Hyperparameters Optimization for Federated Learning System: Speech Emotion Recognition Case Study

Kateryna Mishchenko*, Samaneh Mohammadi*, Mohammadreza Mohammadi*, and Sima Sinaei*
* *RISE Research Institute of Sweden*, Sweden
Email:{kateryna.mishchenko, samaneh.mohammadi, mohammadreza.mohammadi, sima.sinaei}@ri.se

***Abstract*—*Context:* Federated Learning (FL) has emerged as a promising, massively distributed way to train a joint deep model across numerous edge devices, ensuring user data privacy by retaining it on the device. In FL, Hyperparameters (HP) significantly affect the training overhead regarding computation and transmission time, computation and transmission load, as well as model accuracy. This paper presents a novel approach where Hyperparameters Optimization (HPO) is used to optimize the performance of the FL model for Speech Emotion Recognition (SER) application. To solve this problem, both Single-Objective Optimization (SOO) and Multi-Objective Optimization (MOO) models are developed and evaluated. The optimization model includes two objectives: accuracy and total execution time. Numerical results show that optimal Hyperparameters (HP) settings allow for improving both the accuracy of the model and its computation time. The proposed method assists FL system designers in finding optimal parameters setup, allowing them to carry out model design and development efficiently depending on their goals.**

***Index Terms*—Federated Learning, Hyperparameters Optimization, Speech Emotion Recognition**

I. Introduction

Traditional Machine Learning (ML) approaches require training data gathered at a central location where the learning algorithm runs. In real-world scenarios, however, training data is often subject to privacy or regulatory constraints restricting how data can be shared, used, and transmitted. Federated learning (FL), first proposed in [1], has recently become a popular approach to address privacy concerns by allowing collaborative training of ML models among multiple parties where each party can keep its data private.

In the context of deploying the SER model on-device and emphasizing user privacy, we investigate the feasibility of using FL to train speech models directly on users' devices. FL is a decentralized training approach that eliminates the need to transmit raw user data to central servers. Instead, user data is stored in an on-device training cache, allowing local training iterations. FL optimization occurs through synchronous training rounds, where a group of clients (devices) contributes updates to a central model. FL has already demonstrated successful deployment in various large-scale production systems for tasks such as emoji prediction [2], next-word prediction [3], and query suggestion [4].

Since FL involves training models across multiple decentralized devices, careful selection of hyperparameters becomes essential for effective coordination and optimization. The choice of hyperparameters, such as learning rate, batch size, and regularization strength, directly impacts the convergence speed, communication efficiency, and model accuracy in FL settings.

While HPO is extensively researched in the centralized ML context, it introduces distinct challenges when applied to the FL environment. Firstly, most HPO techniques designed for centralized training harness the entire dataset, which is inaccessible in FL due to clients only having their local data. Secondly, these techniques entail training multiple models across various HP configurations. This approach can be exorbitantly costly in terms of both communication and training time within FL scenarios. The selection of an appropriate HPO algorithm is often contingent on the specific use-case, with notable algorithms including grid and random search, Bayesian optimization, evolutionary algorithms, reinforcement learning, gradient-based optimization, and multi-fidelity optimization. For a detailed review of HPO strategies, their classifications, and applications, especially in the context of FL, refer to [4]–[9].

This paper seeks to automate the selection of optimal hyperparameters for the FL model, targeting enhancements in both accuracy and total computation time. Our approach involves tackling both Single-Objective Optimization (SOO) and Multi-Objective Optimization (MOO) problems, considering constraints on predefined hyperparameters. These constraints encompass client-specific training parameters like learning rate, local epochs, batch size, and global FL system design parameters, including the number of global epochs and the client division factor.

We propose a novel method that simultaneously fine-tunes hyperparameters across two categories: local training parameters and global FL system design parameters. The presented outcomes stem from a constrained model concerning its feasibility space and the deterministic optimization algorithm employed. The primary objective was to authenticate and evaluate this fresh approach, utilizing HPO to amplify the performance of an SER FL model. It's worth noting that the scope of this study does not encompass the selection of the

most proficient optimization algorithm, which is earmarked for future exploration.

