Decentralized Data-Privacy Preserving Deep-Learning Approaches for Enhancing Inter-Database Generalization in Automatic Sleep Staging

Adriana Anido-Alonso and Diego Alvarez-Estevez

***Abstract*—Automatic sleep staging has been an active field of development. Despite multiple efforts, the area remains a focus of research interest. Indeed, while promising results have reported in past literature, uptake of automatic sleep scoring in the clinical setting remains low. One of the current issues regards the difficulty to generalization performance results beyond the local testing scenario, i.e. across data from different clinics. Issues derived from data-privacy restrictions, that generally apply in the medical domain, pose additional difficulties in the successful development of these methods. We propose the use of several decentralized deep-learning approaches, namely ensemble models and federated learning, for robust inter-database performance generalization and data-privacy preservation in automatic sleep staging scenario. Specifically, we explore four ensemble combination strategies (max-voting, output averaging, size-proportional weighting, and Nelder-Mead) and present a new federated learning algorithm, so-called sub-sampled federated stochastic gradient descent (ssFedSGD). To evaluate generalization capabilities of such approaches, experimental procedures are carried out using a leaving-one-database-out direct-transfer scenario on six independent and heterogeneous public sleep staging databases. The resulting performance is compared with respect to two baseline approaches involving single-database and centralized multiple-database derived models. Our results show that proposed decentralized learning methods outperform baseline local approaches, and provide similar generalization results to centralized database-combined approaches. We conclude that these methods are more preferable choices, as they come with additional advantages concerning improved scalability, flexible design, and data-privacy preservation.**

***Index Terms*—Data-privacy, deep-learning, domain adaptation, ensemble models, federated learning, inter-database generalization, sleep staging.**

Manuscript received 22 March 2023; revised 13 July 2023; accepted 24 August 2023. Date of publication 31 August 2023; date of current version 7 November 2023. This work was supported in part by Xunta de Galicia under Grant ED431H 2020/10 with open access charge financed by Universidade da Coruña, and in part by Universidade da Coruña and Centro de Investigación de Galicia "CITIC", funded by Xunta de Galicia through the collaboration agreement between Consellería de Cultura, Educación, Formación Profesional e Universidades, and in part by the Galician Universities, for the reinforcement of the research centers of the Galician University System (CIGUS). *(Corresponding author: Diego Alvarez-Estevez.)*

The authors are with the Universidade da Coruña, Centro de Investigación TIC de Galicia "CITIC", Elviña 15701 A Coruña, Spain (e-mail: adriana.anido@udc.es; diego.alvareze@udc.es).

Digital Object Identifier 10.1109/JBHI.2023.3310869

I. Introduction

In SLEEP Medicine, the polysomnographic (PSG) recording of the physiological activity of a patient throughout the night represents the standard tool for the diagnosis of numerous sleep disorders. Sleep macrostructure characterization, a.k.a. sleep staging, constitutes one of the most important tasks involved in the clinical review of the PSG. According to the standard protocol, the process involves the analysis of various recorded electroencephalographic (EEG), electrooculographic (EOG), and electromyographic (EMG) derivations, labeling the corresponding signal activity according to a set of pre-established visual scoring rules. This process, which takes place on a 30 s epoch-by-epoch basis, leads to construction of the so-called hypnogram, i.e. the resulting alternating epoch sequence of five possible sleep stages (W, N1, N2, N3, and R) throughout the night [1].

Visual analysis of vast amounts of data contained in the PSG, however, is complex, which also makes scoring prone to errors and subjective interpretations. Clinician's time, in addition, is expensive and scant. As a consequence, PSG analysis is one of the most time-consuming and costly tasks in the daily routine of a sleep center. Introducing automatic scoring to support clinicians in the sleep staging task, therefore, is interesting. It should contribute to reduce associated analysis times, enhancing production, and reducing the overall associated costs. Furthermore, expert-supervised automatic scoring has been shown to be able to improve inter-rater agreement, reducing variability and improving diagnostic quality [2], [3], [4], [5]. For this reason, many attempts have been made to automatize this process [6], [7], [8], [9], [10], [11], [12]. However, despite promising evaluation results reported in many of these works, uptake of automatic sleep scoring in the clinical setting remains low [13], [14], [15].

Accurate validation of automatic sleep scoring approaches has been traditionally biased due to limitations related to the associated benchmark datasets. Data would be scarce and lack enough heterogeneity, and evaluation would usually involve single source datasets relying on widespread train-test partitioning,

or local k-fold cross-validation. This approach leads to optimistic broad generalization estimation. Effectively, under the former setting, training and testing data are still gathered from the same local distribution. However, when the same algorithm is evaluated on completely external databases (e.g. from another sleep center) scoring performance drops significantly [16], [17], [18], [19], [20]. A number of reasons can be mentioned that contribute to this "database variability problem" in the case of sleep medicine [18]. These include differences among source patient populations, recording and/or acquisition methods, or the aforementioned divergences among clinical experts [3], [5], [21], [22], [23], [24]. More generally, challenges regarding the associated domain-shift are well-known in the scope of machine-learning, leading to related work in the sub-field of "domain adaption" [25]. Some recent approximations to automatic sleep staging are indeed focusing on developing ideas in the context of transfer-learning, i.e. reusing previous parametrization, or parts of a model, trained on a source dataset, to be fine-tuned using external independent datasets on a target domain [26], [27], [28], [29], [30], [31], [32], [33]. Nevertheless, all of the referenced approaches use more or less data from intermediate datasets to perform the transfer step. Therefore, the actual generalization capacity of these models on completely unseen target domains remains uncertain. That we know of, only one study has considered evaluation of an automatic sleep staging model, developed using transfer-learning, on an unbiased direct-transfer scenario [34].

An alternative approach to improve domain adaptation is to train the model by arranging a large centralized database using data from different heterogeneous cohorts [35], [36]. This strategy, however, has its own disadvantages, concerning complex logistics and high demand of resources related to centralization and learning from all these data. The resulting model, in addition, becomes inflexible as new data become available through time. That is, if a new dataset becomes available, any previously derived centralized model would need to be retrained, either completely from scratch, or by relying on transfer-learning methods. Regardless, re-learning can be expensive and, eventually, it is exposed to catastrophic forgetting risk [37]. Furthermore, privacy and ethical problems quickly arise when dealing with potentially sensitive information, as it is the case in the clinical domain. This may prevent data exchange between different centers.

In contrast with these methods, decentralized learning strategies represent an interesting alternative to be used in the context of restrictive data-sharing scenarios. One such possibility is to train machine-learning models locally within each data source location, and then integrate the resulting models using an ensemble. Such shows advantages regarding flexibility and scalability of the design, for which the resulting ensemble can be easily expanded by adding new local models when new training data or datasets become available without the need of re-training from scratch. Furthermore, because each model integrating the ensemble has been locally developed in the context of its data source, there is no need of sharing and/or centralizing data from different centers. Only the resulting local model parameters (i.e. weights) would need to be shared for their integration in the final ensemble. Therefore, potential issues due to patient privacy protection regulations are minimized. This approach has been explored on a recent work by the authors, with preliminary results also suggesting that more robust inter-database generalization can be achieved in comparison to individual models derived from single source datasets [38]. These preliminary results are reviewed and expanded in this work. More specifically, in past experimentation, direct comparison to centralized-based approaches was missing. Moreover, ensemble combination was only considered by assuming a majority voting strategy.

Alternatively, recent progress in the area of federated learning is opening interesting new paths of development. More specifically, the federated approach is based on the idea of collaboratively training a learning model across multiple participating nodes, holding decentralized local samples, without the necessity of exchanging their data [39]. Instead, individual client nodes from different geographic locations would exchange local model parameters or aggregated non-sensitive information, therefore preventing sensitive raw data from being directly shared. An interesting property of federated learning, that contrasts with other distributed learning approaches, is that it does not assume client data to be mutually independent and identically distributed (IID) across the participating nodes. This is a relevant assumption in the clinical setting, where the representative patient-phenotype would presumably diverge across different medical centers. Federated learning has been barely examined in the context of sleep medicine. That we know of, only one recent work has considered this approach, nevertheless in which the corresponding client data were simulated by partitioning one single dataset [40]. As stated before, this approach involves considerable relaxation of the non-IID assumption, neither does it allow proper evaluation of actual generalization capabilities of the proposed solution.

