NEvoFed: A Decentralized Approach to Federated Neuroevolution of Heterogeneous Neural Networks

Leonardo Lucio Custode
leonardo.custode@unitn.it
University of Trento
Trento, Italy

Ivanoe De Falco
ivanoe.defalco@icar.cnr.it
ICAR-CNR
Naples, Italy

Antonio Della Cioppa*
adellacioppa@unisa.it
NCLab, DIEM, University of Salerno
Salerno, Italy

Giovanni Iacca
giovanni.iacca@unitn.it
University of Trento
Trento, Italy

Umberto Scafuri
umberto.scafuri@icar.cnr.it
ICAR-CNR
Naples, Italy

ABSTRACT

In the past few years, Federated Learning (FL) has emerged as an effective approach for training neural networks (NNs) over a computing network while preserving data privacy. Most of the existing FL approaches require the user to define *a priori* the same structure for all the NNs running on the clients, along with an explicit aggregation procedure. This can be a limiting factor in cases where pre-defining such algorithmic details is difficult. To overcome these issues, we propose a novel approach to FL, which leverages Neuroevolution running on the clients. This implies that the NN structures may be different across clients, hence providing better adaptation to the local data. Furthermore, in our approach, the aggregation is implicitly accomplished on the client side by exploiting the information about the models used on the other clients, thus allowing the emergence of optimal NN architectures without needing an explicit aggregation. We test our approach on three datasets, showing that very compact NNs can be obtained without significant drops in performance compared to canonical FL. Moreover, we show that such compact structures allow for a step towards explainability, which is highly desirable in domains such as digital health, from which the tested datasets come.

CCS CONCEPTS

• **Computing methodologies** → **Distributed artificial intelligence**; **Neural networks**; **Genetic algorithms**; • **Theory of computation** → **Evolutionary algorithms**.

KEYWORDS

Federated Learning, Neuroevolution, Supervised Learning

ACM Reference Format:
Leonardo Lucio Custode, Ivanoe De Falco, Antonio Della Cioppa, Giovanni Iacca, and Umberto Scafuri. 2024. NEvoFed: A Decentralized Approach to Federated Neuroevolution of Heterogeneous Neural Networks. In *Genetic and Evolutionary Computation Conference (GECCO '24), July 14–18, 2024, Melbourne, VIC, Australia.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3638529.3654029

*Also with ICAR-CNR.

GECCO '24, July 14–18, 2024, Melbourne, VIC, Australia

ACM ISBN 979-8-4007-0494-9/24/07.
https://doi.org/10.1145/3638529.3654029

1 INTRODUCTION

Recently, data privacy has become an extremely important concern in the field of Machine Learning (ML) [12]. This led to the emergence of *Federated Learning* [15, 16] (FL) as an alternative to traditional (stand-alone) ML to train models over a computing network while maintaining data privacy.

FL is an ML methodology in which a model, typically in the form of a Neural Network (NN), is trained collaboratively on a set of computing nodes (the clients) without the need for exchanging data. The canonical form of FL takes place through the use of a server on which an initial NN is defined and then sent to the clients; on each such client, the NN performs a typical supervised learning phase, running an independent training session on a client-local dataset (with the different local datasets supposed to be heterogeneous). After this step, the resulting trained NNs (but not the data) are sent back to the server. The latter receives a set of NNs with the same structure but with different connection weights, and performs an aggregation phase; in this phase, a new NN (called the "global" model) is obtained that shows the same structure as the incoming NNs, whereas the weight of each connection is computed aggregating (e.g., averaging) the corresponding values from each incoming NN. The resulting global model is then sent again to the clients, and the process iterates until a given stop criterion is met.

This approach allows for effectively dealing with the issue of data privacy, which is particularly significant in all the application fields where the data should remain private and not disclosed to any other participant such as, e.g., in digital health, loan granting, legal procedures, etc. However, while FL proved to work very well in practical applications [39], it still yields major challenges, namely:

- *Choice of the NN architecture*: in most FL approaches, the architecture is fixed by the system designer, which prevents the clients from learning more effective NNs for the task at hand, often resulting in overly complex models.
- *Distribution of the inputs*: often, the data of each client violates the IID (i.e., Independent and Identically Distributed) principle, which is fundamental for effective ML. Thus, canonical FL methodologies could learn suboptimal models.

In this paper, we make an attempt at mitigating these two problems by leveraging principles from *Neuroevolution* [32] (NE), a

branch of Evolutionary Computation that aims to automatically find, at the same time, the most suitable structure for an NN along with the optimal values of its parameters (i.e., the weights). Starting from the seminal papers published in the 1990s [23, 26, 38, 40], many NE algorithms have been proposed which, roughly speaking, can be classified into either direct-encoding or indirect-encoding ones, depending on whether or not the genotype directly maps to the phenotype. So far, only a few works have attempted to link NE and FL. Yet, to the best of our knowledge, none of the existing works combining NE and FL specifically aim to evolve *heterogeneous* NNs in a *decentralized* way, which is the distinctive feature of work.

Our proposed method, that we call NEvoFed, performs FL in such a way that, at each time, the structures of the NNs may differ from client to client. To obtain this, we run an independent NE algorithm on each client, where a set of NNs is randomly generated, and then undergoes a learning phase on the local dataset. Then, the evolutionary process allows to obtain new populations of NNs with different structures and connection weights. At a given number of generations, the clients exchange the local best NNs: the incoming NNs replace bad-performing local ones, and the evolution continues. In this way, the NNs that were good on the clients they came from are tested on other clients, hence on different local data. The rationale behind this mechanism is that, as the number of generations (and thus NN exchanges) increases, the NNs that are able to behave well on more local datasets will emerge.

Two main differences between the approach we propose here and canonical FL can be seen. Firstly, the structure of the NNs may be different both *within each client* and *from client to client*; this helps to find, in an evolutionary way, a good structure, whereas as said in canonical FL such structure (in terms of the number of layers and neurons per layer) should be decided before the execution of the algorithm. Secondly, unlike canonical FL, in NEvoFed there is no need for an explicit aggregation phase to be performed on the server; rather, the NN exchange mechanism we introduce corresponds to an implicit aggregation phase that is driven by evolution.

We experimentally compare our proposed approach against the canonical FL in terms of classification results achieved over three biomedical datasets that are well-suitable for FL. Our results show that, overall, NEvoFed outperforms in most cases the canonical FL at both client and global levels.

The rest of the paper is structured as follows. Section 2 reports the related work in the areas of NE and FL and highlights the originality of our proposal. Section 3 describes the general structure of our proposed algorithm. Section 4 accounts for the experimental setup and the datasets on which our experimentation takes place. Section 5 discusses the results, and finally Section 6 gives our conclusions and suggests future work.