The remainder of this paper unfolds as follows: Section II delves into foundational concepts surrounding SER, FL, and HPO. In Section III, we delineate the system design. Experimental results from our proposed method are presented in Section IV. We wrap up with conclusions and potential avenues for future work in Section V.

II. Background

This section aims to provide a general overview and background of SER, FL, and HPO.

A. Speech Emotion Recognition

Speech Emotion Recognition (SER) was proposed for the first time in [10] and has attracted much attention since then. Several factors contribute to definitions of emotion, including personal experiences, physiologic reactions, and behavioural responses. According to these definitions, two types of SER models have been developed: discrete emotional models and dimensional emotional models. Discrete emotions are categorized into six categories: sadness, happiness, fear, anger, disgust, and surprise. A majority of existing SER systems are based on discrete emotion categories.

Generally, SER consists of three components: speech signal acquisition, feature extraction, and classifiers to identify emotions. Building SER classifiers require significant amounts of data, including sensitive personal information such as speech signals. However, centralized storage of this data presents privacy risks. To mitigate these risks, FL is a promising solution that allows models to be trained collaboratively on decentralized devices without the need to transfer raw data [11].

B. Federated Learning

Federated Learning (FL) is a distributed ML technique that trains the model on a large dataset distributed among multiple devices or clients rather than on a centralized server [1]. This allows training the models on large datasets that would not fit on a single device and also helps protect the data's privacy by keeping it on the device. According to Fig. 1, the FL system for SER application is composed of clients $\{C_1, ..., C_n\}$ like mobile phones, laptops, and TV, which perform SER models on devices without sharing speech data $\{D_1, ..., D_n\}$ with the central server. As shown in Fig. 1, three steps are generally involved in FL training.

- **Step 1:** Central server shares the initialized global SER model and assigns it to selected clients by specifying training parameters.
- **Step 2:** Clients update their local SER model parameters using their own speech data and the global model. When the client's loss function is minimized, W_i is sent to the server for updates.
- **Step 3:** Server aggregate the local models W_i and return the updated global model W_G to clients.

Steps 2-3 are repeated until a satisfactory level of accuracy has been achieved, or the global loss function has converged.

C. Hyperparameters Optimization

In general words, Hyperparameter Optimization (HPO) for ML is the process of finding a set of hyperparameters to achieve minimum loss or maximum accuracy of the objective network. The rigorous definition of the HPO in the general case could be found at [5]. The problem of HPO for ML has a long history, and it was also established early that different hyperparameter configurations tend to work best for different datasets [5], [12]. Besides that, HPO can be used to adapt general-purpose pipelines to specific application domains, and it is also widely acknowledged that tuned hyperparameters improve over the default setting provided by common ML libraries.

As Fig. 1 shows, FL models have complex structures based on several parameters and computationally demanding algorithms. Training of underlying ML models, which is done locally using the dataset of each client, is quite a challenging and computationally heavy task. Besides this local step, sending model parameters to the central server and receiving the updated parameters in each round of training, in addition to the aggregation process, are contributing to complexity. FL models' performance and complexity are partly affected by the choice of a certain set of parameters.

HPO is used in FL as an instrument for enhancing its performance. While Federated Optimization is one known case, a literature search suggests some recent studies where HPO was used to optimize hyperparameters of FL systems of different kinds: neural architectural and non-architectural hyperparameters. Regarding optimization of NN architecture, [9] gives a rather comprehensive survey on parameters and neural architecture search (NAS) approaches based on reinforcement learning, evolutionary algorithms, and gradient-based methods. Concerning non-architectural parameters, since the learning rate has a great impact on the accuracy of the NN algorithms, it is mostly used as an optimization variable in current research in this field, [13]–[15], etc. Besides that, the number of iterations in the Stochastic Gradient Descent (SGD) method [15], weight decay, momentum, and patch drop-out, [13] are optimized. In paper [13], both types of hyperparameters are optimized simultaneously: non-architectural and those related to cell-based architecture, which is a significant contribution.

In a vein similar to the work of Seng et al. [13], this paper proposes a method that optimizes two categories of hyperparameters, illustrated in Fig. 1. These are the training or non-architectural parameters (like learning rate, batch size, and the number of local epochs) and the global FL system design parameters (such as the number of global epochs and client division factor). It is essential to highlight that while there are parallels, discernible differences exist between the two studies regarding the hyperparameters considered. Unlike [13], our second set of parameters does not pertain to the neural architecture and even our roster of non-architectural parameters exhibits variation.