In the light of the above observations, in this work we investigate the use of different decentralized deep-learning approaches based on the two previously described scenarios, ensemble and federated learning, and explore their utility in the context of automatic sleep staging. The main objective is to develop predictive models with robust inter-database generalization capabilities while, at the same time, overcoming limitations due to exchange and centralization of sensitive information. As novel contribution, we expand preliminary work on ensemble learning [38], [41]. First, by including direct comparison to centralized-based approaches. Second, by exploring four different ensemble combination strategies, namely max-voting, output averaging, size-proportional, and Nelder-Mead model weighting approaches. In addition, we explore the use of federated learning and present a new variant of the more general federated stochastic gradient descent (FedSGD) approach [42], namely Sub-sampled Federated SGD (ssFedSGD). Inter-database generalization performance from each of these methods is examined on a leaving-one-database-out direct-transfer scenario using six independent and heterogeneous sleep staging databases collected from public online repositories. For setting up a baseline for comparative analysis, the obtained results are also compared against traditional approaches consisting on training individual models i) on each of the local datasets, and ii) on the centralized dataset that results from gathering together data from the individual cohorts.

Based on the results of our experimentation, we analyze and discuss the advantages and disadvantages of each of the explored approaches.

II. Materials and Methods

In this section we describe the two proposed decentralized learning approaches, ensemble and federated learning, including the specific explored variants on each case. In addition, we detail the general deep neural network architecture and the different sleep staging databases used during experimentation.

A. Ensemble Approach

Ensemble comprises the aggregation of several pre-trained local model outputs to produce a final prediction. Intending to expand results from [38], in this work we explore four different output assembly techniques in order to compare their effectiveness with respect to local and centralized approaches (in addition to federated approach, to be described in the next section). More specifically, the following ensemble combination approaches are considered:

- *Max-voting:* each local model integrating the ensemble selects its output class according to the corresponding highest softmax activation at its output layer. The final ensemble prediction corresponds with the most represented class, that is, the most frequently voted among the models composing the ensemble.
- *Output averaging:* in contrast with max-voting, this method averages each of the corresponding output softmax activations of the models participating in the ensemble, prior to individual class assignment. The final resulting prediction corresponds with class associated to the highest averaged value.
- *Size-proportional weighting:* under this approach different weights c_i are assigned to each of the models $M(i)$ integrating the ensemble, proportionally with respect to the corresponding amount of data contained in their local datasets (n_i). Let us denote $N = \sum n_i$ the total amount of virtual data, then $c_i = \frac{n_i}{N}$. The respective output softmax activations are then balanced by multiplying their value with the corresponding coefficient c_i. The output with the highest score is selected as final predicted class.
- *Nelder-Mead:* this method uses a weighted combination of the output softmax activations from each of the models' integrating the ensemble, similar to the previous method. Here, in contrast, the Nelder-Mead optimization algorithm [43] is used to find the best possible weights combination following an iterative process. The loss function of the corresponding ensemble combination evaluated on an ancillary (validation) dataset is used as reference for this purpose.

B. Federated Learning

Federated learning is a machine-learning technique which involves collaborative learning while preserving data-privacy [39]. It applies the General Data Protection Regulation's (GDPR) data minimization principle [44], for which the information transmitted is intended to be the minimal necessary for guiding the targeted data learning process. In particular, it is assumed that the exchanged information is always less than the raw source data, and it does not contain any personal nor potentially sensitive information. From a general perspective, federated learning comprises a global model, so-called "server", which is successively improved by aggregating parametric information from multiple decentralized local nodes, so-called "clients". Let us consider the general optimization problem to be represented as:

$$\min_{w \in \mathbb{R}^d} f(w) = \frac{1}{n} \sum_{i=1}^{n} \ell(x_i, y_i; M(w)),$$

where $\ell(x_i, y_i; M(w))$ denotes the prediction loss on the sample (x_i, y_i) of a global model M, that depends on the set of parameters w. If we assume that n data points are distributed across K decentralized datasets, let us denote P_k as the set of data indexes within the client k, where $k = 1 \ldots K$, then we can reformulate the problem in a federated setting as:

$$\min_{w \in \mathbb{R}^d} f(w) \quad \text{where} \quad f(w) = \frac{1}{n} \sum_{k=1}^{K} \sum_{i \in P_k} \ell(x_i, y_i; M(w)).$$

Notice, with the above general formulation, we are implicitly assuming that data can be non-IID, and imbalanced across the partitioning P. Effectively, several issues might be attended when considering the optimization of this function collaboratively. First, because the non-IID assumption, clients' particular distribution may not be representative of the entire population. Further, as stated, there can be uneven data availability resulting in unbalanced datasets. In addition, the server might be massively distributed, meaning that the number of participant clients might exceed the amount of corresponding local data. And last but not least, clients-server communication might be limited due to temporal unavailability or slow connectivity [42]. Under this setting, the general federated learning workflow, which is shown in Fig. 1, involves three basic steps repeating along an iterative process: i) distribution of the current server state (w_t) to the clients, ii) computation of the local update for each client ($\theta_{k,t}$), and iii) client parameter aggregation and server model state update (w_{t+1}). In general, $\theta_{k,t} = g(P_k; w_t)$, a certain function over the corresponding set of local data points and the current server model parameters. Similarly, the exact aggregation formula needs to be defined leading to different implementation variants of the federated learning algorithm [42], [45], [46], [47]. Importantly, and regardless of the exact formulation, during the local update step, each client computes its $\theta_{k,t}$ independently, using information from its local data source, but without exchanging raw data with any other client nor the server. After each cycle, the process starts over with the server distributing the new updated global state to the participating clients. The procedure stops when a predefined number of learning rounds is reached, or certain specific stopping condition is met.

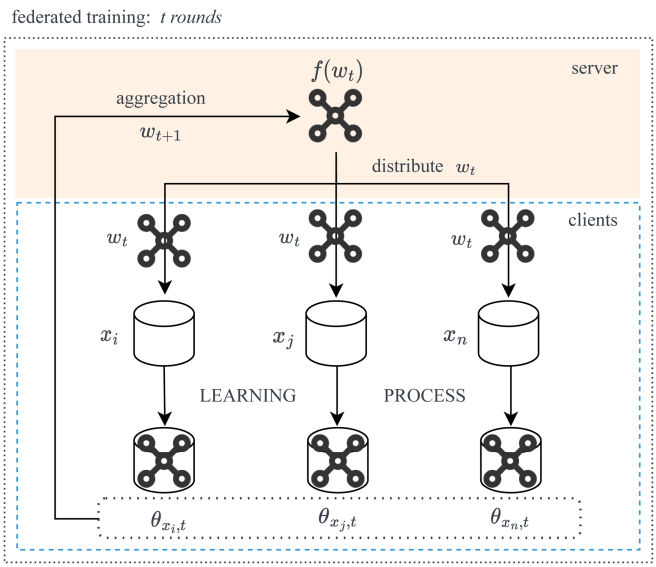


Fig. 1. General Federated learning process. The workflow is divided in t rounds of computation. The current server state (w_t) is distributed to the K clients where the learning process is performed using their corresponding local datasets (x_k). Clients' updated local parameters ($\theta_{k,t}$) are then communicated back to the server, where aggregation and global model state update (w_{t+1}) takes place. The process continues until the predefined number of rounds (t) has been reached.

C. Sub-Sampled Federated Stochastic Gradient Descent (ssFedSGD)

Stochastic gradient descent is the most popular optimization algorithm in (deep) machine-learning [48]. In the federated environment, this optimizer leads to the so-called Federated SGD (FedSGD) algorithm, which applies a single batch gradient descent calculation per round of communication [42].

More specifically, during the learning process in FedSGD, each client computes $\theta_{k,t} = \nabla F_k(w_t)$, where:

$$F_k(w_t) = \frac{1}{n_k} \sum_{i \in P_k} \ell(x_i, y_i; M(w_t)),$$

with $n_k = |P_k|$. That is, $\theta_{k,t}$ represents the average gradient on the local dataset k given the current sever model state w_t. In fact:

$$f(w_t) = \sum_{k=1}^{K} \frac{n_k}{n} F_k(w_t) \quad ; \quad \nabla f(w_t) = \sum_{k=1}^{K} \frac{n_k}{n} \nabla F_k(w_t).$$

The central server thus simply aggregates all the local $\nabla F_k(w_t)$ for which, assuming a fixed learning rate (η), the server state update formula becomes:

$$w_{t+1} \leftarrow w_t - \eta \sum_{k=1}^{K} \frac{n_k}{n} \theta_{k,t}.$$

Notice, under this approach, the averaged local gradients are balanced taking into account the respective amount of data used at each client.

However, one problem with aforementioned FedSGD baseline approach concerns the very large amount of rounds needed for training accurate server models. The associated computation times, in fact, might become unpractical depending on the specific type of application, the amount, or the complexity of data contained on each client [42]. For this reason, many FedSGD variations are emerging aiming to speed up the learning process, and cope with instability due to dissimilar local updates, or the presence of non-IID data [46], [49], [50], [51], [52].

While the matter remains as an open area of research, in this work we propose a new variant, namely Sub-sampled Federated SGD (ssFedSGD) using the above described FedSGD framework as baseline. The main contribution of this method is the use of an arbitrary fixed-length (n_s) sub-sample $S_{k,t}$, by uniformly randomly sampling each client dataset k at the beginning of each federated training round t. Notice, $n_s = |S_{k,t}|, \forall k, t$, where $k = 1 \ldots K$ and $t = 1 \ldots max\_rounds$.