2 BACKGROUND AND RELATED WORK

During the last years, a good number of papers have investigated the intersection between NE and FL; a comprehensive survey on this topic is presented in [45], where the emphasis is given on the evolution from canonical FL to federated neural architecture search (FNAS). In this survey, firstly, NE is considered on its own. In the past few decades of research, the authors identify three main groups of search strategies for finding optimal NN architectures, namely those based on reinforcement learning [11, 42, 46], those relying on Evolutionary Algorithm (EAs) [8, 9, 14, 20, 33], and those making use of gradient-based methods [10, 19, 37]. Further methods, less frequently used, include random search [5, 17], Bayesian optimization [29, 34], and multinomial distribution learning [41].

When NE has to be applied to FL, i.e., when evolutionary FNAS is considered, things get more complicated, given the nature of FL. This results in only a few papers being available in the literature despite an increasing interest in this direction. The approaches presented so far differ in several aspects, summarized below.

① The first difference lies in how the EA is used, depending on whether it is run only on the server and the clients execute the training on local data, or an EA instance runs on every client.

② A second differentiation is represented by the object of this optimization being performed through EAs: it could be the set of the learning parameters, the NN structure (in terms of number of layers, number of nodes per layer, etc.), the values of the connection weights (which can be optimized by using an EA instead of back-propagation), or more than one of these at the same time.

③ A third feature concerns the aggregation phase: this could occur over either models or fitness values.

④ A fourth characterization of these approaches can be made on the information exchange taking place between the server and the clients, as well as possible communication among clients: either all the models could travel, just a subset, or none.

Moreover, concerning communication, the approaches introduced in those papers can be divided into *offline* and *online*.

In *offline* approaches, e.g., [7, 18, 28, 35, 43], each NN in the population on the server must be trained on (a subset of) the clients, and, at each generation, each client has to train all the NNs in the current population. This implies a large demand for both computation and communication resources, yielding high execution times, especially on mobile devices. This prevents these methods from being used for real-time applications.

In *online* approaches, e.g., [25, 44], instead, each client only has to train sub-networks rather than whole NNs, so the number of parameters to be sent from the server to the client is much lower. Therefore, online approaches are much less resource-consuming and communication-intensive, which means that they can be used online for real-world applications.

The description of the above papers reveals that, in all the NE approaches to FL described in the recent literature, an EA to evolve NNs runs on the server while, on each client, one or more solutions generated on the server are received, evaluated on the local data, and slightly modified, which implies an explicit aggregation phase on the server; moreover, for each NN, an FL-based fitness can be computed on the server by averaging the local fitness values.

In our proposal, instead, as it will be described in detail in Section 3, on each client there is an independent EA running a local population of NNs on local data, and only the best individuals on each client are sent (in this, it can be seen as an instance of *island-based models* [13]). Moreover, in our approach there is no fitness averaging during the evolution; rather, the fitness of each individual is computed locally, and only the migrating individuals must have their fitness re-computed on the local data of the new clients they arrive at. These are significant differences that allow us to consider our approach as original.

It should be remarked here that, although strictly speaking our approach is offline, it nevertheless allows for fast evaluation of the candidate NNs because, in general, NE tends to generate structures that are as small as possible; this means that, in principle, our approach can be as resource-efficient as the online approaches.

Table 1 summarizes the main features of the approaches to FL based on NE proposed so far in the literature and compares them against those of our proposed algorithm. For each algorithm, we indicate: whether it optimizes through an EA the learning parameters, the NN structure, and the NN weights; if an EA instance runs on the clients; if the aggregation phase is performed on the local fitness values; if all the models travel between the server and the clients; and, if the approach is online.

3 PROPOSED METHOD

Before discussing our method, we briefly sketch out the canonical FL approach as introduced by Konečny et al. in [15, 16]. According to it, the aim is to identify a general model $\mathcal{M}_g$ by aggregating a set of m local models $\boldsymbol{\mathcal{M}} = \{\mathcal{M}_1, \ldots, \mathcal{M}_m\}$ trained on a number of different clients $\boldsymbol{\mathcal{C}} = \{\mathcal{C}_1, \ldots, \mathcal{C}_m\}$ on their local data $\boldsymbol{\mathcal{D}} = \{\mathcal{D}_1, \ldots, \mathcal{D}_m\}$, whose communication can be arranged either centrally by a server $\mathcal{S}$ or in a peer-to-peer manner. In the following, we will refer to the centralized FL (i.e., based on a server) both for the sake of conciseness and in that little changes apply to the peer-to-peer one.

Centralized FL (Algorithm 1) consists of the following steps:

(1) **Model initialization**: the server initializes a model $\mathcal{M}_g$.
(2) **Client selection**: the server selects at random a subset of clients, and distributes $\mathcal{M}_g$ to them.
(3) **Local model training**: each selected client $\mathcal{C}_i$ performs a learning phase on its own local data, thus obtaining its own updated version of the model $\mathcal{M}_i$. Then, $\mathcal{M}_i$ is sent back to the server.
(4) **Aggregation**: once the server has received all the local updated models $\mathcal{M}_i$, it aggregates them (typically, this is accomplished by averaging the values of the models' parameters, e.g., in the case of an NN, the weights of the neurons) thus obtaining a new global model $\mathcal{M}_g$. Then, such a model is sent to all the selected clients, and the process continues from step (2) until a stopping criterion is met.

By doing so, the FL goal becomes to train separately a general model $\mathcal{M}_g$ on all the local datasets of the different clients, in order to optimize the following performance function:

$$\psi(\mathcal{M}_g) = \frac{1}{m} \cdot \sum_{i=1}^{m} \phi(\mathcal{M}_i) \tag{1}$$

where $\phi(\mathcal{M}_i)$ is the local performance function value of the model $\mathcal{M}_i$ on the i-th client. While it is possible to evaluate $\psi(\mathcal{M}_g)$ each time at the end of step (3), actually it is computed by the server only at the end of the whole learning process. In fact, before stopping, the clients send the last model update to the server along with their local performance; then, the server aggregates all the updates into the final global model, computes $\psi(\mathcal{M}_g)$, and sends the final global model back to the clients.