In this study, we singularly address the HPO problem, striving to pinpoint the ideal harmony between local training and

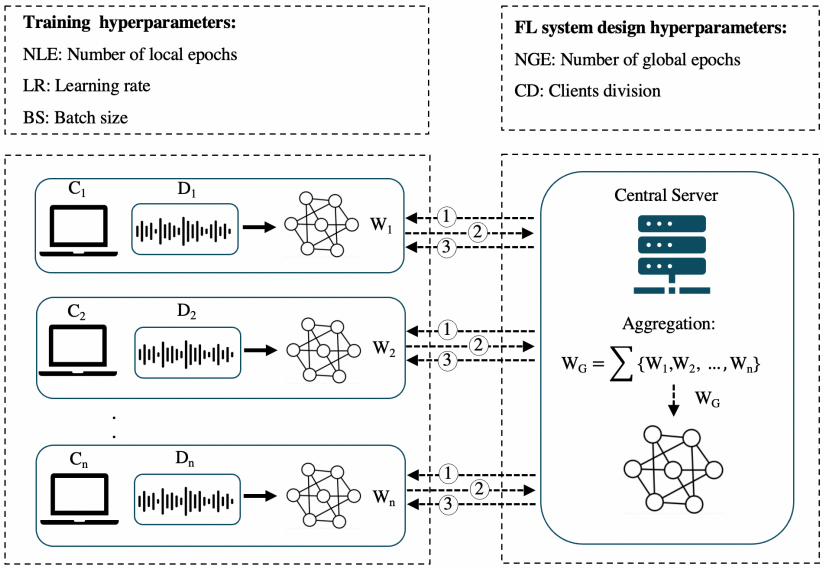

Fig. 1: A schematic diagram of FL in SER applications.

global FL design parameters. After this optimization phase, the hyperparameter values stabilize at their optimum. These hyperparameters, crafted for optimal accuracy and computational efficiency, persist within the FL system until changes in local client data or other influencing factors necessitate adjustments. While our approach hinges on deterministic single and multi-objective optimization, alternative studies have explored techniques, including reinforcement learning.

III. System design

This section delves into the optimization of hyperparameters within an FL system tailored for addressing the SER challenge.

A. Hyperparameters Optimization for SER Model in Federated Learning

In this work, we propose the method that optimizes hyperparameters of the FL model to achieve better accuracy and lower execution time:

NGE Number of global epochs refers to the number of rounds in which the global model is trained and aggregated with the local models of clients.

NLE Number of local epochs refers to the number of times a client device performs local training using its own local data before sending the updated model parameters to the central server for aggregation.

LR Learning rate refers to the parameter that controls how quickly the model parameters are updated during training in FL. It determines the step size that the model takes in the direction of the gradient during the optimization process. The central server sets the learning rate; it can be used to control the overall training speed and the stability of the model updates.

BS Batch size refers to the number of samples from the clients' local datasets that are used in each iteration of the model training in FL. Unlike traditional centralized ML where a fixed batch size is chosen for each iteration, FL has some unique considerations when selecting the batch size.

CD Clients division refers to the proportion of clients that are selected for model training in each round of the learning process. The performance and efficiency of FL systems are affected by the number of client rates.

Acc Accuracy is commonly used as a performance metric that measures the overall performance of the federated model in terms of correctly classified or predicted data points.

CT Total computation time required for training the FL model. Typically, most of the computation time in FL is devoted to training the local models on the clients. This process involves executing the forward and backward passes of the model on the local data. A smaller portion of the computation time is allocated to parameter aggregation and model validation during each round of training on the server side.

As highlighted previously, hyperparameters, also referred to as optimization variables, come in two varieties, as depicted in Fig. 1. The learning rate, batch size, and the number of local epochs are training parameters employed individually on each device. Under this framework, all devices utilize identical neural network models, encompassing both the network architecture and training parameters. Consequently, optimization outcomes prescribe hyperparameters universally applicable to models on local clients. The number of global epochs and the client division factor act as global parameters, shaping the FL system design. These parameters oversee the communication between the central server and the local devices.

