidedm,vladv}@kth.se,{sana.imtiaz,zainab.abbas}@kry.se

***Abstract*—Health monitoring devices are gaining popularity both as wellness tools and as a source of information for healthcare decisions. In this work, we use Spiking Neural Networks (SNNs) for time-series forecasting due to their proven energy-saving capabilities. Thanks to their design that closely mimics the natural nervous system, SNNs are energy-efficient in contrast to classic Artificial Neural Networks (ANNs). We design and implement an energy-efficient privacy-preserving forecasting system on real-world health data streams using SNNs and compare it to a state-of-the-art system with Long short-term memory (LSTM) based prediction model. Our evaluation shows that SNNs trade-off accuracy (2.2× greater error), to grant a smaller model (19% fewer parameters and 77% less memory consumption) and a 43% less training time. Our model is estimated to consume 3.36μJ energy, which is significantly less than the traditional ANNs. Finally, we apply ε-differential privacy for enhanced privacy guarantees on our federated learning-based models. With differential privacy of ε = 0.1, our experiments report an increase in the measured average error (RMSE) of only 25%.**

***Index Terms*—Spiking neural networks, differential privacy, federated learning, smart health care, fitness trackers.**

I. Introduction

Health tracking gadgets have exploded in popularity in the past three decades [1] and their consumers mainly include clinicians for tracking patient's health and individuals for self-monitoring. Self-monitoring the diet and health habits could help users reach fitness and well-being goals effectively as it shows real progress over time. For example, an athlete preparing for a competition may set nutritional objectives. It is beneficial to monitor the projected development of calorie intake over the following weeks using historical data.

Health data contains sensitive information, thus privacy considerations must be taken into account when processing health data. Under the General Data Protection Regulation (GDPR) in the European Union, sensitive data-processing systems must give stringent assurances for data privacy and security to prevent maltreatment of sensitive health care data [2]. We apply federated learning (FL) [3], [4] and differential privacy (DP) techniques [5] for providing privacy guarantees on time-series forecasting. FL is useful for enabling forecasting models to work on edge devices that avoids moving sensitive data to a central server. Instead of data, only model training parameters are shared with central models. We explore energy-efficient learning models for forecasting in order to work with edge devices having limited compute power.

Conventional deep neural networks (deepNets) show effective results in time-series forecasting applications, but they are not energy efficient. In deepNets, majority of the energy consumption happens during the multiply-accumulate (MAC) operations between layers [6]. Therefore, the research community mostly focuses on reducing MACs by using smaller networks, weight quantization, and pruning techniques [7], [8].

To address the energy efficiency issue, we use SNNs [9]. As opposed to deepNets, SNNs are quite energy efficient as the information in SNNs is transmitted by spike timing, including latencies and spike rates. Additionally, the transmission of information is sparse as information is communicated only when the membrane potential of a neuron reaches a certain threshold. Besides this, communication in SNNs is a 1-bit activity, which significantly reduces the amount of data communicated at nodes. Furthermore, the MAC operation is replaced with adders, making the SNNs energy efficient.

Although SNNs are energy efficient [10]–[13], training SNNs is challenging due to the non-differentiable transfer function preventing backpropagation. Therefore, a suitable learning mechanism is required for training SNNs. Among the learning mechanisms, conventional backpropagation is tweaked usually for deep SSNs, where first the network is trained with differential approximated activation functions, and then in testing phase, the functions are replaced with spiking activations to make the network an SNN. We follow the same approach in our work.

The contributions of our work are as follows:

- We have designed a novel time-series forecasting system for user health data streams using SNNs that is more energy-efficient than the state-of-the-art LSTM-based counterpart.
- We have implemented and integrated the proposed forecasting model into an end-to-end health data streams forecasting pipeline based on clustered FL.
- We have incorporated ε-differential privacy into our pipeline and conducted an exhaustive analysis of its accuracy implications on the baseline and clustered models.
- We have compared SNN model with its LSTM-based counterpart by evaluating the model size, number of trained parameters, model accuracy, training time, and estimating energy consumption on an augmented real-world Fitbit dataset.