Hence, under ssFedSGD, each client locally computes $\theta'_{k,t} = \nabla F'_k(w_t)$, where:

$$F'_k(w_t) = \frac{1}{n_s} \sum_{i \in S_{k,t} \subseteq P_k} \ell(x_i, y_i; M(w_t)),$$

and the global server state update formula becomes:

$$w_{t+1} \leftarrow w_t - \eta \sum_{k=1}^{K} \frac{n_s}{n_s \times K} \theta'_{k,t} \equiv w_{t+1} \leftarrow w_t - \eta \sum_{k=1}^{K} \frac{\theta'_{k,t}}{K}.$$

Notice that, because of using uniform random sub-sampling, the resulting $S_{k,t}$ still hold the same data distribution as the original P_k's. By selecting the appropriate n_s we thus hypothesize that effective learning can be still achieved at a fraction of the cost per round, therefore, speeding up the overall learning process in practice. Moreover, by using a fixed-length sub-sample we ensure equal client contribution to the global learning at each step, irrespective of the total amount of local data. Notice, in contrast, that in the original FedSGD setting more relative importance is given to the update resulting from the client with the largest dataset. Furthermore, we would avoid possible collateral effects due to disparity of local computation steps among the client node, in particular, if assuming the use of a common batch size.

D. Deep-Learning Model Architecture

We use a convolutional (CNN) long short-term memory (LSTM) deep-learning model architecture based on the general schema proposed in past work [38]. The model was completely re-implemented in Python (version 3.9.7, Tensorflow 2.7.0) with some additional modifications regarding elimination of an artifact removal preprocessing step, and a batch normalization layer in the operation block, which were included in the original design. The latter was motivated by new experimental results in this work showing convergence problems in the federated scenario. This effect will be further analyzed in the discussion section. An overview of the resulting architecture can be seen in Fig. 2. We refer to past work for detailed discussion on the remaining general architectural design [38].

E. Databases

For testing our methods a heterogeneous dataset comprising six independent sleep staging databases (DREAMS, Dublin,

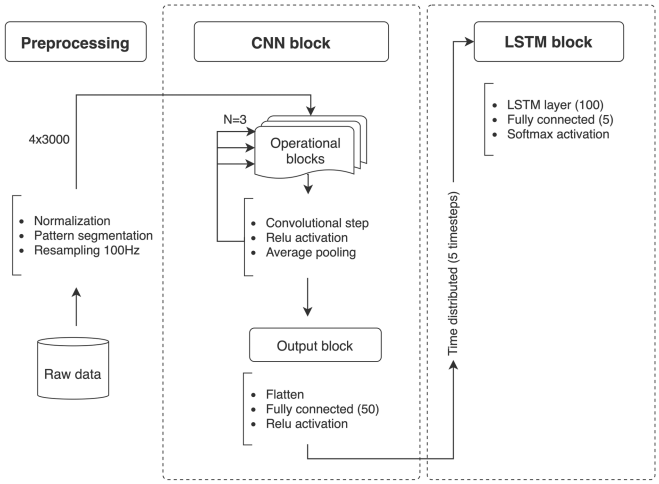


Fig. 2. Preprocessing steps and general CNN-LSTM architecture. The process is divided in three blocks: preprocessing step, convolutional step (CNN block) and time-series dependencies (LSTM block).

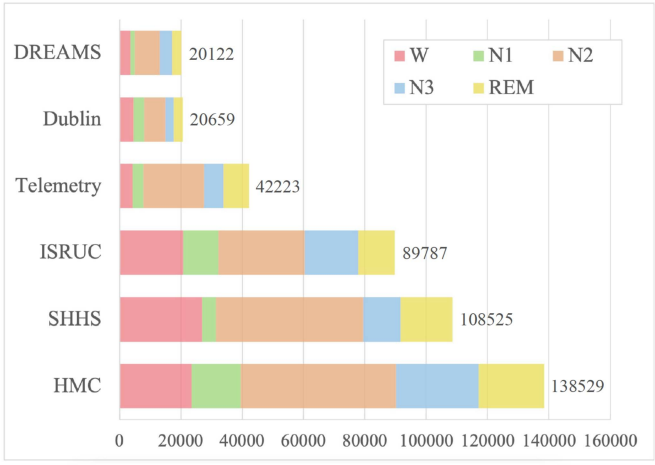


Fig. 3. Amount of sleep stages files used by database.

SHHS, Telemetry, ISRUC and HMC) was used. For the sake of repeatability, all data were collected from public online repositories, digitally encoded using the open EDF(+) format [53]. Fig. 3 summarizes the different number of samples across the six collected databases and their corresponding class distributions, which illustrates the presence of size and class imbalance. Detailed description of each of the databases can be found in past work [38].

III. Experimental Design

The experimental design involves the scheduling of four learning strategies following different local, centralized, and decentralized methods described in the previous section. The purpose is to compare the resulting inter-database generalization performance on the targeted sleep staging prediction task. All experiments use as reference the set of databases mentioned in Section II-E, and the deep neural network architecture referred to in Section II-D. We write TR(x), VAL(x), and TS(x), to refer to the respective training, validation, and testing split partitions resulting from database x. The complete set of data in the corresponding database is denoted as FULL(x). Likewise, we denote M(x) to refer to the model derived from (training) data from dataset x, which can be a single source, or a combination of several databases, depending on the specific experiment as described next. Detailed diagrams of the experiments are shown in Fig. 4.

A. Experiment 1: Local Models

Six different deep-learning models are built. Each M(x) model is trained using data from one single database, i.e. TR(x), using VAL(x) as the corresponding validation set for implementing early stopping. Local generalization performance of the resulting model is evaluated on TS(x), while the actual (database-agnostic) external generalization is assessed on all remaining FULL(i), $i \neq x$ (Fig. 4(a)).

B. Experiment 2: Centralized Database-Combined Models

We built six database-combined models, C_x, by pooling data from five out of the six available databases, following a leaving-one-database-out strategy. In other words, let d be the leaved-out database, the corresponding C_x model is trained using the combined dataset {TR(k), $k = 1...6$, $k \neq d$}. Likewise, the same procedure is followed for arranging the corresponding VAL(x) and TS(x) datasets to implement early stopping and perform local evaluation, respectively. Notice, under this approach, all data but the leaved-out database are locally available. It therefore represents the baseline for the classical centralized learning approach towards which distributed ensemble and federated learning strategies can be compared. Inter-database generalization performance is here evaluated on the leaved-out FULL(d) dataset (Fig. 4(b)).

C. Experiment 3: Ensemble Models

Six ensemble models, E_x, are created by combining five out of the six local models resulting from Experiment 1, thus using a leaving-one-model-out strategy. Let d be the leaved-out model, then $E_x = \{M(k), k = 1...6, k \neq d\}$. In contrast with Experiment 2 involving combined models, this approach is implemented sharing local model's parameters, not local data. External performance for each resulting ensemble is again evaluated on its corresponding leaved out FULL(d) dataset, of which data were not used for derivation of any of the M(k) included in the ensemble. For this purpose the four output assembly strategies described in Section II-A, namely max-voting, output averaging, size-proportional weighting, and Nelder-Mead, are tested and compared (Fig. 4(c)).

D. Experiment 4: Federated Models

We build six different servers, F_x, using the described federated learning approach. As in Experiments 2 and 3, each F_x takes as reference five out of the six gathered databases, leaving one