B. Formulation of Hyperparameters Optimization Problems

In this context, we introduce three optimization problems. The first two are single-objective problems, focusing on optimizing accuracy and computation time respectively, while the third addresses a multi-objective challenge, aiming to optimize both the aforementioned objectives concurrently.

Optimization variables, or hyperparameters, are introduced above. The limits on the variables constrain this problem according to (2)- (6) in the formulation below. Formally, the first problem is formulated as:

$$max \quad Acc(NGE, NLE, LR, BS, CD) \tag{1}$$

$$s.t.$$

$$NGE = \{50, 100, 150, 200\} \tag{2}$$

$$NLE = \{1, 3, 5\} \tag{3}$$

$$LR = \{0.1, 0.01, 0.001\} \tag{4}$$

$$BS = \{8, 16, 32, 64\} \tag{5}$$

$$CD = \{0.10, 0.50, 0.75, 1.0\} \tag{6}$$

As the formulation above shows, the feasibility region (2)-(6) is a discrete 5-dimensional grid. Thus, a simple grid search is used to solve the problem. The maximum of the objective function (1) is found across the 5-dimensional grid consisting of $4 \times 3 \times 3 \times 4 \times 4$ points, using the brute force search.

The secondary problem is the minimization of computation time. The formulation of this problem is similar to the problem (1) -(6). The difference is the objective function:

$$min \quad TotalComputationTime(NGE, NLE, LR, BS, CD) \tag{7}$$

Finally, we present a bi-objective optimization that addresses the two preceding challenges simultaneously. Due to the inherent conflict between these objectives, maximizing accuracy can potentially lead to prolonged computation time; a singular point does not represent the solution. Instead, it is captured by the Pareto frontier. The subsequent section provides a detailed formulation of this problem, which is approached within the framework of the previously established constraints.

$$\{max \quad Acc, \quad min \quad TotalComputationTime\} \tag{8}$$

IV. Evaluation

This section will explain the industrial use case and simulation setting. Next, the experimental result of tuning hyperparameters to optimize the accuracy and computation time will be presented as single-objective and multiobjective problems.

A. Use case description and simulation setting

DAIS [1] (Distributed Artificial Intelligent Systems) [16] is a pan-European project that aims to improve interoperability and trustworthiness by leveraging the Internet of Things (IoT) and artificial intelligence (AI) to develop a distributed edge intelligence system. The project encompasses a range of industry-driven use cases in domains such as Digital Life, Digital Industry, and Smart Mobility.

In the Digital Life domain, SER is a critical use case. SER is employed in home entertainment recommendation systems, such as smart TVs, where digital content, such as movies, is recommended based on user emotion. FL setup meets these requirements to ensure privacy, efficiency, and a distributed system of SER. Hyperparameters greatly influence the performance of SER models in FL setups. The choice of hyperparameters significantly impacts the accuracy of the models, and the optimal hyperparameters may vary depending on the data characteristics and task requirements. It is imperative to identify the appropriate hyperparameters to optimize performance in SER using an FL setup.

We evaluated our method on one of the most widely used SER datasets, namely CREMA-D [17]. The CREMA-D dataset contains $7,442$ original speech recordings from 91 actors. These recordings were made by 48 males and 43 females with different accents. To train the SER model, we chose the four most commonly occurring emotion labels (neutral, sad, happy, and angry) based on the possible emotions expressed in the sentences. Each speaker serves as a unique client for the FL training on the CREMA-D dataset since there are 91 distinct speakers in the dataset.

We use the multilayer perceptron (MLP) as the SER model architecture. The model has two dense layers with layer sizes of [256, 128], ReLU as an activation function, and a 0.2 dropout rate. We employed 80% of the clients (73 clients) for local training as edge nodes and reserved the remaining 20% (18 clients) for validation. We considered the average accuracy of these 18 clients for the optimization problem (1). For the computation time optimization problem (7), we reported the training time of the FL system for the 73 clients. As all the clients were created on one laptop with equal computation power, our simulated FL system did not have device heterogeneity. At the same time, we considered the Non-IID dataset, and our research worked well considering our challenging data distribution. The experimental environment is Windows 10 Enterprise, processor Intel(R) Core(TM) i7-9850H CPU @ 2.60GHz 2.59 GHz and RAM 32,0 GB.