2324-9013/22/$31.00 ©2022 IEEE
DOI 10.1109/TrustCom56396.2022.00080

II. Preliminaries

A. Spiking Legendre Memory Unit

In neural communication, spikes are transmitted and filtered through synapses using complex processes. These complex processes can be modelled using ordinary differential equations (ODEs) to approximate the behaviour of a cell time [14]. The Legendre memory unit (LMU) in this regard, is capable of approximating in similar fashion a continuous-time delay [15]. The LMU works by finding weights on a collection of Legendre basis functions that predict an output time-varying signal when assessed at a certain time [14]. If the LMU network is properly trained on the training data, the projected output signal should match the input signal. For example, the Fourier coefficients are similar to how the LMU weights the Legendre basis functions in the LMU network. Using Fourier coefficients, one may reconstruct any time-varying signal by weighting and combining the weighted sine functions [16]. The spiking version of the LMU is obtained by replacing the non-spiking *Tanh* activation function to spiking *Tanh* activation during the test phase. This is explained further in section IV.

B. Federated Learning

FL is a distributed machine learning (ML) technique to train a generic model by combining multiple local models trained on local data originating from multiple distributed clients [3], [4]. FL algorithm consists of two kinds of actors: *clients* and a *central coordinator* (often called a *server*). The FL mechanism learns from all the clients' data as follows: Each client holds a local dataset containing only that clients' data. The server shares a central model with all the clients. Each client improves its current model by training on local data and sends updated training parameters back to the server. The server aggregates these updates from multiple clients, and an improved central model is created and shared back with the clients. The process repeats as the clients collect more data. Thus, the FL mechanism learns from all the clients' data without seeing it. Moreover, FL supports a variety of features attributed to distributed ML on mobile client devices. For example, catering to unbalanced and massively distributed datasets, processing non-independent and identically distributed data, and high capability to function with limited communication.

C. Differential Privacy

DP can be defined mathematically as follows [5]. Given a randomized algorithm A, the set of all datasets D and D' that differ on at most one row (one individual's data), and any subset $S \subseteq range(A)$:

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

Here, ε is used to quantify privacy loss for one row (one individual's data). This ε-DP is achieved by noise addition according to ε. Small values of ε imply better privacy preservation and higher amounts of noise addition. Absolute privacy is obtained when $\varepsilon = 0$, where the output of algorithm A is exactly the same with or without an individual's data. However, achieving these high levels of privacy involves using highly noised data which leads to a decrease in the accuracy of the algorithm. Therefore, a trade-off must be made between preserving user privacy and achieving meaningful and accurate results. When an algorithm requires multiple additive noise mechanisms, the privacy guarantees follow from the basic DP composition theorem [17], [18] or advanced composition theorems [19]–[21].

III. Data processing pipeline

This section explains the overall data processing pipeline for our health data prediction system. The health data streams collected from Fitbit devices consist of: user's daily meal logs indicating the carbohydrates, fats, and proteins intake; the calories burned by the user, the resting heart rate, and the active minutes. The forecasting system has two main components: 1) the clustering mechanism that uses streaming k-means and pattern matching for grouping users into *k* groups based on their meal logs, and 2) the health prediction system that uses FL-based models for each user group to forecast user health and meal data. Figure 1 shows our health forecasting system pipeline with its clustering and forecasting components in combination with DP.

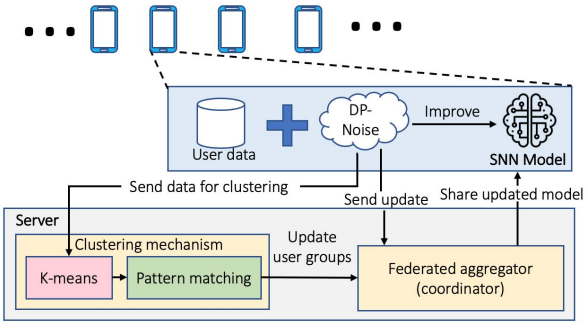

Fig. 1: Clustered FL with differential privacy.

