Using Federated Learning to Achieve Proactive Context-Aware IoT Environments*

Rubén Rentero-Trejo†, Daniel Flores-Martín, Jaime Galán-Jiménez, José García-Alonso, Juan Manuel Murillo and Javier Berrocal

University of Extremadura, Spain
E-mail: rrenterot@unex.es; dfloresm@unex.es; jaime@unex.es; jgaralo@unex.es; juanmamu@unex.es; jberolm@unex.es
†*Corresponding Author*

Received 16 July 2021; Accepted 12 September 2021;
Publication 23 November 2021

Abstract

The Internet of Things (IoT) is more present in our daily lives than ever before, turning everyday physical objects into smart devices. However, these devices often need excessive human interaction before reaching their best performance, making them time-consuming and reducing their usability. Nowadays, Artificial Intelligence (AI) techniques are being used to process data and to find ways to automate different behaviours. However, achieving learning models capable of handling any situation is a challenging task, worsened by time training restrictions. This paper proposes a Federated Learning solution to manage different IoT environments and provide accurate predictions, based on the user's preferences. To improve the coexistence between devices and users, this approach makes use of other users' previous behaviours in similar environments, and proposes predictions for newcomers

*Rentero-Trejo, R.; Flores-Martín, D.; García-Alonso, J.; Galán-Jiménez, J.; Murillo, J.M.; Berrocal, J. Towards Proactive Context-Aware IoT Environments by means of Federated Learning. To appear in the proceedings of the 2021 International Workshop on Big data driven Edge Cloud Services (BECS 2021), May 18, 2021.

Journal of Web Engineering, Vol. 21_1, 53–74.
doi: 10.13052/jwe1540-9589.2113

to the federation. Also, for existing participants, it provides a closer personalization, immediate availability and prevents most manual interactions. The approach has been tested with synthetic and real data and identifies the actions to be performed with 94% accuracy on regular users.

Keywords: Federated learning, mobile devices, context-aware, IoT.

1 Introduction

There are currently 12 billion connected IoT devices [15] and nearly 4 billion smartphones around the world [21]. These devices come equipped with complex sensors, computing, and communication capabilities. The amount of information handled by end devices opens up endless possibilities when it comes to how IoT can improve our lives [23].

More and more users are using IoT devices in their daily lives, and concepts such as Smart-Home [2], Health-care [23], or Smart-cities [3] are emerging. These specialized devices often need configuration and human interaction to achieve their most optimal performance. However, with the growing number of devices [15], it is becoming more difficult to manage them properly without investing a lot of time. In addition, when a user adds new devices, the effort increases accordingly, which may reduce the perceived benefits of the IoT.

AI, Deep Learning (DL) and Federated Learning (FL) techniques emerged in order to analyze different information and to make predictions in order to automate or reduce the effort of users doing different tasks [19]. Hence, these techniques could be used to reduce the effort required to use the IoT. In addition, several paradigms, such as Edge and Mist Computing (EC, MC) bring computing resources closer to the user. They are able to execute these techniques on these resources in order to reduce the response time, increase the privacy and, thus, provide a good quality of service [10]. Specifically, FL tries to exploit the characteristics of distributed computing.

Some solutions to get this personalized behavior are based on large recommender systems leading to heavy architectures [5, 12], making them hard to deploy in cost-effective devices with lower specifications (edge nodes or smart devices), usually present in this kind of environments. Others define behaviors based on rule sets [12], but they are limited, do not cover all possible cases, and do not adapt dynamically to new situations. A truly smart environment should be able to adapt its behaviour to the user independently

if it is a known or a new user. Finally, most of the proposed solutions exploiting the characteristics of edge/mist devices [13, 14, 29] study the statistical heterogeneity of user data, but do not attempt to find a personalized and context-aware solution for each user.For instance, these solutions try to identify the most common configuration or behaviour of an IoT device according to previous interactions, but they are not usually aware of the context nor the preferences of the specific user interacting with it. Approaches are needed to deal with any user and environment, providing context-aware predictions, but without heavy deployments and long adaptation periods.