B. Experimental Result

As mentioned, the optimization problem (1) - (6) is solved using the direct search across the 5-dimensional grid by brute force method. This means the objective function is evaluated on all grid points constituting the feasibility set. This approach confirms that global optimality is achieved. The optimal solution is compared with the standard setup of hyperparameters used in the FL model; see Table I. Results obtained show improvement both in terms of accuracy and computation time. The solution to the problem of maximization of accuracy gives an optimal accuracy of 0.771, while it was 0.751 in the default setup. At the same time, this solution gives a sufficient decrease in computation time, too: it takes around 97 sec against 351 sec for the standard setup. The solution to the second problem, computation time minimization, shows a

[1]DAIS Project Website: https://dAIS-project.eu

TABLE I: Solutions of single objective optimization problems.

Results	Acc	NGE	NLE	LR	BS	CD	Time
Default Setup	0.751	200	3	0.001	32	1.0	352
Optimal Accuracy	0.771	50	5	0.01	16	0.50	97
Optimal Comp. Time	0.729	50	1	0.001	16	0.10	25

TABLE II: Pareto Optimal points (POP).

POP	Acc	NGE	NLE	LR	BS	CD	Time
Point 1	0.729	50	1	0.001	16	0.10	25.55
Point 2	0.737	50	1	0.001	32	0.10	25.66
Point 3	0.743	50	5	0.001	32	0.10	33.52
Point 4	0.760	50	1	0.01	8	0.50	59.41
Point 5	0.771	50	5	0.01	16	0.50	96.98

significant decrease in time, which is around 25 sec, and it is in order of magnitude better than the one for the standard setup. Table I below summarizes numerical results obtained by solving problem (1) - (6) and (7), (2) - (6).

MOO problem (8), (2) - (6) is solved by the posterior method. This means that the approximate Pareto frontier is calculated first and then the decision maker chooses one or several alternative Pareto optimal solutions. These solutions will be used further to build an optimal FL model. Two single objective problems were used to sample dominated and non-dominated points to create the Pareto frontier. Non-dominated points are those which have at least one multiobjective function value better than other points, which are called as dominant. In fact, all dominated points were filtered out from the set of grid points, and the remaining non-dominated points constitute optimal Pareto Frontier, see Fig.4. In other words, the convex hull was created from the set of all grid points (2) - (6). It consists of 5 points, where the first and last points are optimal solutions to the single objective problems. Information about the points constituting the Pareto frontier is gathered in Table II.

Another important aspect is that solutions to optimization problems help identify significant variables (model parameters) affecting objective functions. This could be done by analyzing the dependencies of the objective function on certain model parameters. As an example, see Fig. 2, where the accuracy is plotted as a function of LR, the learning rate of the model. Three accuracy curves correspond to LRs equal to $0.1, 0.01$, and 0.001. This plot shows that $LR = 0.1$ produces the lowest accuracy for all experiments, while the other two LR values overperform the case with $LR = 0.1$. This means that fixing this parameter, for example, to 0.01, increases the accuracy of the FL model irrespective of the values of other parameters. The rest of the parameters do not affect accuracy a lot.

On the contrary, total computation time is affected by multiple parameters, see Fig. 3. The number of global epochs and the client division factor contribute most to the computation timing. Fig. 3 confirms the trend that a lower number of global epochs and client division factor gives shorter computation time. This is easily explained by the fact that the complexity of the FL model grows as these parameters increase. Except for these parameters, the number of local epochs should be mentioned, too.