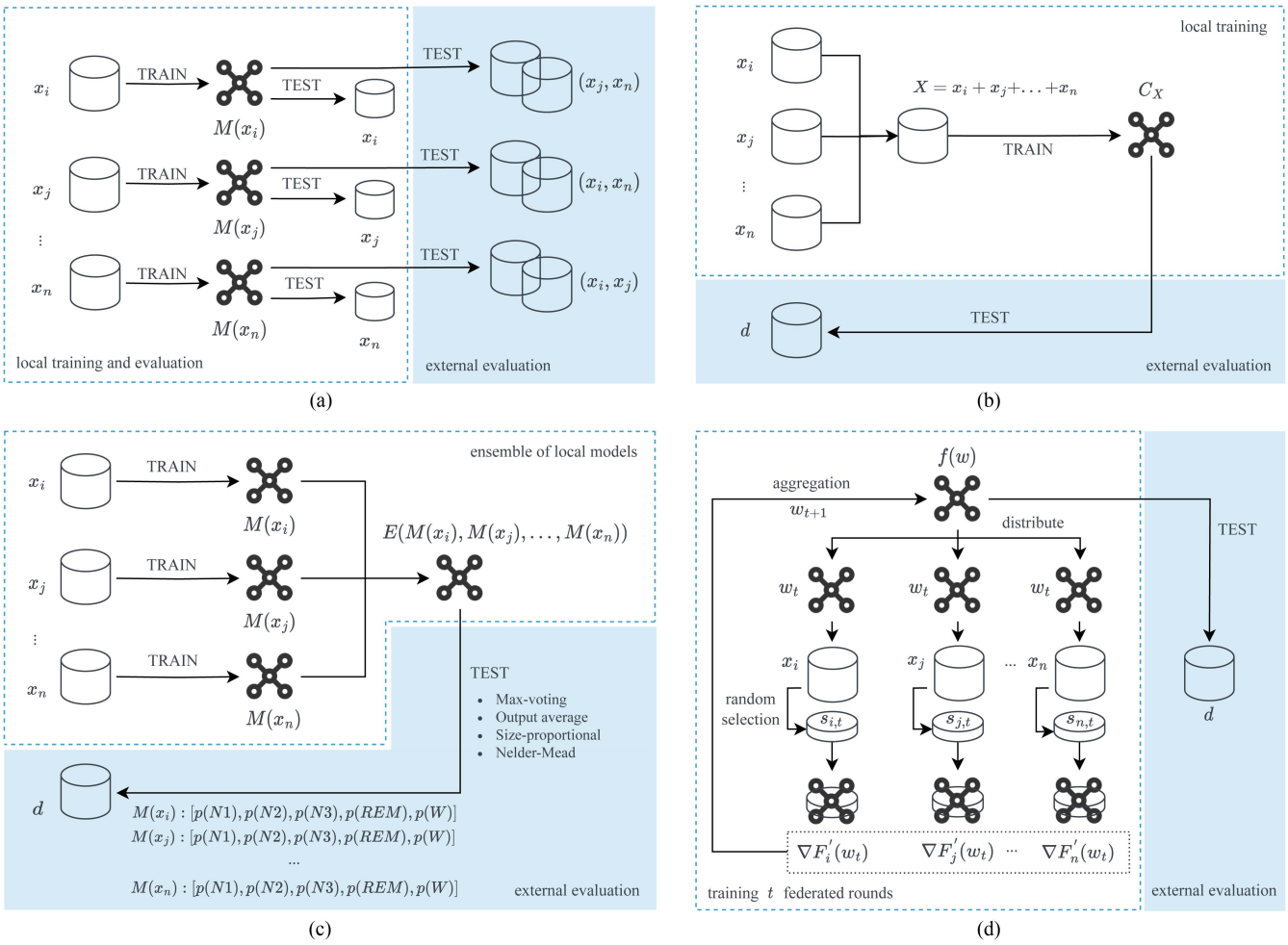
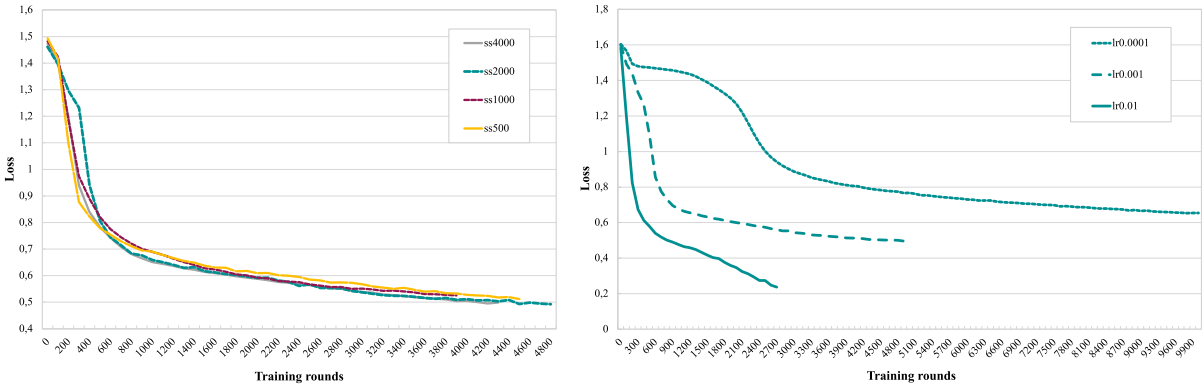


Fig. 4. Diagram of the experimental design. Figures are divided in two panels depending on the performance evaluation location: local (dashed-blue panel) and external (blue-colored panel). (a) Experiment 1: Local models. Local models $M(x)$ are built using $TR(x)$ and $VAL(x)$ datasets split from database source x. Local $TS(x)$ is used for local generalization evaluation. The remaining databases, other than x are used for evaluation of external generalization performance. (b) Experiment 2: Centralized database-combined models. TR and VAL datasets are combined using a leave-one-database-out strategy into a central larger and single dataset. Resulting combined-datasets are then used to train the model C_X while the discarded database d is used to perform external evaluation. (c) Experiment 3: Ensemble models. Local models $M(x)$ are assembled using a leave-one-database-out strategy. On each case the excluded database d is used to evaluate generalization capability. Output activations of each model integrating the ensemble, which correspond to the five sleep stages, are combined for calculation of the final ensemble classification. We explore four approaches namely max-voting, output averaging, size-proportional weighting and Nelder-Mead weight optimization. (d) Experiment 4: federated models. As well as in the other experiments, each federated model is built using a leave-one-database-out strategy. Databases included in federated model, are used as an independent clients where ssFedSGD is used to perform local gradient calculations and server model integration during t rounds. Resulting federated models are tested on the corresponding leaved-out d databases.

out for the purpose of evaluating the corresponding external generalization performance. In order to simulate the federated environment, each of the five used databases is distributed and treated separately as one independent client. Hence, each client $k, k \neq d$ corresponds to one database and uses its own TR(k) dataset to perform the local update step. The proposed ssFedSGD learning algorithm, described in Section II-C, is then used training and deriving the corresponding global model. Early stopping is here implemented using as reference the aggregated performance of the corresponding clients' local validation partitions. For this purpose, each client k locally evaluates its performance on the corresponding VAL(k) dataset using the last communicated server's state. The result is then sent back to the server which, in the general FedSGD scenario, averages the individual client's validation performances proportionally to the respective number of local data samples n_k. Notice that in the context of ssFedSGD equal weighting results as $n_k = |S_{k,t}|, \forall k = 1..K, t = 1..max\_rounds$. Early training stopping then takes place once the number of rounds configured in the patience setting have been surpassed without further improvement over the so-far best validation performance obtained. Finally, as in previous cases, inter-database generalization performance is evaluated on the leaved-out FULL(d) dataset (Fig. 4(d)). Additionally, a set of ablation experiments involving different parameter configurations are conducted in order to analyze their effects on ssFedSGD performance. More specifically, the experiments focused on evaluating the influence of different learning rates and sub-sample sizes.



External evaluation: HMC dataset

	acc	precision	recall	macro-F1 score	cohen kappa
ss4000	0,69	0,69	0,63	0,61	0,58
ss2000	0,70	0,71	0,63	0,62	0,59
ss1000	0,67	0,66	0,61	0,60	0,56
ss500	0,70	0,71	0,63	0,63	0,59

(a)

External evaluation: HMC dataset

	acc	precision	recall	macro-F1 score	cohen kappa
lr0.0001	0,64	0,64	0,56	0,55	0,51
lr0.001	0,70	0,71	0,63	0,62	0,59
lr0.01	0,69	0,68	0,63	0,63	0,59

(b)

Fig. 5. Ablation experiments of the proposed federated algorithm (ssFedSGD) using several parameter settings in F_5 configuration. Results regard loss values during training and accuracy, precision, recall, macro-F1 score and Cohen Kappa metrics when attempting to predict the corresponding external dataset (HMC). (a) Convergence analysis using several sub-sample sizes (500, 1000, 2000 and 4000) and a fixed learning rate (0.001). (b) Convergence analysis of different learning rates (0.0001, 0.001 and 0.01) and a fixed sub-sample (2000).

E. Configuration Settings

For all the above described experiments, the same epoch-wise database partitioning configuration scheme was used. For each local database 20% of their data were set aside as testing dataset, while the remaining 80% were further split following a 80-20 proportion into training and validation data respectively. Results of all experiments were evaluated on the respective datasets using Cohen's Kappa (κ) as the main reference for performance assessment [54]. The multi class and imbalanced nature of the sleep staging scenario makes the use of Kappa a more adequate criterion in this context than other widespread metrics in machine learning, such as accuracy, as the former corrects by agreement due by chance. Remarkably, Kappa is the standard measure of agreement reported across literature regarding human and computer-based inter-rater scoring agreement in the context of sleep studies [3], [5], [21], [22], [23], [24]. Regardless, in order to provide a more complete picture, final experimental summary results will be quantified, in addition, using supplementary widespread evaluation metrics including accuracy, macro-F1 score, precision, and recall. For the learning step, stochastic gradient descent was used with constant learning rate (lr $=$ 0.001) and momentum (p $=$ 0.9). The categorical cross-entropy was selected as the loss function to be minimized. A batch size of 100 was used, and the maximum number of iterations was set high enough so that learning would stop based on the corresponding validation performance, namely early stopping criterion. In this regard a patience of 5 was used for all but Experiment 4 regarding federated learning, were patience parameter of 100 was used. The latter larger patience was required to compensate for the sub-sampling step in ssFedSGD, effectively making it necessary to perform more federated rounds to process a comparable amount of samples as in one regular training epoch in the case of Experiments 1 to 3. More specifically, for ssFedSGD a random sub-sample size $n_s = 2000$ was used for each participating client.