A first criticism of this canonical FL model is that all clients must share the same NN architecture. This assumes that information about the architecture of the NN to be used somehow must be shared among all clients prior to the training phase. Several different

Algorithm 1 Federated Learning (canonical version)

```
1:  procedure FederatedLearning
2:      Initialize global model M_g
3:      for each round t = 1, 2, ..., T do
4:          Select a subset C_t ⊆ C of n ≤ m clients
5:          for each client C_i ∈ C_t in parallel do
6:              M_i ← ClientUpdate(i, M_g)
7:          end for
8:          M_g ← AggregateModels(M_1, ..., M_n)
9:      end for
10: end procedure
11:
12: procedure ClientUpdate(k, M_g)
13:     Initialize local model M_k with M_g
14:     for each local epoch i = 1, 2, ..., E do
15:         Update M_k using local data of C_k
16:     end for
17:     return M_k
18: end procedure
19:
20: procedure AggregateModels(M_1, ..., M_n)
21:     M_new ← Aggregate M_1, ..., M_n
22:     return M_new
23: end procedure
```

methodologies have been proposed in the literature to overcome such a limitation:

- *Knowledge Distillation* [24], where a teacher-student model is used, i.e., client models (teachers) are used to train a centralized model (student). The central model learns to mimic the outputs of the distributed models.
- *Model Personalization* [1], where rather than trying to aggregate different architectures into a single global model, the aim is to personalize the global model for each client. Techniques like adding small, client-specific layers to a shared base model are typically used, allowing for diverse model architectures while maintaining a common core.
- *Layer-wise Aggregation* [21], where the focus is on aggregating models at the layer level rather than as a whole. This approach may involve standardizing specific layers among all models while allowing diversity in other layers.
- *Sub-model Aggregation* [2], where models are decomposed into sub-models. Such sub-modules are then trained independently and aggregated separately, thus allowing different clients to contribute to different parts of a model.
- *Hybrid Models and Meta-Learning* [31], where a meta-model is trained using different architectures to learn how to aggregate models.

However, each of these approaches comes with its own set of challenges, such as increased computational complexity and potential privacy concerns, and requires a careful design to ensure effective learning across different models.

As mentioned earlier, in our proposed NEvoFed, instead, each client evolves a population of heterogeneous NNs using an NE algorithm, in our case NEAT [32, 33] (although, in principle, any

Table 1: The main features of the approaches proposed in the literature and their comparison against our proposal. Seven features of interest are considered, namely: ① Learning parameter optimization: does the approach use an EA to optimize the learning parameters of the NN? ② NN structure optimization: does the approach use an EA to optimize the structure of the NN? ③ NN weight optimization: does the approach use an EA to optimize the connection weights of the NN? ④ EA runs on clients: does an EA algorithm also run on any client? ⑤ Aggregation is done on local fitness: is the fitness computed as an aggregation over the local fitness values? ⑥ Not all the models travel: are not all the NN models sent at each generation? ⑦ Approach is online: is the approach online? (✓: yes; ✗: no; ✓(+S): the same EA is run on both server and clients; N/A: not available).

Paper	Year	Learning parameter optimization	NN structure optimization	NN weight optimization	EA runs on clients	Aggregation is done on local fitness	Not all the models travel	Approach is online
[35]	2019	✗	✗	✓	✗	✗	✗	✗
[43]	2019	✓	✓	✗	✗	✓	✗	✗
[44]	2021	✓	✓	✗	✗	✗	✗	✓
[25]	2022	✗	✗	✓	✗	✗	✗	✓
[18]	2022	✓	✓	✓	N/A	✗	✗	✗
[7]	2023	✓	✓	✗	✗	✓	✗	✗
[28]	2023	✗	✗	✓	✓(+S)	✓	✓	✗
NEvoFed		✗	✓	✓	✓	✓	✓	✗

other NE method could be used). The proposed approach is based on the following key assumptions:

- The architecture of the NN should not be *a priori* set, but it should naturally emerge from the learning process.
- The model complexity should be as low as possible.
- The aggregation step should not be performed by the server, but it should be done on the client side by exploiting the information about the exogenous models used on either all of the other clients or a subset of them.
- The aggregation should not be accomplished in an explicit way (e.g., by averaging the values of the models' parameters), but it should be related to the exogenous models' performance on the local data. In other words, no explicit aggregation should be devised, but rather it should take place in an implicit way during the NE process thanks to the recombination and mutation mechanisms, thus pushing the learning process to induce the emergence of an effective global model.

The steps that NEvoFed undergoes are the following:

(1) **Client selection**: the server selects randomly a subset of clients.
(2) **Local training phase**: each client participating in FL performs a learning phase on its own local data. At the end of the learning phase, the most performing model $\mathcal{M}_i$ is sent to the server.
(3) **Aggregation**: once the server has received the local models $\mathcal{M}_i$, it sends them to all the selected clients. Then, the received exogenous models replace the worst local models, if better, on local data, and the process continues from step (1) until a stopping criterion is met.

Finally, at the end of the whole training, the model with the best global performance, measured according to Eq. (1), is taken as the final global model. Algorithm 2 depicts the whole NEvoFed process.

It should be remarked here that, notwithstanding it produces an overhead of communication with respect to canonical FL, the aggregation step allows each client to receive and consequently exploit the information about the best models on all the clients involved in each FL round. In this way, we favor the emergence of a general global model while preserving a personalized local model. On the other hand, the communication overhead can be significantly reduced if the NNs sent have a small complexity.

4 EXPERIMENTAL SETUP

We now present the datasets used in the experimentation and details concerning the implementation and parameter setting.

4.1 Datasets

The choice of the datasets has been based on the consideration that they should be specific to FL and should have been gathered to be perfectly suited to FL. In fact, many papers in the literature exist that apply FL methodologies to datasets that were not explicitly created for FL; for example, wide use is made of datasets such as MNIST[1], CIFAR[2], or even much simpler ones, such as the Pima Indians diabetes dataset[3]. In these cases, a global dataset is typically split into subsets, each of which is placed on a client. However, as there could be relationships, even strong, between different subsets of data placed on different clients, this approach could lead to an easier learning task, because locally learning on a client's subset could yield unwillingly learning also on the data contained in the other subsets available to the other clients.

Therefore, in this paper, we consider the following three datasets from the scientific literature, which are instead specific to FL tasks:

- **Apnea-ECG**: The apnea-ECG database was originally presented in [27]. The original dataset contains 70 recordings, each obtained at night through single-lead ECG continuous monitoring of a subject. 35 of these recordings are annotated for each one-minute segment with respect to the presence of Obstructive Sleep Apnea (OSA) in that minute. 20 of these recordings refer to subjects definitely suffering from OSA. In [30], a new dataset was obtained from each of the 35 recordings, through Heart Rate Variability

[1] https://paperswithcode.com/dataset/mnist
[2] https://www.cs.toronto.edu/~kriz/cifar.html
[3] https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database