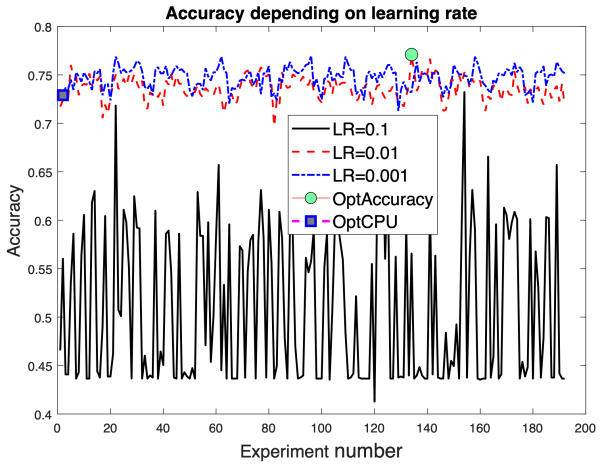

Fig. 2: Accuracy in each round of training

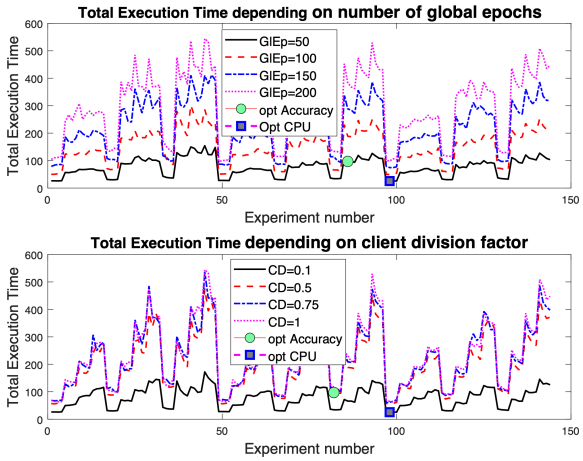

Fig. 3: Total Execution time in each round of training

Finally, analysis of the Pareto optimal solutions shows that the optimal number of global epochs is equal to 50 in all cases. The optimal values of the rest of the parameters vary. This means that accuracy and computation time are not conflicting concerning the number of global epochs. Thus, the size of the MOO problem could be reduced by fixing this parameter. This information should be used when the MOO is extended to include more optimization variables. These results provide valuable information that should be used while building optimal FL models providing high accuracy and efficient CPU time.

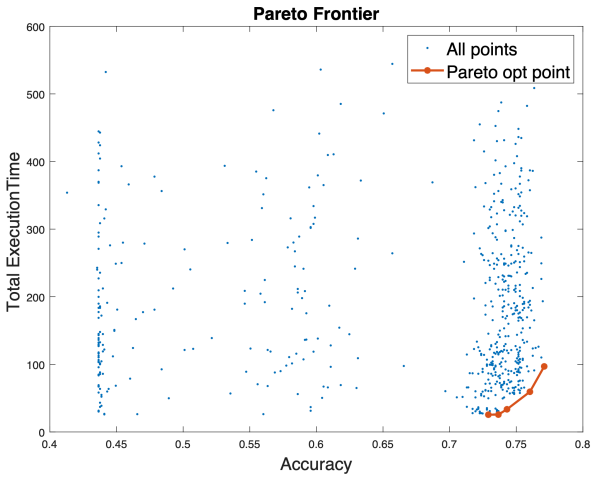

Fig. 4: Pareto Frontier.

V. Conclusion and Future Work

Conclusions This paper presents a novel approach to create a flexible framework for solving FL model for SER application. This is done by introducing an extra step to optimize the training, further FL model development and parameter design in an automatic way. To this end, HPO is used to maximize the performance of the FL model in terms of model accuracy and computational time.

The strong feature of this approach is its flexibility, since it could be applied for any type of FL model. Optimization is created and solved based upon FL model and delivers optimal hyperparameters set. This fixed HP setup is used further in training and model development as long as other FL model parameters stay unchanged. Thus, optimization problem, which is quite computationally expensive, is solved once for each predefined model setup. If model designer alters the choice of hyperparameters, their ranges or any other model setup, the new optimization problem should be solved.

The optimization problem itself could be formulated and solved as a discrete SOO and MOO, including different hyperparameters. Implementation of MOO is an important feature since it finds parameters optimizing several objectives simultaneously, which boosts the performance of FL models.

Presented optimization model includes two objectives: accuracy and computation time with 5 discrete variables. Optimization problems were solved by the simplest grid search, where all feasible points belonging to the 5 - dimensional grid were inspected. While the SOO solution finds a single optimal point for the accuracy or computation time and correspondent values for hyperparameters, the MOO solution is presented as Pareto Frontier, including all non-dominated points or alternative solutions. This Pareto Frontier is used by the decision maker, who chooses a proper alternative solution to set up the FL model. Numerical results show that optimal HP settings allow for significantly improved both accuracy of the model and its computation time.