A. The Clustering Mechanism.

User diet health data streams are used to group users with similar eating patterns by using streaming k-means clustering. The diet data consists of daily logs for breakfast, lunch and dinner. The streaming k-means clustering algorithm takes the first unique meals as the centroids and starts assigning the next meals in the streams to these centroids based on their Euclidean distance from the centroids. After clustering 7 days of logs with the chosen centroids, pattern matching is applied for grouping users with similar eating behaviour. The pattern for each user is created using centroid IDs. For example, consider a user's meal logs for 1 day that are assigned to centroids with IDs (0,1,2,...) per meal as shown in blue in Figure 2. Here, each number indicates a centroid ID for three meals (breakfast, lunch, and dinner) for three users. Based on the given clusters, the pattern for each user is: user1 (0,0,0), user2 (0,0,1), and users3 (1,1,1). Hamming distance [22] is computed to find similarities between eating patterns. Users

with the lowest Hamming distance have the closest diet pattern and end up in the same user group. FL is then applied to train the forecasting model per group and a central generic model.

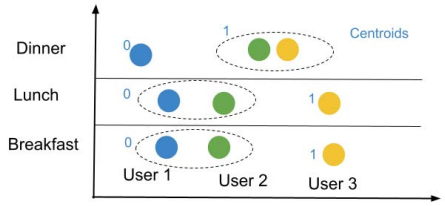

Fig. 2: Clustering users based on their meal logs.

B. Data Model for Training SNNs

Our dataset is a time-series of health data streams containing the meal logs, active minutes, resting heart rate and calories burned. All these data are used to train the forecasting models. Our goal is to make a single-step prediction or to estimate a user's health and meals based on previous data. We model our data according to the multiple-input single-output (MISO) strategy (also called single-step prediction).

Given a time-series $[x_1, x_2, x_3, ..., x_n]$, an input array of size M, we predict the next element x_{n+1} of the series. The MISO approach intends to learn a function $\hat{F}$ as follows:

$$\hat{F}(x_{t+1}, ..., x_{t+M}) = \hat{x}_{t+M+1} \tag{2}$$

where $\hat{x}_{t+M+1}$ represents the *single-step* approximation through the function $\hat{F}$. In other words, we predict the next data point x_{t+1} using a sliding window of length M and incremented by 1. Here, the length of the sliding window M is a design choice.

C. Clustered Federated Learning and Differential Privacy

We train a separate FL model for each user cluster found by the clustering mechanism. Moreover, we add noise to the data itself using Laplacian noise addition mechanism for DP. Each participant model in the learning process learns from the noised data and tries to improve the FL model (or its respective FL model in clustered FL scenario). Figure 1 shows the noise addition mechanism for FL in the clustering scenario. Here, since the clustering mechanism also receives noised data when training on clusters of users, the user clusters change for each experiment. It should be noted that the prediction is performed locally and on clean data.

IV. Proposed Spiking Neural Networks Architecture

We propose using LMUs [23] in our novel SNN architecture for time-series forecasting as shown in Figure 3. The network is built in NengoDL simulator [24] which provides a spiking version of rate-based neurons. The network is trained with non-spiking *Tanh* as an activation function in the conventional backpropagation method. There are 3 fundamental parameters in an LMU network: units, order, and theta. Units represent the number of employed neurons, which are 64 in our case. The order represents LMU basis functions, which are the number

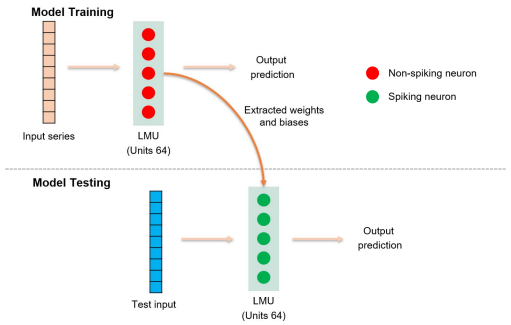

Fig. 3: The proposed SNN architecture where the model is first trained with non-spiking neurons with backpropagation. The test model is reconstructed with spiking neurons, and the weights and biases from the pre-trained model are used.