This paper proposes a novel solution based on FL where mobile devices take a more active role for learning the preferences of their owners and adapting the behavior of already known and new smart environments to these preferences. To get context-aware and personalized predictions, this solution proposes the consideration of two models. First, a global model with the knowledge generated in the federation and which provides predictions to new users and/or new environments. Second, a personalized model which adjusts to the needs of a particular user for already known environments. Both models can be retrained to meet the needs of the federation and the users. This approach allows fast personalization and deployments, offers predictions in every environment the user visits and manages multiple environments.

The rest of the document is structured as follows. Section 2 presents the motivations of this work. Section 3 explains the proposal, divided into Data Model and Architecture. Section 4 presents a two-phase validation, evaluating the approach with synthetic and real data and providing a brief comparison. Section 5 presents some related works. Finally, concluding remarks are given in Section 6.

2 Motivation

IoT devices make people's lives easier by automating a wide variety of tasks. Nowadays, it is easy to find devices such as light bulbs, TVs or wearables in any workplace or home environment. Managing one device is a simple task after the initial configuration. However, when the number of devices is increased, the device management becomes more difficult and time-consuming. Specially, when the interactions among them have also to be managed.

If we analyze most IoT devices, their functionality is based on running specific, direct and not very complex commands, which are usually linked to

a user's patterns repeated over time and with minimal variations. Thus if we know the actions to be performed, we can automate them. In short, we need a system to make decisions in place of users.

Normally, the behavior is device and environment-dependent and can be reproduced for similar devices in similar environments. We call *environment* to the set of different IoT devices and the circumstances/context around them (e.g. time, place, mode, patterns, etc.). Common examples are workplaces or home. People usually move through different environments, and automation should persist in all of them, always adapted to the user's preferences.

To better understand this problem, an example is proposed below: *Suppose a user named John, a technology enthusiast who commonly interacts with IoT devices. John wakes up every morning at the same time, turns on the lights and gets ready for work. After finishing, he turns off all devices and goes to his office. During working hours, he prefers a specific light and temperature, as well as background music to help him concentrate. When he arrives home, he wants his devices to recognize his arrival, turn on the lights, and tune the TV to his favorite channel. John has tried programming his devices to trigger at specific times but his schedule is flexible and do not achieve the desired behavior. He usually sleeps late, but recently has more work than usual and decides to go to sleep earlier, so he must turn off the TV and lights manually. Changes in his schedule required him to set up multiple configurations and manual changes, which makes John wonder if the deployment of smart devices is worth it*.

As shown, approaches that learn from user behavior and contextual information are needed to automate actions without requiring users to reconfigure their devices every time their context change. The availability should be immediate from the model's acquisition, so it would not be reasonable to offer a model which requires long adaptation periods. Recommendation systems offer a partial solution to these situations; however, dependence on cloud services causes high latency, issues in low connectivity situations and privacy concerns. In this context, MC solutions that can make decisions locally without sharing sensitive data are desirable.

While avoiding cloud alternatives, it is necessary to solve how to allow knowledge to travel between different environments, since the system needs to operate in all of them. Since the beginning of this century, smartphones have been classified as personal objects that accompanies people in their daily activities [26]. They are the users' closest elements. In addition, considering their computational possibilities, they present a promising option. Therefore, mobile devices can be the main learning nodes and will act as storage and

as information vehicles, capable of moving between environments. This solution would allow John to change environments freely, transfer previously acquired knowledge to a new environment and use it to make new decisions.

To ensure new users fast deployments and customization tasks, the *global knowledge* offered by FL is used in this proposal. FL is a learning technique that allows devices to collaboratively learn a shared prediction model by keeping all training data on them, sharing only gradients or weights, instead of raw data. It enables smarter models, low latency, lower power consumption and ability to use the model immediately, while improving privacy [17].