Future work This approach has a great potential and will be developed further. Firstly, the optimization model should be extended to consider the FL system's stochastic nature due to the Stochastic Gradient Descent method used by local clients. This could be done in different ways, such as using other algorithms as e.g. reinforcement learning or creating more complex models, where optimization is done as a part of training. The choice of an appropriate optimization algorithm was not the focus of this paper and is a subject of separate research.

Secondly, additional objectives such as differential privacy and communication should be included concerning the model complexity. The solution to such a problem will regulate the trade off between model accuracy, privacy, and communication costs. This gives extra flexibility to the model when researchers can work with the different HP setups depending on their research questions. That is, if, for example, the model is to be trained for better accuracy, the parameters setup for optimal accuracy is relevant. When the FL model itself is to be revised by including new parameters/features, the setup with the computation time is highly relevant. Thus, Pareto Frontier could be used to find the most efficient parameters setting depending on the research question.

VI. Acknowledgements

This work was supported by EU ECSEL project DAIS which has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No.101007273. The work reflects only the authors' views; the European Commission is not responsible for any use that may be made of the information it contains.

References

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

[2] S. Ramaswamy, R. Mathews, K. Rao, and F. Beaufays, "Federated learning for emoji prediction in a mobile keyboard," *arXiv preprint arXiv:1906.04329*, 2019.

[3] A. Hard, K. Rao, R. Mathews, S. Ramaswamy, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage, "Federated learning for mobile keyboard prediction," *arXiv preprint arXiv:1811.03604*, 2018.

[4] T. Yang, G. Andrew, H. Eichner, H. Sun, W. Li, N. Kong, D. Ramage, and F. Beaufays, "Applied federated learning: Improving google keyboard query suggestions," *arXiv preprint arXiv:1812.02903*, 2018.

[5] M. Feurer and F. Hutter, *Hyperparameter Optimization*. Cham: Springer International Publishing, 2019, pp. 3–33. [Online]. Available: https://doi.org/10.1007/978-3-030-05318-5_1

[6] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, "Hyperband: A novel bandit-based approach to hyperparameter optimization," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 6765–6816, 2017.

[7] D. Maclaurin, D. Duvenaud, and R. Adams, "Gradient-based hyperparameter optimization through reversible learning," in *International conference on machine learning*. PMLR, 2015, pp. 2113–2122.

[8] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization." *Journal of machine learning research*, vol. 13, no. 2, 2012.

[9] H. Zhu and Y. Zhang, Haoyu andJin, "From federated learning to federated neural architecture search: a survey." *Complex Intelligent Systems*, vol. 7, 2021.

[10] S. Poria, E. Cambria, R. Bajpai, and A. Hussain, "A review of affective computing: From unimodal analysis to multimodal fusion," *Information Fusion*, vol. 37, pp. 98–125, 2017.

[11] S. Latif, S. Khalifa, R. Rana, and R. Jurdak, "Federated learning for speech emotion recognition applications," in *2020 19th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*. IEEE, 2020, pp. 341–342.
[12] R. Kohavi and G. H. John, "Automatic parameter selection by minimizing estimated error," in *Machine Learning Proceedings 1995*. Elsevier, 1995, pp. 304–312.
[13] J. Seng, P. Prasad, D. S. Dhami, and K. Kersting, "Hanf: Hyperparameter and neural architecture search in federated learning." *arXiv preprint arXiv:2206.12342*, 2022.
[14] A. Koskela and A. Honkela, "Learning rate adaptation for federated and differentially private learning." *arXiv preprint arXiv:1809.03832*, 2019.
[15] H. Mostafa, "Robust federated learning through representation matching and adaptive hyperparameters." *arXiv preprint arXiv:1912.13075*, 2019.
[16] A. Balador, S. Sinaei, M. Pettersson, and I. Kaya, "Dais project - distributed artificial intelligence systems: Objectives and challenges," in *26th Ada-Europe International Conference on Reliable Software Technologies (AEiC'22)*, 2022.
[17] H. Cao, D. G. Cooper, M. K. Keutmann, R. C. Gur, A. Nenkova, and R. Verma, "Crema-d: Crowd-sourced emotional multimodal actors dataset," *IEEE transactions on affective computing*, vol. 5, no. 4, pp. 377–390, 2014.