of Legendre polynomials used to orthogonally represent the sliding window which is set to 128 in our case. Increasing the LMU order helps to represent (and store) faster changes in the input signal. Finally, theta is the amount of time kept in the LMU internal memory. Theta affects how much time the basis functions represent, and hence how much temporal information is kept inside the LMU network for later use in training (and decoding). We set theta to 360 for our system.

1) Loss Function: We use Root Mean Square Error (RMSE) as a loss function. Since we are aiming to achieve accurate forecasting, we give more weightage to large errors. RMSE is mathematically expressed as:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{3}$$

where y_i denotes the ground truth, $\hat{y}_i$ denotes the prediction, and n denotes the number of samples.

2) Learning Schedule: For the learning step, we employ Adam (the adaptive moment estimation) technique for stochastic gradient descent, based on adaptive estimate of first- and second-order moments. The learning rate (α) is set to 0.003. The exponential decay rates for the 1st (β_1) and 2nd moment (β_2) estimates are respectively 0.9 and 0.999. Epsilon is set to $1e^{-7}$ for numerical stability.

3) Model Testing: The trained network is reconstructed during the testing phase using spiking *Tanh* neurons to create an SNN. Then, the weights of the trained model are extracted and used for spiking *Tanh* neurons. Finally, the test inputs are repeated multiple times and fed to the network to get an accurate measure of the spiking neuron output over time.

4) Model Hyperparameters: The hyperparameters involved in this study belong to three different groups: the spiking neuron model, LMU, and the model itself. These parameters are chosen empirically and are presented in Table I.

V. Evaluation Methodology

In our experiments we quantify the accuracy, model size, and the number of network parameters of our SNN model and compare them to the results obtained with the LSTM.

TABLE I: Hyperparameters for SNN

Hyperparameter	Value	Definition
Type of Neuron	Spiking Tanh	Neuron model used
τ_{ref}	0.005	Absolute refractory period expressed in seconds
LMU's units	64	Number of employed neurons
LMU's order	128	Number of Legendre polynomials
LMU's theta	360	Amount of time kept in the LMU's internal memory
Learning	0.003	Learning rate of our Adam optimiser
Minibatch Size	100	Number of concurrent inputs fed to the network
Epochs	10	Number of passes of the entire training dataset
Delay	10 ms	Latency of the output in respect to the input
Inference's time steps	50	Number of inferences of SNN before results

Evaluation is done on both clustered and non-clustered users. Finally, we determine how privacy guarantees influence overall prediction accuracy, by comparing the accuracy obtained when these privacy-preserving techniques are toggled on and off.

A. Dataset

a) The Original Fitbit Dataset: The Original Fitbit Dataset consisted of 25 users and was obtained by exporting data from Fitbit wearable devices. Because the meals' nutritional breakdown was missing, we used Nutritionix API [25] to populate the information based on logged quantities of meals. Not all subjects kept a consistent record of their eating habits. This mirrors a real-world scenario in which it may not be in everyone's best interest to document every meal. Lastly, we limit the time-related labels to only include breakfast, lunch, and dinner; as these are the most common categories of meal in a day. Everything else was aggregated with their corresponding major meal (i.e. the late afternoon snacks were added to the dinner).

b) Data Augmentation Using Generative Adversarial Networks: The Augmented Fitbit Dataset is used as a real-world example of a health and nutrition monitoring dataset in this work. In addition, this dataset includes sensitive data, which is useful to investigate the effect of applying privacy protection measures on prediction accuracy.

TABLE II: The Augmented Fitbit dataset value range.