Dealing with a mobile environment means dealing with Non-IID [30], unbalanced and massively distributed data, problems that do not occur in centralized environments. Each user has different preferences, actions, favorite devices, etc. This nature has a high impact on traditional ML models [11, 20] and is one of the reasons to use FL and the Federated Averaging (FedAvg) algorithm, a generalization of FedSGD [24]. FedAvg exchanges the updated weights without sharing personal data, and performs an average among the participants to obtain a generalized model [17].

Usually in FL, once the global model is complete, it is provided to users by transferring weights and retrained to perform the personalization. Hybrid models provide a middle ground between generalization and personalization. Although it could be a good approach for existing users, for new users the model could be biased and too general for their preferences. Therefore, changing the behavior of this hybrid model to get more personalized predictions for each new user could require a higher effort than having both a general model for the unknown behavior and a local one for the more personalized behavior [6].

Therefore, we present a solution with a double model. The global model has the federation's knowledge and will evolve with it. The strength of this model is its ability to offer predictions to new users and those who encounter new environments. The local model has the knowledge of the individual user. It provides the highest possible degree of personalization to provide effective predictions. This way, any kind of user can deal with any kind of environment.

By keeping two independent models, the user is always allowed to have a copy of the global model, which is sensitive to changes in the federation, without affecting personalization on the local model.

Thanks to this approach, devices will receive commands and automatically react based on each user preferences, making them focus only on exceptional situations that do not follow a usual behavior pattern, greatly reducing the user's manual actions.

3 Proactive Context-Aware IoT Environments

The aim of this proposal is to achieve proactive IoT environments, adapted to the user context and needs. To achieve this goal, mobile devices computing capabilities are exploited through FL, as companion devices that can interact in the user's different environments, reducing the dependency on the cloud. FL is applied by using complementary models to address different situations: a global model collaboratively trained which generalizes the federation behavior; while another model of similar features is in charge of carrying out the personalization of user behaviors. The first model allows new users to have a starting point for decision making while the second model has personalization as its objective.

In order to detail the architecture of the proposal, this section is divided into two: first, a *data model* analysis to deal with IoT environments, user preferences and context; and second, the required *architecture* with a dual learning model.

3.1 Data Model

The following are the operations to be performed by the models and the data groups required to develop them. Also, an example of the tuples extracted during data collection according to these functions is shown in Table 1.

- **Environment**: It is used to identify IoT devices within the same environment and then generalize that behavior to new scenarios where the environment, or even the user himself, is different.
- **Device type**: It is specified to distinguish behaviors between different devices (e.g. actions performed on a TV from those on a light bulb).
- **Device ID**: It complements the above attribute since there may be several devices of the same type in one environment, and is used to discern them.
- **Connection type**: It is necessary to know the communication protocol used by the devices to add them to the environment and to send actions.

Table 1 Example tuples extracted during data collection

Env.	Device Type	Device ID	Conn. Type	Action	Time	Arrival	Prob.
Home	Air conditioner	AC1	WiFi	24°C	2019-10-14 15:07:51	False	95%
Home	SMARTTV	TV1	WiFi	Tele5	2019-10-14 22:00:03	True	80%
Office	SPEAKER	S2	BLE	On	2019-10-20 09:05:10	True	75%

It does not affect decision making but it is needed to allow effective action forwarding to the devices in the future.

- **Action**: It specifies the direct command to be executed on the IoT device and is the target variable for predictions.
- **Time**: The model must be able to understand the temporal aspect of events and identify patterns based on them. For now, the identified behaviors are those related to hours of the day (e.g. 8:00, 15:00) and weekdays (workdays, weekends, free days), but can be extended to long term patterns such as seasons of the year.
- **Arrival**: It indicates the user presence and can be implemented through mobile device location, being transparent to him. This information is useful to control time patterns, where a prediction must be performed but the user is not in the environment. It must be evaluated if the execution of the action makes sense when the user is not present.
- **Probability**: The reliability level is associated with the final predictions. It will be used later to finally decide on the action to be taken.