Algorithm 2 Federated Learning (proposed NEvoFed)

procedure NEvoFed
  $\mathcal{M} \leftarrow \emptyset$
  for each round $t = 1, 2, \ldots, T$ **do**
    Select a subset $\boldsymbol{\mathcal{C}}_t \subseteq \boldsymbol{\mathcal{C}}$ of $n \leq m$ clients
    for each client $\mathcal{C}_i \in \boldsymbol{\mathcal{C}}_t$ **in parallel do**
      if $\mathcal{C}_i$ not inizialized **then**
        ClientInizialize(i)
      end if
      $\mathcal{M}_i \leftarrow$ ClientUpdate(i)
    end for
    for each client $\mathcal{C}_i \in \boldsymbol{\mathcal{C}}_t$ **in parallel do**
      ClientAggregate($i, \mathcal{M}_1, \ldots, \mathcal{M}_n$)
    end for
  end for
  for each client $\mathcal{C}_i \in \boldsymbol{\mathcal{C}}_t$ **in parallel do**
    $\phi_i(\mathcal{M}_1), \ldots, \phi_i(\mathcal{M}_n) \leftarrow$ ClientEval($i, \mathcal{M}_1, \ldots, \mathcal{M}_n$)
  end for
  Compute global performance ψ_i (Eq. (1))
  $\mathcal{M}_g \leftarrow \max\{\psi_i\}_{i \in \{1,\ldots,n\}}$
  return $\mathcal{M}_g$ ▷ model with best global performance
end procedure

procedure ClientInitialize(k)
  Initialize a population of local models $\{\mathcal{M}_1 \ \ldots, \mathcal{M}_p\}$
end procedure

procedure ClientUpdate(k)
  for each local epoch $i = 1, 2, \ldots, E$ **do**
    Perform a NE generation using local data $\mathcal{D}_k$ of $\mathcal{C}_k$
  end for
  $\mathcal{M}_b \leftarrow \mathcal{M}_k^b$ ▷ best local model found so far
  return $\mathcal{M}_b$
end procedure

procedure ClientAggregate($k, \mathcal{M}_1, \ldots, \mathcal{M}_n$)
  Evaluate $\mathcal{M}_1, \ldots, \mathcal{M}_n$ on local data $\mathcal{D}_k$
  Sort the population of local models
  Replace the n worst models with $\{\mathcal{M}_1, \ldots, \mathcal{M}_n\}$ if better
end procedure

procedure ClientEval($k, \mathcal{M}_1, \ldots, \mathcal{M}_n$)
  Evaluate $\mathcal{M}_1, \ldots, \mathcal{M}_n$ on local data
  return $\phi_k(\mathcal{M}_1), \ldots, \phi_k(\mathcal{M}_n)$
end procedure

(HRV) analysis. In this new dataset, for each minute, each item contains the values of 12 HRV parameters plus the class expressing whether or not in that item the subject experienced apnea. This is the version of the dataset we used in the experiments. For this dataset, we have not considered the data from all the healthy subjects; moreover, we discarded two sets of data where the class imbalance is too high. On this dataset, we performed *binary classification*.

- **Ohio T1DM**: The Ohio T1DM dataset [22] consists of the data recorded by continuously monitoring 12 subjects suffering from Type 1 Diabetes Mellitus (T1DM). Each subject underwent continuous glucose monitoring for about 8 weeks while on insulin pump therapy. The original goal was to perform an as-accurate-as-possible forecasting of blood glucose levels for each subject. In [8, 9], the original regression problem was cast into a *seven-class classification* problem, where each class represents a range of the glucose values predicted for the next 30 minutes, with the various ranges corresponding to the occurrence of dangerous situations such as hyper-glycemic peaks or hypo-glycemic events. This is the version of the dataset we used in the experimentation.
- **HAR70+**: The Human Activity Recognition (HAR) database [36] contains data from 18 older adults (70 years old or more) gathered through the use of cameras to identify six different daily physical activities: walking, stairs (up / down), shuffling, standing, sitting, and lying. For each of the 18 participants, each item in the corresponding subset contains the values of 161 parameters extracted from five-second time windows (250 samples at 50 Hz), plus the class corresponding to the action being performed in those five seconds. Since the data from three subjects showed very high class imbalance, we have discarded them and have considered only the data from the other 15 subjects. On this dataset, we performed *seven-class classification*.

It is worth noting that each of these datasets makes reference to the digital health field and contains sets of data, each of which is specific to a subject, so an FL approach is well suited for them.

Table 2: Datasets used in the experimentation.

Dataset	Reference paper(s)	Number of parameters	Number of classes	Number of clients
Apnea-ECG	[30]	12	2	12
Ohio T1DM	[8, 9]	39	7	12
HAR70+	[36]	161	7	15

Table 2 summarizes the characteristics of the datasets, namely the reference to the version of the dataset used in the experiments; the number of parameters in the dataset; the number of output classes; and, the number of subjects considered in the experiments, which corresponds to the number of clients used.

4.2 Implementation and parameter setting

The NEvoFed algorithm has been implemented in Python starting from the `NEAT-Python` library, which is freely downloadable from [3]. It uses a configuration file in which different types of parameters have to be set [4]. The setting of the parameters modified with respect to the default values as a result of a preliminary tuning process is reported in Table 3. Note that the kind of NNs is different for the Ohio T1DM dataset in that it is a time series forecasting problem (hence requiring recurrence), cast into a classification one, while the other two datasets are related to pure classification.

As regards the canonical FL algorithm we compare with, we have used the framework `Flower` [6] for building an FL algorithm with aggregation based on averaging (FedAvg). Also for FedAvg, a preliminary tuning phase has been performed for each dataset to identify a suitable NN architecture. Figure 1 reports the NN architecture configurations used for the FedAvg algorithm, with their related parametrization.

Table 3: NEvoFed parameter configuration.

Parameter	Value
Neural network	recurrent (Ohio T1DM) feed-forward (Apnea-ECG, HAR70+)
Initial connections	no hidden layer
Population size	200
No. of epochs	50
No. of generations per epoch	1
Activation functions	sigmoid, tanh, relu, sin, softplus
Aggregation functions	sum, product
Activation/Aggregation mut. rate	0.4/0.4
Connection add/delete probability	0.4/0.1
Node add/delete probability	0.4/0.1
Elitism, species elitism	1, 1
Species max stagnation	10

It should be noted here that, given that in the three datasets considered the number of subjects is relatively low, we have chosen to avoid the client selection step for both NEvoFed and FedAvg, thus allowing, for both algorithms, all the clients to communicate with the server at each epoch (note that, in this way, every epoch corresponds to one generation of NE). Finally, 20 independent runs (with random initialization) with a number of epochs equal to 50 have been run for each algorithm.

To evaluate the effectiveness of the solutions found, we have adopted the weighted F1 score $\mathrm{F1}_w$, which takes into account the contributions of the F1 scores computed for each class, weighted based on the number of instances belonging to the class:

$$\mathrm{F1}_w = \frac{1}{c}\sum_{n=1}^{c} p_i \cdot \mathrm{F1}_i \tag{2}$$

where c represents the number of classes present in the data set, p_i indicates the percentage of instances in the i-th class, and $\mathrm{F1}_i$ stands for the F1 score calculated for the i-th class.

Finally, the Apnea-ECG and HAR70+ datasets have been split into a 70/30% training and test set, while, for the Ohio T1DM dataset, we have used the splitting provided in [22].