TABLE I
LOCAL MODELS' SELF PERFORMANCE (COHEN'S KAPPA)

CNN-LSTM

Model	TR	VAL	TS
M(Dublin)	0.93	0.84	0.83
M(HMC)	0.86	0.79	0.78
M(Telemetry)	0.93	0.83	0.83
M(DREAMS)	0.88	0.81	0.82
M(SHHS)	0.90	0.82	0.81
M(ISRUC)	0.88	0.80	0.79

Performance of local models M(x) during the training process. TR = Train VAL = Validation; TS = Test partition.

Concerning the Nelder-Mead ensemble combination strategy described in Section III-C, proportional weights were used for initializing the ensemble combination of the participating models at iteration 0. That is, the initial conditions match the output of the additionally tested output averaging strategy. A maximum of 10 optimization iterations rounds were then allowed. At each iteration, the loss function evaluated on the corresponding combined validation dataset was taken as reference to guide the underlying search process.

All experiments were conducted on an Intel Xeon CPU E5-2620 v3 @ 2.40 GHz x 8, equipped with 2 NVIDIA RTXA 6000-48 C GPUs. Source code for reproducibility of our experiments and methods will become available online at Github https://github.com/adrania/Decentralized-deep-learning.git.

IV. RESULTS

Results of the experiments described above are detailed in Tables I, II, III, IV, and V and Fig. 5. Tables I and II refer to

TABLE II
LOCAL MODELS' EXTERNAL PERFORMANCE (COHEN'S KAPPA)

CNN-LSTM

Model	FULL(Dublin)	FULL(HMC)	FULL(Telemetry)	FULL(DREAMS)	FULL(SHHS)	FULL(ISRUC)	Model average
M(Dublin)	-	0.50	0.57	0.47	0.50	0.46	0.50
M(HMC)	0.55	-	0.65	0.48	0.62	0.60	0.58
M(Telemetry)	0.40	0.41	-	0.43	0.40	0.36	0.40
M(DREAMS)	0.44	0.40	0.38	-	0.47	0.52	0.44
M(SHHS)	0.46	0.53	0.14	0.60	-	0.63	0.47
M(ISRUC)	0.56	0.52	0.48	0.70	0.58	-	0.57
Database average	0.48	0.47	0.44	0.54	0.51	0.51	

Performance of local models on predicting FULL(i) external databases. FULL = Full dataset.

TABLE III
DATABASE-COMBINED MODELS' PERFORMANCE (COHEN'S KAPPA)

CNN-LSTM

Model	TR	VAL	TS	Test database	FULL
C1	0.89	0.79	0.79	ISRUC	0.67
C2	0.89	0.78	0.78	SHHS	0.64
C3	0.88	0.79	0.79	DREAMS	0.71
C4	0.85	0.78	0.78	Telemetry	0.67
C5	0.87	0.80	0.79	HMC	0.61
C6	0.86	0.79	0.78	Dublin	0.63
				Average	0.66

C_x references the model combination which involves TR, VAL and TS data from all databases but the external "Test database" indicated in the fourth column. TR = Training; VAL = Validation; TS = self test partition; FULL = Full dataset from the discarded database.

TABLE IV
ENSEMBLE MODELS' PERFORMANCE (COHEN'S KAPPA)

CNN-LSTM

		Max-voting	Average	Size-proportional	Nelder
Model	Test database	FULL	FULL	FULL	FULL
E1	ISRUC	0.59	0.60	0.65	0.61
E2	SHHS	0.63	0.65	0.66	0.66
E3	DREAMS	0.66	0.67	0.65	0.67
E4	Telemetry	0.54	0.60	0.64	0.60
E5	HMC	0.56	0.59	0.60	0.59
E6	Dublin	0.57	0.61	0.63	0.62
	Average	0.59	0.62	0.64	0.63

E_x references the ensemble combination which includes all local models M(x) but the one trained with the "Test database" indicated in the second column. FULL = Full dataset. All results regard external database evaluations.

TABLE V
FEDERATED MODELS' PERFORMANCE (COHEN'S KAPPA)

CNN-LSTM

Model	TR	VAL	TS	Test database	FULL
F1	0.78	0.76	0.76	ISRUC	0.66
F2	0.77	0.75	0.75	SHHS	0.67
F3	0.74	0.73	0.73	DREAMS	0.70
F4	0.74	0.74	0.73	Telemetry	0.64
F5	0.75	0.74	0.74	HMC	0.59
F6	0.74	0.73	0.73	Dublin	0.64
				Average	0.65

F_x references the federated combination which includes as clients all the databases but the one indicated in "Test database" column. TR = Training; VAL = validation; TS = Testing set; FULL = Full data referred to the database in "Test database" column.

results of Experiment 1, showing performances of local models on their respective datasets (including training, validation and test partitions) and on the external FULL databases, respectively, the latter meant to assess their generalization capabilities on the direct-transfer scenario. Comparing Tables I and II, it can be seen that when local test partitions are evaluated, performance ranges between $\kappa = 0.79$ to 0.83. Notice that reasonable generalization is achieved with respect to their corresponding training and validation partitions. However, when an external data source is used, generalization performance decreases to ranges between $\kappa = 0.14$ to 0.70 ($\kappa = 0.40$ to 0.58, on average, per model). The best scenario corresponds to M(ISRUC) in FULL(DREAMS) with $\kappa = 0.70$, the worst corresponding to M(SHHS) predicting FULL(Telemetry). Overall, M(HMC) is the best generalizing model (averaged κ, $\kappa_{avg} = 0.58$, across all leaved-out external databases), followed by M(ISRUC) with $\kappa_{avg} = 0.57$. In contrast, the worst generalization capability corresponds to M(Telemetry), with $\kappa_{avg} = 0.40$. Database-wise, DREAMS seems to be the easiest database to be predicted ($\kappa_{avg} = 0.54$) while Telemetry results the most difficult one ($\kappa_{avg} = 0.44$).

Table III details results of Experiment 2, regarding centralized database-combined models. Similarly to local models, the corresponding train, validation and test performances are reported on the first columns. For each combined model (C_x), inter-database generalization is assessed on the corresponding leaved-out FULL(i) as described in III. This database is identified in the fifth column of Table III. A similar performance downgrading effect can be observed as in the case of local models, whereby performance on the local TS dataset shows higher and more stable results ($\kappa = 0.78$ to 0.79) than in the corresponding external databases ($\kappa = 0.61$ to 0.71). Performance drop in this case is more contained, as expected, due to the higher amount and heterogeneity of training data used. The best scenario is obtained by (C_3) which uses DREAMS as the predicting database ($\kappa = 0.71$), in line with results observed for Experiment 1, while the worst generalization is observed for (C_5), predicting HMC ($\kappa = 0.61$).

Ensemble models' results are described in Table IV. Generalization performances obtained for each of the corresponding external FULL datasets and the different tested combination strategies can be compared. Overall, the best results are achieved using the size-proportional weighting strategy ($\kappa_{avg} = 0.64$), followed by Nelder-Mead ($\kappa_{avg} = 0.63$), output averaging ($\kappa_{avg} = 0.62$) and lastly, max-voting ($\kappa_{avg} = 0.59$). Size-proportional weighting approach also shows the most stable results ranging $\kappa = 0.60$ to 0.65 across databases.