Features	Unit	Min. Value	Max. Value
Macronutrients	grams	0.313	1463.108
Calories burned	kCal	467.006	2335.445
Resting heart rate	bpm	55.029	89.995
Active minutes	minutes	0	1242.413

To validate our clustering algorithm, the dataset must have sufficient information to properly represent and learn from each group. To address this problem, the dataset was augmented using Generative Adversarial Network (GANs) [26], resulting in a dataset of 630 users with an average of around two months of logs per person. The details of data augmentation process can be found in [27].

Table II shows a summary of the features included in the Augmented Fitbit Dataset, along with their corresponding units and value ranges. The *Macronutrients* composition reflects the *carbohydrate*, *protein*, and *fat* content of the meals at each timestamp. The *Calories burned* at time t are the calories burned by the user between time $t-1$ and time t. The same logic applies to *Active minutes*. The *Resting heart rate* is calculated throughout the whole day.

B. Performance Metrics

During the learning phase, we evaluate our model accuracy using RMSE as shown in Eq. 3. To explain the performance measurements for the FL models, let m denote the total number of clients involved in the course of training. Let us assume that $X_1, X_2, ..., X_m$ are local datasets, one for each client. We divide the datasets as follows: for each local dataset, we utilize 80% of the data for training and 20% for testing. The overall performance is determined by calculating and averaging the selected errors from each local test dataset:

$$RMSE_{federated} = \frac{1}{m}\sum_{i=1}^{m} RMSE_i \tag{4}$$

VI. Results and Discussion

This section presents the results obtained in the experiments conducted over on the augmented Fitbit dataset. The four experiment configurations are: 1) the baseline model, which is the model trained on the whole dataset, 2) the fFL model, 3) the differential private baseline model, and 3) the differential private FL model. We also analyse and discuss the findings.

A. Baseline Model

We measure the baseline model accuracy, training time, and the model size using both SNNs and LSTMs.

TABLE III: Prediction accuracy of our SNN model in comparison with LSTM baseline model. Units are as in Table II

Predicted	RMSE LSTM [27]	RMSE SNN	Change in error
Macronutrients	1.540	3.364	+2.19x
Calories burned	14.460	31.732	+1.19x
Resting heart rate	0.0058	0.298	+4.13x
Active Minutes	2.983	6.457	+1.16x

1) Prediction Accuracy: We perform single-step prediction to measure prediction accuracy i.e. we forecast our features at the next meal (Breakfast, Lunch, or Dinner). Table III reports the accuracy achieved by both SNN and LSTM models. The results denote an increase positive increase in the error. On average, we achieved an error 2.17x higher than the one achieved in the LSTM model. We can say that although SNN is worse at predicting users' health habits than LSTM, it is still quite accurate with respect to the range of values (Table II). In fact, our forecast is within 0.331% of the ground truth value for the range of each feature.

2) Model Size: The proposed SNN model has significantly less ($\approx$19%) parameters and is 77% lighter than its counterpart LSTM as shown in Table IV.

TABLE IV: Number of parameters and model size (in bytes) of SNN and LSTM.

Metric	LSTM	SNN	Change
# Parameters	36003	28,929	-19.6%
Memory size (bytes)	483784	108856	-77.5%

TABLE V: Accuracy of clustered FL models using SNNs. The *Change in error* is shown against our baseline model (trained on the whole dataset).

Clusters	Total users	Predicted	RMSE	Change in error	Average training time
Cluster 1	209	Macronutrients	1.983	-41.1%	434s
		Calories Burned	18.071	-43.2%	
		Resting Heart Rate	0.166	-44.3%	
		Active Minutes	3.778	-41.5%	
Cluster 2	171	Macronutrients	2.353	-30.1%	387s
		Calories Burned	25.985	-18.2%	
		Resting Heart Rate	0.202	-32.2%	
		Active Minutes	4.751	-26.4%	
Cluster 3	56	Macronutrients	3.728	+10.8%	204s
		Calories Burned	37.742	+19%	
		Resting Heart Rate	0.375	+25.8%	
		Active Minutes	7.533	+16.6%	
Cluster 4	33	Macronutrients	5.192	+54.3%	147s
		Calories Burned	58.184	+83.4%	
		Resting Heart Rate	0.471	+58.1%	
		Active Minutes	10.464	+62%	