5 RESULTS

Table 4 report the results for Apnea-ECG, Ohio T1DM, and HAR70+, respectively, computed on all the 20 runs by considering for each of them the performance exhibited by the best global model over each subject. Moreover, Table 5 shows the aggregated results of the best models found among all the runs, obtained by averaging the performance across all the subjects of each dataset. While we observe that FedAvg obtains the best subject-wise accuracy in about 64% of the cases, we also observe that, when aggregating performance, NEvoFed obtains better average performance over all the datasets.

Statistical analysis has been conducted by considering the distributions of the fitness values of the global best solutions found by the two algorithms; namely, a two-tailed non-parametric Mann-Whitney U test has been run for each dataset, with a confidence level $\alpha = 0.05$. The null hypothesis of statistical equivalence can be rejected on the Apnea-ECG ($p = 0.001$) and Ohio T1DM ($p = 10^{-5}$) datasets, while it cannot be rejected for HAR70+ ($p = 0.8534$).

One possible explanation for the better performance of NEvoFed could be that our paradigm allows for finding models that are more tailored to the entire dataset. This is significantly different from the canonical FL paradigm. In fact, while the canonical FL methodology optimizes the models locally and then aggregates the models' weights, in our approach the best model is evolved *simultaneously* by all the clients, meaning that the phenotype (i.e., the NN) of the best solution has been shaped by *all* the clients simultaneously, making each promising candidate *directly* interact with all the dataset. This property is highly desirable, as it allows learning a model that maximizes the performance on the entire dataset (i.e., the union of all the clients' datasets) and not only on each specific local subset.

A more detailed view of the distribution of the best model performance across clients is shown in Figure 2. Here, we observe that the performance distribution of NEvoFed tends to be skewed towards higher values of weighted F1 scores with respect to FedAvg.

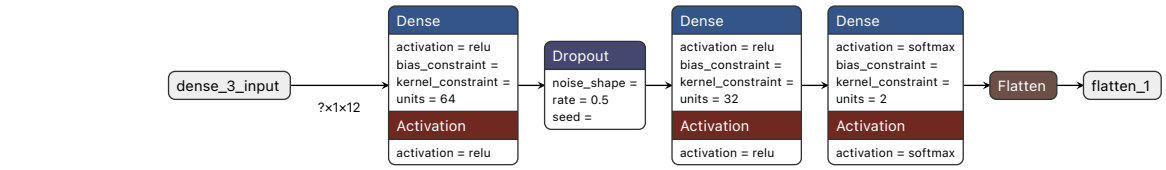
5.1 Discussion

Based on the results obtained in our experiments, we can highlight multiple advantages of our approach:

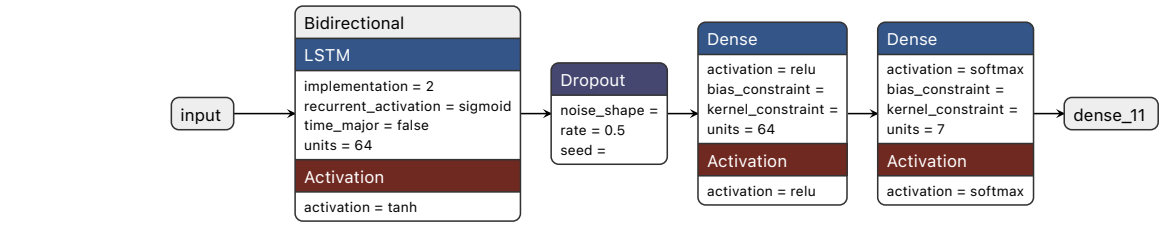
- *Lower NN complexity*: NEvoFed allows obtaining NNs with much lower complexity (i.e., with a very low number of neurons); as said, this implies that these NNs can be executed consuming fewer hardware resources, so they are better suited for low-power and memory-limited devices.
- *A step towards explainability*: the NNs evolved by NEvoFed tend to be very compact, potentially allowing for the interpretation of their behavior. In fact, given the conciseness of the achieved NNs, for the three problems considered, the best NNs obtained show an easy-to-spot connection between a very small subset of the dataset input parameters and the different classes; this can be seen in Figure 3.
  For the Apnea-ECG dataset, it can be seen that, out of the 12 input parameters, just three are relevant for the classification (see Figure 3a): the parameters V-9 and V-7 have a direct relationship with the unhealthy status (subject experiencing apnea), whereas V-10 implies a healthy status.
  For the Ohio T1DM dataset, instead, out of the 39 available input parameters, the best NN evolved (Figure 3b) uses as inputs only the glucose level at time t and three of its backward finite difference derivatives. By inspecting the weights of the NN, we observe that the model can be written as:

$$\begin{aligned} G(t+6) = ReLU(&0.97 \cdot G(t) + 0.09 \cdot \nabla_{12} G(t) \\ &+ 0.69 \cdot \nabla_{11} G(t) + 2.26 \cdot \nabla_{9} G(t) + 0.22) \end{aligned} \tag{3}$$

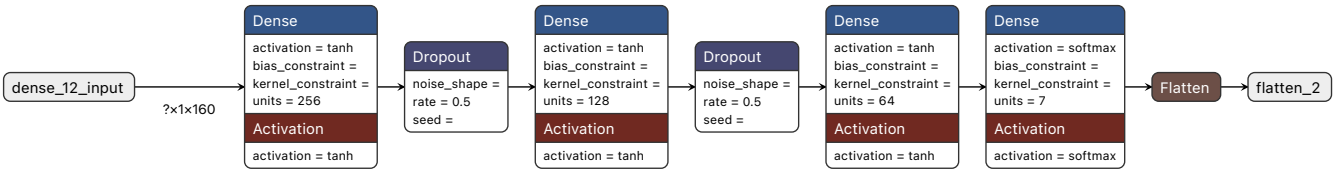
  which clearly shows that the prediction of the glucose at step $t+6$ depends on the current level plus an offset depending on the backward finite difference derivatives of the glucose level at $t-12$, $t-11$, and $t-9$ timesteps prior to the prediction (i.e., 1 hour, 55 minutes, and 45 minutes earlier, respectively). In other words, the glucose level in the future depends on both the current level and a combination of the variations of its trend in the recent past. Conciseness is even more evident for the HAR70+ task, where only 5 out of the 161 parameters are employed to recognize human activities. This means that an automatic selection of the



(a) Apnea-ECG



(b) Ohio T1DM



(c) HAR70+

Figure 1: Neural network architectures used for the FedAvg algorithm.

Table 4: Comparison, computed on the 20 runs available for each subject, of the weighted F1 score exhibited on the test set by the global models found on the three datasets by FedAvg and NEvoFed.