TABLE VI
SUMMARY OF AVERAGE GENERALIZATION PERFORMANCE (COHEN'S KAPPA)

CNN-LSTM

Test database	Local	Combined	Max-voting	Averaging	Size-proportional	Nelder	Federated
ISRUC	0.51	0.67	0.59	0.60	0.65	0.61	0.67
SHHS	0.51	0.64	0.63	0.65	0.66	0.66	0.67
DREAMS	0.54	0.71	0.66	0.67	0.65	0.67	0.70
Telemetry	0.44	0.67	0.54	0.60	0.64	0.60	0.64
HMC	0.47	0.61	0.56	0.59	0.60	0.59	0.59
Dublin	0.48	0.63	0.57	0.61	0.63	0.62	0.64
Average	0.49	0.66	0.59	0.62	0.64	0.63	0.65

Performance results in regard to federated models are detailed in Table V. As in Experiments 2 and 3, each federated model F_x is built using a leave-one-database-out strategy, while in this case, the remaining databases used for developing the model are distributed and treated as independent local clients. In order to assay the resulting local database performance, the final server state is independently evaluated, on each client, using their corresponding local train, validation, and test partitions. These metrics are finally averaged resulting in TR, VAL and TS values shown in the the corresponding columns of Table V. Generalization performance evaluated on the corresponding leaved-out FULL(i) database is indicated in the last column. From the Table it can be observed that the range of obtained metric values varies with $\kappa = 0.59$ to 0.70. In comparison with local testing set, a similar downgrading trend can be observed when predicting external data. The effect is similar to the one observed in Table III with regard to centralized database-combined models, with perhaps slight less local over-fitting in this case. On an individual basis, the best generalization was obtained by F_3 ($\kappa = 0.70$), corresponding to the prediction of the DREAMS database. Likewise, the worst scenario was represented by F_5 when attempting to predict HMC data ($\kappa = 0.59$). Additionally, intending to analyze the effects of different parameter setting in ssFedSGD convergence and generalization, we conducted several ablation experiments regarding different sub-sampling sizes and learning rates. We evaluate four sub-sampling scenarios (500, 1000, 2000 and 4000) and three different learning rates (0.0001, 0.001 and 0.01) using F_5 configuration as baseline. The results of these experiments are shown in Fig. 5. As it can be seen in Fig. 5(a)) no convergence variations are observed when using different sub-sample sizes, neither on the corresponding generalization results as similar performances are obtained when these models are presented to the external database ($\kappa = 0.56$–0.59). Notice that model ss1000 is the one that converges earliest (round $\approx$ 3800), however, it has presented the worst results when predicting HMC data ($\kappa = 0.56$). Regarding learning rate effects, the results can be seen in Fig. 5(b)). Differences in speed convergence appeared, as expected, with normal behavior since the higher learning rate, the earlier the convergence and vice versa. Notice that variations in convergence speed among different settings, specifically the number of learning rounds or iterations, occur naturally when applying a consistent early stopping criterion (with a patience of 100) for all federated learning experiments. Despite loss differences are displayed, it has no significant variations in generalization performance ($\kappa = 0.51$–0.59).

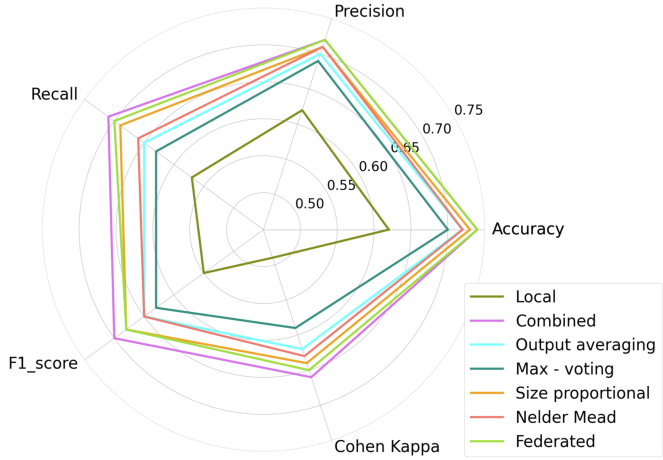


Fig. 6. Overview of the generalization performance metrics (accuracy, precision, recall, macro-F1 score and Cohen Kappa) by explored strategy. Displayed results regard the averaged performances of each approach when predicting full external datasets.

Finally, for easy comparison of the different tested methods, Table VI and Fig. 6 show a summary of the averaged generalization performances obtained on each case. VI regards Cohen Kappa results and Fig. 6 details the complete performance picture involving accuracy, precision, recall, macro-F1 score and Kappa metrics. As it can be seen, the federated model practically matches the classical database centralization approach with respective $\kappa = 0.65$ and 0.66. This result is followed by the size-proportional ensemble combination strategy with $\kappa = 0.64$, and Nelder-Mead with $\kappa = 0.63$. All decentralized ensemble and federated approaches obtain considerable better generalization results than those of models derived from local databases.

V. DISCUSSION

In this work we have explored several machine-learning strategies applied to the medical domain of sleep staging. More specifically, special focus was targeted toward assessment of generalization robustness in the context of a multi-database prediction scenario, and the preservation of local data-privacy.

We have proposed different decentralized learning approaches, namely ensembles and federated learning, whereby global model development can take place taking advantage of heterogeneous data distributed across independent decentralized nodes. Remarkably, these approaches avoid direct sharing of local raw data contained on each node, therefore allowing different medical centers (in this case) to contribute without

unnecessary exposure of potentially sensitive or restricted information. We explored different configurations of such learning approaches, using a deep-learning neural network architecture as reference, and studied their generalization capabilities on the prediction of several independent and heterogeneous sleep staging databases. Traditional approaches that derive local models from individual databases, or from centralizing data from multiple cohorts, were submitted to the same experimental benchmarking for baseline comparison of the resulting inter-database generalization performances.

Our experimentation has evidenced that performance of local models is influenced by the specifics of the source dataset, leading to poor inter-database generalization performance. In particular, while comparable performance between local validation and test partitions were observed, considerable performance drop was experienced when local models were targeted the prediction of data from external sources (see Tables I and II). Similar performance downgrading effects were reported in recent literature regarding the same automatic sleep staging scenario [16], [17], [18], [19], [20]. Our current results confirm this trend. The effect is directly related to the aforementioned database variability due to differences in patient's conditions and physiology, signal acquisition methods, or intra- and inter-expert scoring interpretations [18]. Thereunder, local models are biased to the specific training source domain, providing unreliable results when being evaluated external data, leading to the need of retraining the model from scratch. Transfer-learning has been recently explored as a means to mitigate the amount of required retraining by focusing on fine-tuning only certain parts of the model, or the use minimal subsets of targeted domain data [26], [27], [28], [29], [30], [31], [32], [33]. However, certain amounts of target data and model re-parameterization are still needed. Alternatively, one might opt to centralize data from different cohorts to develop more robust prediction models [35], [36]. Our experimentation does also confirm this result. Overall, models derived from combined datasets have achieved the best generalization performance ($\kappa = 0.66$, see Table VI). This is expected as it is well-known that an enlarged and more casuistic-rich dataset will contribute to reduce database dependency. However, as introduced before, this approach has several disadvantages. First, all data need to be centralized in a single dataset which might be technically difficult and cost-expensive. Second, data sharing might be problematic if involving sensitive information, as in the medical domain. Third, this strategy is inflexible and does not escalate well if new data becomes available through time, therefore still requiring to re-adapt the model or re-train completely from scratch.

In contrast, decentralized methods proposed in this work provide a promising alternative to cope with the aforementioned problems. In this regard, our experimental data have shown that better generalization can be achieved in comparison with local models. Moreover, similar performance can be obtained as with respect to the centralized database-combination approach (see Table VI).

Model ensemble has been introduced as a suitable option in terms of scalability and dynamism. It address catastrophic forgetting as the resulting global model can be easily enlarged by just adding new local models derived from newly available data sources. Integrity of previously integrated models, therefore, remains intact. Moreover, ensemble naturally addresses data-privacy problems, as it is the model, and not data, what is shared to build the ensemble. Likewise, less memory and computational resources are needed for the implementation of individual local training in contrast with combined database models. One caveat here regards the Nelder-Mead approximation, which is not fully database-independent. This is because Nelder-Mead algorithm uses the combined VAL datasets of models integrating the ensemble to guide the weights optimization search. In this regard, it resembles more as a sort of transfer-learning approach, where partial data (i.e. validation dataset) is used for guiding the weights combination fine-tuning. Similar to transfer-learning, flexibility and data-privacy are therefore compromised in this case. In contrast, target domain data (leaved-out dataset) remains completely independent. Regardless, notice that the other three ensemble combination strategies analyzed in this study are not subject to such limitations. Moreover, size-proportional weighting shows the best performance results overall.