B. Clustered Federated Learning

For this experiment, we train single-feature models for each of the features within each cluster. Table V demonstrates a significant improvement in accuracy for two out of four clusters when compared to our SNN-based baseline performance. Additionally, Cluster 1, which is the model with the slowest training (with higher number of seconds), is more than 30× faster than our baseline model. Our baseline model outperforms Clusters 3 and 4. We conclude that cluster size has an effect on prediction accuracy only when the model has insufficient data to learn from. When the data in a cluster is sufficient, the similarity between the cluster components can be used to improve results. As a result, the similarity of users is a property that can trigger more accurate predictions.

C. Differentially Private Learning

We examine the effect of incorporating privacy-preserving techniques into the system, specifically in the baseline model for this experiment. Our method aims to ensure privacy by adding noise to the data itself. For this reason, we run the experiments 5 times and compute the average results for each configuration.

Table VI illustrates the accuracy decay associated with increasing degrees of privacy applied to the training data. It is possible to achieve DP with ε = 2 under certain conditions as demonstrated by Google [28]. For this reason, $\varepsilon = 2$ is the starting point for our experiments; in fact, we introduce Laplace noise with a mean of 0 and a variance of $\frac{1}{\varepsilon}$. As can be seen in Table VI, we achieve this degree of DP with an average increase of only 2% in the prediction's error. Furthermore, our investigations reveal that for ε spanning from 1 to 0.025, we can ensure an acceptable trade-off between privacy and accuracy. We estimate that the optimal trade-off occurs when ε = 0.1, at which point we detect an increase in prediction error of around 25%. As a result, we investigate the impact of using this level of privacy ($\varepsilon = 0.1$) in clustered FL configuration.

TABLE VI: Results after adding noise to our training data in order to ensure DP in our baseline model. The *Average change in error* is shown against our baseline model (trained on the whole dataset).

Predicted	RMSE				
	$\varepsilon = 2$	$\varepsilon = 1$	$\varepsilon = 0.1$	$\varepsilon = 0.025$	$\varepsilon = 0.01$
Macronutrients	3.408 (1.30%)	3.507 (4.25%)	4.142 (23.13%)	4.863 (44.56%)	7.533 (123.93%)
Calories Burned	32.541 (2.55%)	32.763 (3.25%)	39.891 (25.71%)	46.724 (47.25%)	59.263 (86.76%)
Resting Heart Rate	0.307 (3.02%)	0.315 (5.70%)	0.387 (29.87%)	0.428 (43.62%)	0.579 (94.30%)
Active Minutes	6.587 (2.01%)	6.694 (3.67%)	7.850 (21.57%)	9.941 (53.96%)	13.071 (102.43%)
Average change in error	2.22%	4.22%	25.07%	47.35%	101.86%

TABLE VII: Results after adding noise to our training data to ensure DP in the clustering configuration with $\varepsilon = 0.1$. Average change in error is shown against our baseline model (trained on the whole dataset) with differentially private learning (Section VI-C). Increases in the average change in error imply a reduction in accuracy

Clusters	Total users	Predicted	RMSE	Change in error	Average training time
Cluster 1	158	Macronutrients	2.821	-31.89%	376s
		Calories Burned	26.078	-34.63%	
		Resting Heart Rate	0.253	-34.63%	
		Active Minutes	5.272	-32.84%	
Cluster 2	101	Macronutrients	3.058	-26.17%	268s
		Calories Burned	31.853	-20.15%	
		Resting Heart Rate	0.301	-22.22%	
		Active Minutes	6.747	-14.05%	
Cluster 3	71	Macronutrients	3.601	-13.06%	214s
		Calories Burned	33.445	-16.16%	
		Resting Heart Rate	0.344	-11.11%	
		Active Minutes	6.681	-14.89%	
Cluster 4	34	Macronutrients	4.277	+3.26%	151s
		Calories Burned	40.172	+0.70%	
		Resting Heart Rate	0.396	+2.33%	
		Active Minutes	8.034	+2.34%	