Method	Apnea-ECG – Subject ID											
	3	5	7	8	9	11	13	14	15	16	19	20
FedAvg	**0.86 ± 0.02**	**0.80 ± 0.02**	**0.88 ± 0.02**	**0.76 ± 0.01**	**0.86 ± 0.03**	0.66 ± 0.03	0.75 ± 0.04	**0.85 ± 0.01**	0.71 ± 0.02	0.79 ± 0.02	**0.86 ± 0.00**	0.69 ± 0.01
NEvoFed	**0.86 ± 0.02**	**0.80 ± 0.04**	0.77 ± 0.02	0.74 ± 0.02	**0.86 ± 0.03**	**0.70 ± 0.02**	**0.87 ± 0.02**	**0.85 ± 0.03**	**0.78 ± 0.03**	**0.82 ± 0.01**	0.72 ± 0.03	**0.88 ± 0.03**

Method	Ohio T1D – Subject ID											
	540	544	552	559	563	567	570	575	584	588	591	596
FedAvg	0.68 ± 0.00	**0.78 ± 0.00**	0.69 ± 0.01	0.76 ± 0.00	0.68 ± 0.00	**0.71 ± 0.00**	**0.82 ± 0.00**	0.68 ± 0.01	0.70 ± 0.00	**0.73 ± 0.01**	0.64 ± 0.01	0.70 ± 0.01
NEvoFed	**0.69 ± 0.00**	0.77 ± 0.01	**0.72 ± 0.00**	**0.77 ± 0.00**	**0.74 ± 0.00**	0.70 ± 0.01	0.81 ± 0.01	**0.71 ± 0.00**	**0.71 ± 0.00**	0.73 ± 0.00	**0.68 ± 0.00**	**0.73 ± 0.00**

Method	HAR70+ – Subject ID														
	501	504	505	507	508	509	510	511	512	513	514	515	516	517	518
FedAvg	**0.92 ± 0.01**	**0.94 ± 0.00**	**0.91 ± 0.03**	**0.86 ± 0.01**	0.90 ± 0.02	**0.92 ± 0.00**	**0.92 ± 0.00**	**0.87 ± 0.01**	0.85 ± 0.01	0.83 ± 0.02	0.78 ± 0.04	0.87 ± 0.00	0.83 ± 0.01	0.84 ± 0.01	**0.89 ± 0.01**
NEvoFed	0.91 ± 0.03	0.93 ± 0.03	0.90 ± 0.07	**0.86 ± 0.05**	**0.93 ± 0.05**	0.83 ± 0.07	0.85 ± 0.08	0.81 ± 0.06	**0.87 ± 0.02**	**0.85 ± 0.06**	**0.80 ± 0.08**	**0.88 ± 0.06**	**0.84 ± 0.03**	**0.85 ± 0.09**	0.87 ± 0.11

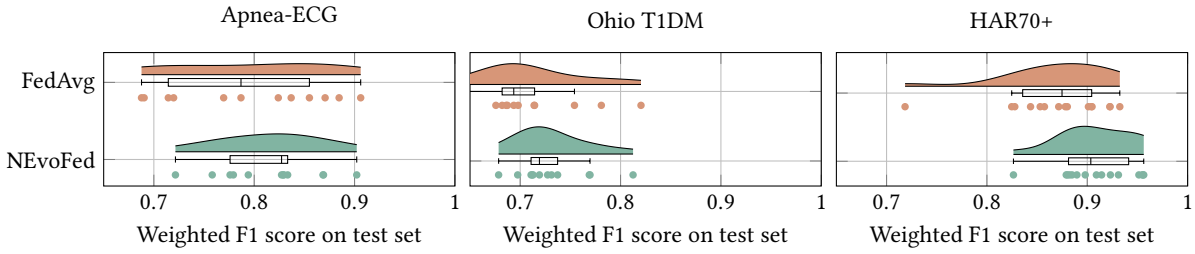
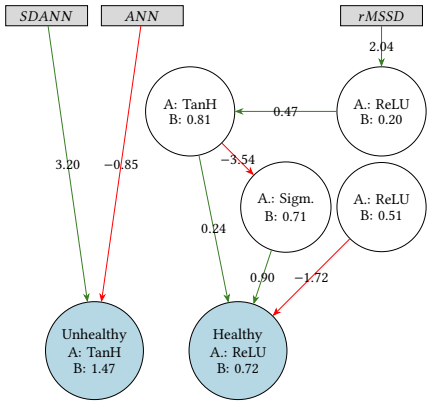


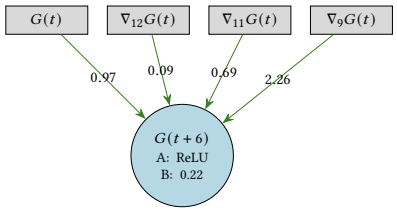
Figure 2: Performance of the best global model found for each dataset by NEvoFed across each client.

Table 5: Weighted F1 score on the test sets of the best models found among all the runs on all the clients.

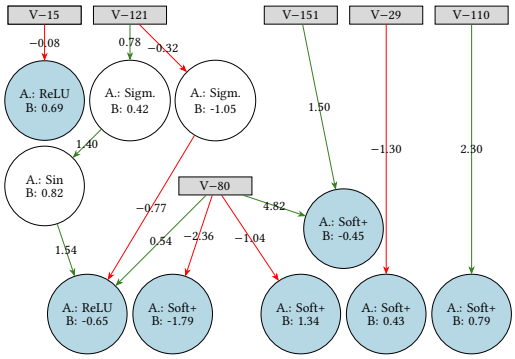
Dataset	FedAvg	NEvoFed
Apnea-ECG	0.7955 ± 0.0784	**0.8156 ± 0.0518**
Ohio T1DM	0.7122 ± 0.0500	**0.7314 ± 0.0366**
HAR70+	0.8695 ± 0.0540	**0.9091 ± 0.0372**



(a) Apnea-ECG



(b) Ohio T1DM



(c) HAR70+

Figure 3: Best NNs evolved by NEvoFed. Grey rectangles represent inputs, white circles represent inner nodes, and azure circles represent output nodes. "A" stands for "Activation", while "B" stands for "Bias".

most relevant features is performed and can be provided to the experts for an evaluation of their meaningfulness.

- *Simpler design phase*: in the FL approach implemented in `Flower`, the structure of the NNs must be the same for all the clients. This poses problems in the design phase because the user must make decisions *a priori* on the structure in terms of the number of layers, number of nodes, etc.; these decisions could influence the quality of the obtained results, which could lead to a trial-and-error design approach. This is clearly not the case with NEvoFed, which does not require to specify the NN architecture beforehand. Instead, such architecture is automatically evolved by all the involved clients together. Thus, we can say that the process automatically learns the best architecture for the problem at hand without requiring any prior knowledge. Moreover, as shown in Figure 3, the NNs evolved with NEvoFed are extremely small compared to those usually designed for canonical FL (e.g., the ones shown in Figure 1). This allows for smaller communication costs, as the amount of information that has to be sent to the clients is substantially lower, and also enables inference on tiny devices, as the NNs use very small amounts of FLOPs, which can be computed by essentially any embedded system.