A further step in decentralized strategies research involves the study of federated learning. As mentioned above, federated scenario hosts collaborative and distributed training without sharing of local sensitive information. Applicability of this technique within the field of sleep medicine has been barely investigated. The only study that we know of [55] simulated a federated environment by partitioning one database (Sleep-EDF [56]) among different nodes. This approach hence violates the non-IID assumption, nor does it allow assessment of inter-database generalization capabilities of the resulting model. To extend knowledge in this area, we have experimented on a federated scenario involving six independent and heterogeneous sleep staging databases. In addition, we have proposed a new federated learning algorithm, namely ssFedSGD, as a variant of the baseline FedSGD approach. Our fist attempt, was in fact to directly apply FedSGD in the context of our problem. However, we experienced one of the main reported limitations regarding this method, that is, the very large amount of required rounds to achieve effective training convergence [42]. More specifically, with our described setting and available hardware resources, experimental times by using this method became intractable with estimations above the six-months of uninterrupted computation. We also experimented with alternative approaches aimed to solve aforementioned FedSGD problems, in particular Federated Averaging (FedAvg) [42]. Our experiments with FedAvg, however, were also unsuccessful. In particular, we were unable to achieve stable convergence trend during the federated training. We could speculate with the possibility that the unbalanced nature and non-IID properties of our experimental cohort came into detriment of applicability of this algorithm. More specifically, we hypothesize that because the aggregation step in FedAvg involves local model weights (instead of gradients) together with the fact that disparate amount local learning updates occur on the same learning round (due to the differences in size for each database) leads to quick misalignment of the local models in the parameter space. Literature, in fact, has reported on certain limitations of this approach depending on the specific node data distribution [49], [50], [52], [57]. In this scenario, we have proposed ssFedSGD, whereby using a fixed-length client sub-sample on each round, we enforce equal client contribution to the global learning, regardless of the specific amount of local

data contained on each node. By using this approach we were able to speed up computation time per round with respect to baseline FedSGD by a factor of 17.5 x, enabling tractability of the problem with the same setting and computational resources. Moreover, our ablation experiments have demonstrated that varying the sub-sample size and learning rate has no significant effects on the generalization performance, therefore, these parameters can be conveniently fine-tuned according to clients' requirements. At the same time, we avoided aforementioned problematic related to FedAvg. Our experimental data confirms the intuition, as we show that similar inter-database generalization performance can be achieved as with respect to baseline database centralized approach.

One final remark with regard to the federated learning scenario concerns the usage of batch normalization layers in the architecture of the underlying deep neural network learning model. The original version of our design, in fact, included the use of this layer following the output of the average pooling operation in each of the three operational blocks at the CNN block [38]. In the context of non-federated experiments (local, combined, and ensemble models) the use of this layer was able speed up the training process by reducing the number of learning epochs necessary to achieve model convergence. Deep-learning literature has extensively discussed the general benefits of the input layer normalization provided by this method [58]. While experimental results including batch normalization are omitted due to length restrictions, it is worth mentioning that no improvement on the local or external generalization performance among these methods was observed. However, in the federated scenario, we experienced convergence problems when including this layer in our deep-learning pipeline. It is possible that the non-IID properties of our data, involving different databases, cause weights and bias of batch normalization layer to be affected by local offset, which might penalize client parameter aggregation at the server. Similar problems have also been reported in the literature [46], [47], [49], [59], however the exact implications still remain uncertain, and debate around inclusion of batch normalization on federated learning is considered an open area of research. In light of the experimental data, for our final design described in Section II-D, we finally excluded the use of this layer.

Further investigation is needed for better understanding of some of the reported effects. One interesting line of future work will be to explore additional federated learning algorithms recently proposed in literature to address some of the described convergence problems [49], [50], [59], [60]. Future development should also incorporate additional realistic assumptions to the experimental setting such as time-varying and unobservable content in designing the distributed learning approach. One possible limitation our current work concerns the assumption of a rather static environment where clients (i.e. hospital centers) are assumed to be remain stable through time. While this might be a valid assumption in the context of in-hospital PSG, it might compromise applicability in the context of a more dynamical setting related to mobile and edge-devices (e.g. sleep apps or wearables). More in general, in the context of the rapid progress that takes place in the area, future work should also reassess optimality of the baseline deep learning model used as reference in this work. Likewise, we would also like to extend the proposed methods and experimentation to other applied domains beyond sleep staging to check generalization of our results.

VI. Conclusion

With our current setting, we conclude that the use of decentralized learning methods outperform local methods in terms of inter-database generalization, and provide similar results to centralized learning, however with additional advantages concerning scalability, flexibility, and data-privacy protection. Ensemble learning with size-proportional weighting, in particular, shows a good compromise between generalization performance and simplicity of the method. The reported federated approach shows slightly improved performance, and similar advantages regarding privacy preservation, however it usually involves more cumbersome communication infrastructure and training convergence. Further work and experiments have to be performed in order to confirm generalization of these results.

Acknowledgment

Authors would like to thank CITIC-UDC researches Alberto Pedro Manzano Herrero and Mateo Gende Lozano for their constructive feedback and stimulating discussions.

References

[1] R. Berry et al., "The AASM manual for the scoring of sleep and associated events: Rules, terminology and technical specifications (version 2.6)," Ph.D. dissertation, American Acad. Sleep Med., Darien, IL, USA, 2020.
[2] M. Younes, J. Raneri, and P. Hanly, "Staging sleep in polysomnograms: Analysis of inter-scorer variability," *J. Clin. Sleep Med.*, vol. 12, no. 06, pp. 885–894, Jun. 2016.
[3] C. Berthomier et al., "Exploring scoring methods for research studies: Accuracy and variability of visual and automated sleep scoring," *J. Sleep Res.*, vol. 29, no. 5, Oct. 2020, Art. no. e12994.
[4] D. Alvarez-Estevez and R. M. Rijsman, "Computer-assisted analysis of polysomnographic recordings improves inter-scorer associated agreement and scoring times," *PLoS One*, vol. 17, no. 9, Sep. 2022, Art. no. e0275530.
[5] J. P. Bakker et al., "Scoring sleep with artificial intelligence enables quantification of sleep stage ambiguity: Hypnodensity based on multiple expert scorers and auto-scoring," *Sleep*, vol. 46, no. 2, Jul. 2022, Art. no. zsac154.
[6] H. Sun et al., "Large-scale automated sleep staging," *Sleep*, vol. 40, no. 10, Oct. 2017, Art. no. zsx139.
[7] A. Supratak, H. Dong, C. Wu, and Y. Guo, "DeepSleepNet: A model for automatic sleep stage scoring based on raw single-channel EEG," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 25, no. 11, pp. 1998–2008, Nov. 2017.
[8] H. Phan, F. Andreotti, N. Cooray, O. Y. Chén, and M. De Vos, "SeqSleepNet: End-to-end hierarchical recurrent neural network for sequence-to-sequence automatic sleep staging," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 27, no. 3, pp. 400–410, Mar. 2019.
[9] H. Zhu et al., "MaskSleepNet: A cross-modality adaptation neural network for heterogeneous signals processing in sleep staging," *IEEE J. Biomed. Health Inform.*, vol. 27, no. 5, pp. 2353–2364, May 2023.
[10] W. Zhou et al., "A lightweight segmented attention network for sleep staging by fusing local characteristics and adjacent information," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 31, pp. 238–247, 2022.
[11] D. Zhou et al., "SingleChannelNet: A model for automatic sleep stage classification with raw single-channel EEG," *Biomed. Signal Process. Control*, vol. 75, 2022, Art. no. 103592.
[12] D. Zhou et al., "Alleviating class imbalance problem in automatic sleep stage classification," *IEEE Trans. Instrum. Meas.*, vol. 71, 2022, Art. no. 4006612.
[13] M. Younes, "The case for using digital EEG analysis in clinical sleep medicine," *Sleep Sci. Pract.*, vol. 1, no. 1, Dec. 2017, Art. no. 2.
[14] L. Fiorillo et al., "Automated sleep scoring: A review of the latest approaches," *Sleep Med. Rev.*, vol. 48, Dec. 2019, Art. no. 101204.
[15] H. Phan and K. Mikkelsen, "Automatic sleep staging of EEG signals: Recent development, challenges, and future directions," *Physiol. Meas.*, vol. 43, no. 4, Apr. 2022, Art. no. 04TR01.

[16] S. Biswal, H. Sun, B. Goparaju, M. B. Westover, J. Sun, and M. T. Bianchi, "Expert-level sleep scoring with deep neural networks," *J. Amer. Med. Inform. Assoc.*, vol. 25, no. 12, pp. 1643–1650, Dec. 2018.

[17] L. Zhang, D. Fabbri, R. Upender, and D. Kent, "Automated sleep stage scoring of the sleep heart health study using deep neural networks," *Sleep*, vol. 42, no. 11, Oct. 2019, Art. no. zsz159.

[18] D. Alvarez-Estevez and I. Fernández-Varela, "Addressing database variability in learning from medical data: An ensemble-based approach using convolutional neural networks and a case of study applied to automatic sleep scoring," *Comput. Biol. Med.*, vol. 119, 2020, Art. no. 103697.

[19] A. N. Olesen, P. Jørgen Jennum, E. Mignot, and H. B. D. Sorensen, "Automatic sleep stage classification with deep residual networks in a mixed-cohort setting," *Sleep*, vol. 44, no. 1, Jan. 2021, Art. no. zsaa161.

[20] M. Cesari et al., "Interrater sleep stage scoring reliability between manual scoring from two European sleep centers and automatic scoring performed by the artificial intelligence–based Stanford-STAGES algorithm," *J. Clin. Sleep Med.*, vol. 17, no. 6, pp. 1237–1247, Jun. 2021.