D. Differentially Private Federated Learning

We now measure the effect of DP noise addition on the user clustering in our FL environment. Each participant to the learning process learns from the noised data and tries to improve the federated model. It should be noted that when training on clusters of users, the clustering mechanism also receives noised data. Thus, the clusters change for each experiment. It

is worth mentioning that compared to our second experiment (Section VI-B), increasing the noise in the data results in a fewer number of users per clusters overall. Consequently, the algorithm identifies a greater number of smaller clusters. This happens because the noise in the data is inversely proportional to degree of similarity between the users.

Table VII shows the results in terms of accuracy obtained when clustering the users with $\varepsilon = 0.1$ DP-noise addition. It is demonstrated that even if the data is noised, clustering the users has still a positive impact on the overall accuracy. This means even though noise is introduced to the data, the users manage to preserve a certain level of similarity which can be leveraged to increase the prediction accuracy. Our results show that when the clusters are small as in Cluster 4 (34 users), the model has still some difficulty in training due to scarce amount of data. By comparing the results obtained in this experiment with the non-noised model, we obtain high accuracy and low performance drop. In fact, adding a certain amount of noise to the data serves as a regularisation technique to boost accuracy and prevent overfitting. However, too much noise can also have a negative impact on the performance of the model.

E. Energy Estimation of SNN Model

For estimating the energy consumption of the proposed model, we use the hardware metrics of (μBrain) chip defined in [29], where the energy consumption per classification inference is given by:

$$E_i = N_{spikes} \times E_{spikes} + \delta T \times P_{leakage}$$

where E_i is the energy used per inference, $N_{spikes} = 3247$ denotes the maximum number of spikes during forecasting, $E_{spikes} = 2.1pJ$ denotes the energy per spike, $P_{leakage} = 73\mu W$ denotes the static leakage power, and $\delta T = 46ms$ denotes the inference time. The estimated energy consumption of our baseline model per inference is $E_i = 3.36\mu J$ per inference. Thus, our spiking neural network when predicting values on spiking neuromorphic hardware is estimated to be quite energy efficient as compared to deepNets solutions [30].

VII. Conclusion

Spiking neural networks prove to be efficient in terms of consuming less training time and memory with an acceptable accuracy, making them suitable for edge devices. Our research on developing a clustered FL-based health data prediction model demonstrates that user similarities may be leveraged to increase performance of forecasting models, and the success of clustering method is very data-dependent. When clusters lack sufficient data to train the algorithm, the prediction accuracy suffers significantly. Moreover, we also show that our clustering FL approach benefits from ε-DP. In fact, a significant improvement is noted in terms of prediction accuracy when similar users are brought together.

References

[1] R. Zenun Franco *et al.*, "Popular nutrition-related mobile apps: A feature assessment," *JMIR mhealth and uhealth*, vol. 4, 08 2016.

[2] P. Voigt and A. Von dem Bussche, "The EU General Data Protection Regulation (GDPR)," *A Practical Guide, 1st Ed., Cham: Springer International Publishing*, 2017.

[3] H. B. McMahan *et al.*, "Federated learning of deep networks using model averaging," *CoRR*, 2016. [Online]. Available: http://arxiv.org/abs/1602.05629

[4] B. McMahan *et al.*, "Communication-efficient learning of deep networks from decentralized data," in *AISTATS*. PMLR, 2017, pp. 1273–1282.

[5] C. Dwork *et al.*, "Calibrating noise to sensitivity in private data analysis," in *TCC*. Springer, 2006, pp. 265–284.

[6] V. Sze *et al.*, "Efficient processing of deep neural networks: A tutorial and survey," *Proc. of the IEEE*, vol. 105, no. 12, pp. 2295–2329, 2017.

[7] E. Hayashi *et al.*, "Radarnet: Efficient gesture recognition technique utilizing a miniature radar sensor," in *Proceedings of the CHI*, 2021.