Yet, our approach presently suffers also from some limitations:

- *Number of clients*: We have considered datasets requiring the use of up to 20 clients, whereas we know that many datasets exist for FL that require the use of hundreds or even thousands of clients. We will be working on this issue and will improve our present methodology so that this issue will be positively solved.
- *Types of data*: All the datasets are structured in the form of tables and are well-suited for classification purposes. However, some changes and improvements will be needed to use our method for evolving deep NNs capable of directly classifying images. We will overcome this limitation in the next version of our methodology by considering NE algorithms specifically devised for deep NNs.

6 CONCLUSIONS

In this paper, a new approach to FL called NEvoFed has been proposed, in which NE takes place independently on clients. This implies that the NN structures may be different both on the same client and on different clients; this feature prevents users from dealing with the burden of needing to decide *a priori* the NN structure to be used on all clients.

This approach has been tested on three datasets well suited to FL. The experiment outcomes have shown that very compact NNs can be obtained, without compromising the classification ability compared to a canonical FL approach. Furthermore, the compactness of the NNs allows for a step towards explainability, which is highly desirable in application areas such as digital health, from which the three datasets come.

As part of future developments, we plan to conduct larger experiments that incorporate more challenging datasets, such as image datasets, and test our approach on a greater number of clients. Additionally, we intend to make use of different NE algorithms to evolve deep NNs able to solve more demanding tasks.

ACKNOWLEDGMENT

This work was partially supported by: the grant Hub Life Science-Advanced Diagnosis (HLS-AD), PNRR PNC-E3-2022-23683266 PNC-HLS-DA, INNOVA – CUP: E63C22003780001, funded by the Italian Ministry of Health under the National Complementary Plan Innovative Health Ecosystem - Unique Investment Code: PNC-E.3; and the European Union (project no. 101071179).

REFERENCES

[1] Agarwal, Mayank and Yurochkin, Mikhail and Sun, Yuekai. 2022. *Personalization in Federated Learning*. Springer, Cham, Switzerland, 71–98.

[2] Alam, Samiul and Liu, Luyang and Yan, Ming and Zhang, Mi. 2022. FedRolex: Model-Heterogeneous Federated Learning with Rolling Sub-Model Extraction. In *Advances in Neural Information Processing Systems*, Vol. 35. Neural Information Processing Systems Foundation, San Diego, CA, USA, 29677–29690.

[3] Alan McIntyre and Matt Kallada and Cesar G. Miguel and Carolina Feher da Silva. 2019. NEAT-Python. https://github.com/CodeReclaimers/neat-python.

[4] Alan McIntyre and Matt Kallada and Cesar G. Miguel and Carolina Feher da Silva. 2020. NEAT-Python Documentation: Configuration file description. https://neat-python.readthedocs.io/en/latest/config_file.html

[5] Bergstra, James and Bengio, Yoshua. 2012. Random search for hyper-parameter optimization. *Journal of Machine Learning Research* 13, 2 (2012), 281–305.

[6] Beutel, Daniel J and Topal, Taner and Mathur, Akhil and Qiu, Xinchi and Fernandez-Marques, Javier and Gao, Yan and Sani, Lorenzo and Kwing, Hei Li and Parcollet, Titouan and Gusmão, Pedro PB de and Lane, Nicholas D. 2020. Flower: A Friendly Federated Learning Research Framework. arXiv preprint arXiv:2007.14390.

[7] Chai, Zheng-Yi and Yang, Chuan-dong and Li, Ya-Lun. 2023. Communication efficiency optimization in federated learning based on multi-objective evolutionary algorithm. *Evolutionary Intelligence* 16, 3 (2023), 1033–1044.

[8] De Falco, I and Della Cioppa, A and Koutny, T and Scafuri, U and Tarantino, E. 2024. Model-Free-Communication Federated Learning: Framework and application to Precision Medicine. *Biomedical Signal Processing and Control* 87 (2024), 105416.

[9] De Falco, Ivanoe and Della Cioppa, Antonio and Koutny, Tomas and Ubl, Martin and Krcma, Michal and Scafuri, Umberto and Tarantino, Ernesto. 2023. A Federated Learning-Inspired Evolutionary Algorithm: Application to Glucose Prediction. *Sensors* 23, 6 (2023), 2957.

[10] Fang, Jiemin and Sun, Yuzhu and Zhang, Qian and Li, Yuan and Liu, Wenyu and Wang, Xinggang. 2020. Densely connected search space for more flexible neural architecture search. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, New York, NY, USA, 10628–10637.

[11] Hinton, Geoffrey and Vinyals, Oriol and Dean, Jeff. 2015. Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531.

[12] Horvitz, Eric and Mulligan, Deirdre. 2015. Data, privacy, and the greater good. *Science* 349, 6245 (2015), 253–255.

[13] Iacca, Giovanni. 2013. Distributed optimization in wireless sensor networks: an island-model framework. *Soft Computing* 17, 12 (2013), 2257–2277.

[14] Jin, Yaochu. 2011. Surrogate-assisted evolutionary computation: Recent advances and future challenges. *Swarm and Evolutionary Computation* 1, 2 (2011), 61–70.

[15] Konečnỳ, Jakub and McMahan, Brendan and Ramage, Daniel. 2015. Federated optimization: Distributed optimization beyond the datacenter. arXiv preprint arXiv:1511.03575.

[16] Konečnỳ, Jakub and McMahan, H Brendan and Ramage, Daniel and Richtárik, Peter. 2016. Federated optimization: Distributed machine learning for on-device intelligence. arXiv preprint arXiv:1610.02527.

[17] Li, Liam and Talwalkar, Ameet. 2020. Random search and reproducibility for neural architecture search. In *Uncertainty in artificial intelligence*. PMLR, Tel Aviv, Israel, 367–377.

[18] Liu, Xin and Zhao, Jianwei and Li, Jie and Xu, Dikai and Tian, Shan and Cao, Bin. 2022. Large-Scale Multiobjective Federated Neuroevolution for Privacy and Security in the Internet of Things. *IEEE Internet of Things Magazine* 5, 2 (2022), 74–77.

[19] Lorraine, Jonathan and Vicol, Paul and Duvenaud, David. 2020. Optimizing millions of hyperparameters by implicit differentiation. In *International Conference on Artificial Intelligence and Statistics*. PMLR, Palermo, Italy, 1540–1552.

[20] Lu, Zhichao and Whalen, Ian and Boddeti, Vishnu and Dhebar, Yashesh and Deb, Kalyanmoy and Goodman, Erik and Banzhaf, Wolfgang. 2019. NSGA-Net: neural architecture search using multi-objective genetic algorithm. In *Genetic and Evolutionary Computation Conference*. ACM, New York, NY, USA, 419–427.