[21] C. Whitney et al., "Reliability of scoring respiratory disturbance indices and sleep staging," *Sleep*, vol. 21, pp. 749–757, 1998.

[22] H. Danker-hopfe et al., "Interrater reliability for sleep scoring according to the rechtschaffen & kales and the new AASM standard," *J. Sleep Res.*, vol. 18, no. 1, pp. 74–84, 2009.

[23] R. S. Rosenberg and S. Van Hout, "The american academy of sleep medicine inter-scorer reliability program: Sleep stage scoring," *J. Clin. Sleep Med.*, vol. 9, no. 1, pp. 81–87, 2013.

[24] H. Van Gorp, I. A. M. Huijben, P. Fonseca, R. J. G. van Sloun, S. Overeem, and M. M. Van Gilst, "Certainty about uncertainty in sleep staging: A theoretical framework," *Sleep*, vol. 45, no. 8, Aug. 2022, Art. no. zsac134.

[25] M. Wang and W. Deng, "Deep visual domain adaptation: A survey," *Neurocomputing*, vol. 312, pp. 135–153, 2018.

[26] S. Chambon, M. N. Galtier, and A. Gramfort, "Domain adaptation with optimal transport improves EEG sleep stage classifiers," in *Proc. IEEE Int. Workshop Pattern Recognit. Neuroimag.*, 2018, pp. 1–4.

[27] S. Nasiri and G. D. Clifford, "Attentive adversarial network for large-scale sleep staging," in *Proc. 5th Mach. Learn. Healthcare Conf.*, 2020, pp. 457–478.

[28] A. Guillot and V. Thorey, "RobustSleepNet: Transfer learning for automated sleep staging at scale," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 29, pp. 1441–1451, 2021.

[29] N. Banluesombatkul et al., "MetaSleepLearner: A pilot study on fast adaptation of bio-signals-based sleep stage classifier to new individual subject using meta-learning," *IEEE J. Biomed. Health Inform.*, vol. 25, no. 6, pp. 1949–1963, Jun. 2021.

[30] H. Phan et al., "Towards more accurate automatic sleep staging via deep transfer learning," *IEEE Trans. Biomed. Eng.*, vol. 68, no. 6, pp. 1787–1798, Jun. 2021.

[31] J. Fan et al., "Unsupervised domain adaptation by statistics alignment for deep sleep staging networks," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 30, pp. 205–216, 2022.

[32] E. Eldele et al., "ADAST: Attentive cross-domain EEG-Based sleep staging framework with iterative self-training," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 7, no. 1, pp. 210–221, Feb. 2023.

[33] C. Yoo, H. W. Lee, and J.-W. Kang, "Transferring structured knowledge in unsupervised domain adaptation of a sleep staging network," *IEEE J. Biomed. Health Inform.*, vol. 26, no. 3, pp. 1273–1284, Mar. 2022.

[34] M. Abou Jaoude et al., "Expert-level automated sleep staging of long-term scalp electroencephalography recordings using deep learning," *Sleep*, vol. 43, no. 11, 2020, Art. no. zsaa112.

[35] J. Stephansen et al., "Neural network analysis of sleep stages enables efficient diagnosis of narcolepsy," *Nature Commun.*, vol. 9, pp. 1–15, 2018.

[36] M. Perslev, S. Darkner, L. Kempfner, M. Nikolic, P. J. Jennum, and C. Igel, "U-Sleep: Resilient high-frequency sleep staging," *NPJ Digit. Med.*, vol. 4, no. 1, Dec. 2021, Art. no. 72.

[37] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," *Psychol. Learn. Motivation*, vol. 24, pp. 109–165, 1989.

[38] D. Alvarez-Estevez and R. M. Rijsman, "Inter-database validation of a deep learning approach for automatic sleep scoring," *PLoS One*, vol. 16, no. 8, 2021, Art. no. e0256111.

[39] C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li, and Y. Gao, "A survey on federated learning," *Knowl.-Based Syst.*, vol. 216, 2021, Art. no. 106775.

[40] A. Saeed, F. D. Salim, T. Ozcelebi, and J. Lukkien, "Federated self-supervised learning of multisensor representations for embedded intelligence," *IEEE Internet Things J.*, vol. 8, no. 2, pp. 1030–1040, Jan. 2021.

[41] A. Anido-Alonso and D. Alvarez-Estevez, "Analysis of ensemble-combination strategies for improving inter-database generalization of deep-learning-based automatic sleep staging," in *Proc. IEEE-EMBS Int. Conf. Biomed. Health Inform.*, 2022, pp. 01–06.

[42] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2017, pp. 1273–1282.

[43] J. A. Nelder and R. Mead, "A simplex method for function minimization," *Comput. J.*, vol. 7, no. 4, pp. 308–313, 1965.

[44] G. D. P. Regulation, "General data protection regulation (GDPR)," *Official J. Eur. Union*, pp. 1–88, 2016. [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679

[45] H. Kim, J. Park, M. Bennis, and S.-L. Kim, "Blockchained on-device federated learning," *IEEE Commun. Letters*, vol. 24, no. 6, pp. 1279–1283, Jun. 2020, doi: 10.1109/LCOMM.2019.2921755.

[46] M. Andreux, J. O. du Terrail, C. Beguier, and E. W. Tramel, "Siloed federated learning for multi-centric histopathology datasets," in *Proc. 2nd Med. Image Comput. Comput. Assist. Interv. Workshop Domain Adapt. Representation Transfer, Distrib. Collaborative Learn., 1st Med. Image Comput. Comput. Assist. Interv. Workshop, Held Conjunction*, 2020, pp. 129–139.

[47] X. Li, M. Jiang, X. Zhang, M. Kamp, and Q. Dou, "FedBN: Federated learning on non-iid features via local batch normalization," in *Proc. Int. Conf. Learn. Representations*, 2021, pp. 1–27. [Online]. Available: https://openreview.net/forum?id=6YEQUn0QICG

[48] N. Ketkar and E. Santana, *Deep Learning With Python*, vol. 1. Springer, 2017, pp. 113–132, doi: 10.1007/978-1-4842-2766-4_8.

[49] M. J. Idrissi, I. Berrada, and G. Noubir, "Fedbs: Learning on Non-IID data in federated learning using batch normalization," in *Proc. IEEE 33rd Int. Conf. Tools Artif. Intell.*, 2021, pp. 861–867.

[50] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, "Scaffold: Stochastic controlled averaging for federated learning," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 5132–5143.

[51] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," in *Proc. Int. Conf. Mach. Learn. Syst.*, 2020, pp. 429–450.

[52] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," 2018, *arXiv:1806.00582*.

[53] B. Kemp and J. Olivan, "European data format 'plus' (EDF+), an EDF alike standard format for the exchange of physiological data," *Clin. Neurophysiol.*, vol. 114, pp. 1755–1761, 2003. [Online]. Available: https://www.edfplus.info/

[54] J. Cohen, "A coefficient of agreement for nominal scales," *Educ. Psychol. Meas.*, vol. 20, no. 1, pp. 37–46, 1960.

[55] A. Saeed, F. D. Salim, T. Ozcelebi, and J. Lukkien, "Federated self-supervised learning of multisensor representations for embedded intelligence," *IEEE Internet Things J.*, vol. 8, no. 2, pp. 1030–1040, Jan. 2021.

[56] B. Kemp, A. H. Zwinderman, B. Tuk, H. A. C. Kamphuisen, and J. J. L. Oberyé, "Analysis of a sleep-dependent neural feedback loop: The slow-wave microcontinuity of the EEG," *IEEE Trans. Biomed. Eng.*, vol. 47, no. 9, pp. 1185–1194, Sep. 2000.

[57] S. J. Reddi et al., "Adaptive federated optimization," in *Proc. Int. Conf. Learn. Representations*, 2021, pp. 1–38. [Online]. Available: https://openreview.net/forum?id=LkFG3lB13U5

[58] N. Bjorck, C. P. Gomes, B. Selman, and K. Q. Weinberger, "Understanding batch normalization," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, pp. 1–12, 2018.

[59] Y. Wang, Q. Shi, and T.-H. Chang, "Batch normalization damages federated learning on NON-IID data: Analysis and remedy," in *Proc. IEEE Int. Conf. Acoustics, Speech Signal Process.*, Rhodes Island, Greece, 2023, pp. 1–5.

[60] C. Zheng, S. Liu, Y. Huang, W. Zhang, and L. Yang, "Unsupervised recurrent federated learning for edge popularity prediction in privacy-preserving mobile-edge computing networks," *IEEE Internet Things J.*, vol. 9, no. 23, pp. 24328–24345, Dec. 2022.

Open Access provided by 'Universidade da Coruña/CISUG' within the CRUI-CARE Agreement.