[8] M. Scherer *et al.*, "TinyRadarNN: Combining Spatial and Temporal Convolutional Neural Networks for Embedded Gesture Recognition With Short Range Radars," *IEEE IoT-J*, vol. 8, pp. 10 336–46, 2021.

[9] W. Maass, "Networks of spiking neurons: The third generation of neural network models," *Neural Networks*, vol. 10, no. 9, pp. 1659–1671, 1997.

[10] G. Indiveri and T. K. Horiuchi, "Frontiers in neuromorphic engineering," *Frontiers in neuroscience*, vol. 5, p. 118, 2011.

[11] M. Pfeiffer and T. Pfeil, "Deep learning with spiking neurons: opportunities and challenges," *Frontiers in neuroscience*, vol. 12, p. 774, 2018.

[12] P. Panda *et al.*, "Toward scalable, efficient, and accurate deep spiking neural networks with backward residual connections, stochastic softmax, and hybridization," *Frontiers in Neuroscience*, vol. 14, p. 653, 2020.

[13] D.-A. Nguyen *et al.*, "A review of algorithms and hardware implementations for spiking neural networks," *J. Low Power Electron. Appl.*, vol. 11, no. 2, May 2021.

[14] A. R. Voelker and C. Eliasmith, "Improving spiking dynamical networks: Accurate delays, higher-order synapses, and time cells," *Neural Comput.*, vol. 30, no. 3, p. 569–609, mar 2018.

[15] A. Voelker *et al.*, "Legendre memory units: Continuous-time representation in recurrent neural networks," vol. 32, 2019.

[16] N. Chilkuri *et al.*, "Language modeling using lmus: 10x better data efficiency or improved scaling compared to transformers," *arXiv preprint arXiv:2110.02402*, 2021.

[17] C. Dwork et al., "Our data, ourselves: Privacy via distributed noise generation," in *EUROCRYPT*. Springer, 2006, pp. 486–503.

[18] C. Dwork and J. Lei, "Differential privacy and robust statistics," in *Proceedings of STOC'09*, 2009, pp. 371–380.

[19] M. Bun and T. Steinke, "Concentrated differential privacy: Simplifications, extensions, and lower bounds," in *Theory of Cryptography Conference*. Springer, 2016, pp. 635–658.

[20] C. Dwork, G. N. Rothblum, and S. Vadhan, "Boosting and differential privacy," in *FOCS'10*. IEEE, 2010, pp. 51–60.

[21] P. Kairouz, S. Oh, and P. Viswanath, "The composition theorem for differential privacy," in *ICML*, 2015, pp. 1376–1385.

[22] X.-S. Yang, *Nature-inspired optimization algorithms*. Academic Press, 2020.

[23] A. Voelker, I. Kajić, and C. Eliasmith, "Legendre memory units: Continuous-time representation in recurrent neural networks," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper/2019/file/952285b9b7e7a1be5aa7849f32ffff05-Paper.pdf

[24] NengoDL. [Online]. Available: https://www.nengo.ai/nengo-dl/

[25] Nutritionix API. [Online]. Available: https://developer.nutritionix.com/

[26] I. Goodfellow *et al.*, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[27] S. Imtiaz *et al.*, "Synthetic and private smart health care data generation using gans," in *2021 International Conference on Computer Communications and Networks (ICCCN)*, 2021, pp. 1–7.

[28] Ú. Erlingsson *et al.*, "Rappor: Randomized aggregatable privacy-preserving ordinal response," in *Proceedings of the 2014 ACM SIGSAC CCS*, 2014, pp. 1054–1067.

[29] J. Stuijt *et al.*, "μBrain: An Event-Driven and Fully Synthesizable Architecture for Spiking Neural Networks," *Frontiers in Neuroscience*, vol. 15, p. 538, may 2021.

[30] P. Blouw *et al.*, "Benchmarking keyword spotting efficiency on neuromorphic hardware," *ArXiv*, vol. abs/1812.01739, 2019.