[21] Ma, Xiaosong and Zhang, Jie and Guo, Song and Xu, Wenchao. 2022. Layer-wised Model Aggregation for Personalized Federated Learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, New York, NY, USA, 10082–10091.

[22] Marling, Cindy and Bunescu, Razvan. 2020. The OhioT1DM dataset for blood glucose level prediction: Update 2020. In *CEUR Workshop Proceedings*, Vol. 2675. CEUR-WS.org, Aachen, Germany, 71.

[23] McDonnell, John Robert and Waagen, D. 1994. Evolving recurrent perceptrons for time-series modeling. *IEEE Transactions on Neural Networks* 5, 1 (1994), 24–38.

[24] Mora, Alessio and Tenison, Irene and Bellavista, Paolo and Rish; Irina. 2022. Knowledge Distillation for Federated Learning: a Practical Guide. arXiv preprint arXiv:2211.04742.

[25] Morell, José Ángel and Dahi, Zakaria Abdelmoiz and Chicano, Francisco and Luque, Gabriel and Alba, Enrique. 2022. Optimising Communication Overhead in Federated Learning Using NSGA-II. In *International Conference on the Applications of Evolutionary Computation (Part of EvoStar)*. Springer, Cham, Switzerland, 317–333.

[26] Moriarty, David E and Miikkulainen, Risto. 1997. Forming neural networks through efficient and adaptive coevolution. *Evolutionary Computation* 5, 4 (1997), 373–399.

[27] Penzel, Thomas and Moody, George B and Mark, Roger G and Goldberger, Ary L and Peter, J Hermann. 2000. The apnea-ECG database. In *Computers in Cardiology*, Vol. 27. IEEE, New York, NY, USA, 255–258.

[28] Rahimi, Mohammad Mahdi and Bhatti, Hasnain Irshad and Park, Younghyun and Kousar, Humaira and Kim, Do-Yeon and Moon, Jaekyun. 2023. EvoFed: Leveraging Evolutionary Strategies for Communication-Efficient Federated Learning. In *Advances in Neural Information Processing Systems*. Neural Information Processing Systems Foundation, San Diego, CA, USA, 14 pages.

[29] Rasmussen, Carl Edward. 2003. Gaussian processes in machine learning. In *Summer school on machine learning*. Springer, Berlin Heidelberg, Germany, 63–71.

[30] Sannino, Giovanna and De Falco, Ivanoe and De Pietro, Giuseppe. 2014. Monitoring obstructive sleep apnea by means of a real-time mobile system based on the automatic extraction of sets of rules through differential evolution. *Journal of Biomedical Informatics* 49 (2014), 84–100.

[31] Shaoxiong, Ji and Yue, Tan and Teemu, Saravirta and Zhiqin, Yang and Lauri, Vasankari and Shirui, Pan and Guodong, Long and Anwar, Walid. 2023. Emerging Trends in Federated Learning: From Model Fusion to Federated X Learning. arXiv preprint arXiv:2102.12920.

[32] Stanley, Kenneth O and Clune, Jeff and Lehman, Joel and Miikkulainen, Risto. 2019. Designing neural networks through neuroevolution. *Nature Machine Intelligence* 1, 1 (2019), 24–35.

[33] Stanley, Kenneth O and Miikkulainen, Risto. 2002. Evolving neural networks through augmenting topologies. *Evolutionary Computation* 10, 2 (2002), 99–127.

[34] Swersky, Kevin and Snoek, Jasper and Adams, Ryan Prescott. 2014. Freeze-thaw Bayesian optimization. arXiv preprint arXiv:1406.3896.

[35] Szegedi, Gábor and Kiss, Péter and Horváth, Tomás. 2019. Evolutionary Federated Learning on EEG-data. In *Information Technologies – Applications and Theory*. CEUR-WS.org, Aachen, Germany, 71–78.

[36] Ustad, Astrid and Logacjov, Aleksej and Trollebø, Stine Øverengen and Thingstad, Pernille and Vereijken, Beatrix and Bach, Kerstin and Maroni, Nina Skjæret. 2023. Validation of an Activity Type Recognition Model Classifying Daily Physical Behavior in Older Adults: The HAR70+ Model. *Sensors* 23, 5 (2023), 2368.

[37] Wan, Alvin and Dai, Xiaoliang and Zhang, Peizhao and He, Zijian and Tian, Yuandong and Xie, Saining and Wu, Bichen and Yu, Matthew and Xu, Tao and Chen, Kan and others. 2020. Fbnetv2: Differentiable neural architecture search for spatial and channel dimensions. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, New York, NY, USA, 12965–12974.

[38] Whitley, Darrell and Dominic, Stephen and Das, Rajarshi and Anderson, Charles W. 1993. Genetic reinforcement learning for neurocontrol problems. *Machine Learning* 13 (1993), 259–284.

[39] Yang, Timothy and Andrew, Galen and Eichner, Hubert and Sun, Haicheng and Li, Wei and Kong, Nicholas and Ramage, Daniel and Beaufays, Françoise. 2018. Applied federated learning: Improving google keyboard query suggestions. arXiv preprint arXiv:1812.02903.

[40] Yao, Xin. 1999. Evolving artificial neural networks. *Proceedings of the IEEE* 87, 9 (1999), 1423–1447.

[41] Zheng, Xiawu and Ji, Rongrong and Tang, Lang and Zhang, Baochang and Liu, Jianzhuang and Tian, Qi. 2019. Multinomial distribution learning for effective neural architecture search. In *IEEE/CVF International Conference on Computer Vision*. IEEE, New York, NY, USA, 1304–1313.

[42] Zhong, Zhao and Yang, Zichen and Deng, Boyang and Yan, Junjie and Wu, Wei and Shao, Jing and Liu, Cheng-Lin. 2020. Blockqnn: Efficient block-wise neural network architecture generation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43, 7 (2020), 2314–2328.

[43] Zhu, Hangyu and Jin, Yaochu. 2019. Multi-objective evolutionary federated learning. *IEEE Transactions on Neural Networks and Learning Systems* 31, 4 (2019), 1310–1322.

[44] Zhu, Hangyu and Jin, Yaochu. 2021. Real-time federated evolutionary neural architecture search. *IEEE Transactions on Evolutionary Computation* 26, 2 (2021), 364–378.

[45] Zhu, Hangyu and Zhang, Haoyu and Jin, Yaochu. 2021. From federated learning to federated neural architecture search: a survey. *Complex & Intelligent Systems* 7 (2021), 639–657.

[46] Zoph, Barret and Le, Quoc V. 2016. Neural architecture search with reinforcement learning. arXiv preprint arXiv:1611.01578.