From a data perspective, we call *usual environments* (UE) to those statistically predominant in the user's dataset. Similarly, *new environments* (NE) are those with insufficient or no data. After a previous analysis of possible user behaviors, we identified 3 different situations: (1) The user is in his UE and requests predictions for an already known environment; (2) The user requests predictions after changing to a NE where devices or interactions are different; (3) It is a new user with no previous information.

3.2 Architecture

The main learning approach consists of neural networks (NNs) selected given FL's ease [20]. The presented solution is based on standard FL and *FedAvg*, adapted to use a dual model. A **global model** that is in charge of performing all the usual FL tasks (GM in Figure 1). Its goal is to create a global model from the knowledge of the federated users, then downloaded to the user's device and used as a global knowledge base. And a **local model** with similar characteristics to the previous one (LM in Figure 1), created on the device itself, that is in charge of training only with user data and without interference from external models (as in FL) to achieve a personalized behavior.

On the one hand, the **global model** is the one built from the knowledge of the entire network of users. Initially, each user has an empty model that is initialized by a set of parameters defined in the server (e.g. epochs, learning rate, etc), providing greater control of the server over the federation.

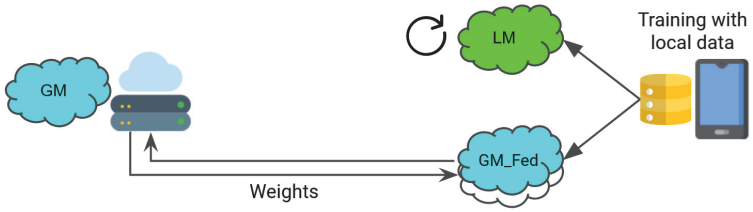


Figure 1 FL with a double local model architecture. Global model (GM) runs FL as usual by taking GM_Fed models as participants while Local model (LM) does the personalization.

Then, it is trained for several epochs, and the NN weights are serialized into an update package and sent to the server. This process occurs with all connected devices, and the server aggregates all packages to create the global model. This process is repeated until the set rounds or until the desired accuracy is obtained. At the end of the process, the user obtains this global model to provide a starting point for predictions.

This method allows devices to make predictions based on the general federated behavior. It is especially useful in situations where new users join the federation. Since there is no previous information about them, they do not have a personalized model to request predictions from. However, this global model allows predictions to be made, in exchange for a small penalty in the accuracy of the predictions. As for the aggregations carried out in the server, with FedAvg, each of the NN weights will be determined by the result of averaging the weights of NNs located in each device (McMahan et al. [17]).

In addition, some models are usually unable to learn in one round, leading to a negative impact on the global model from low-quality weights. To solve this problem in the implemented algorithm, those models are not aggregated and are given additional rounds to improve. This is used to prevent poisoning the global model and as a preventive measure against malicious users.

On the other hand, the **local model** is the one isolated in the device, i.e. this model will not be affected by the actions performed in the federation, and starts its learning from a clean model. It trains with device local data, in the same way that the downloaded global model does. However, this model does not share its weights with any server and remains independent of the federated environment. This model objective is to obtain the highest degree of personalization (skew) possible from device usage. The main benefit of this model lies in situations where the user is in one of his UEs to make accurate predictions adjusted to his preferences.

In short, the global model serves as a source of general knowledge, basically, the actions that would be taken by the federation in a similar situation. Eventually, the global model will be replaced by the personalized one, which best knows the particular user behavior. Both models compete to offer the best predictions; if one cannot give a sufficiently accurate prediction, the model's prediction with the highest reliability index for that situation will be chosen.

4 Validation

The validation of this work is divided into two phases: First, the approach is validated using synthetically generated data, in order to choose the best model configuration; Second, the approach is validated with real users to check its performance in real scenarios, as well as its impact on users' mobile devices. Then, our proposal is evaluated through an experimental comparison with two related works. This Section ends with a validation discussion.

4.1 Phase 1: Synthetic Data

4.1.1 Data generation

To measure the performance of our approach, a large amount of datasets are needed. At the moment of developing this work, there are no available datasets with the contextual characteristics to fulfill our needs. Therefore, we need to generate our own data. Synthetic data was generated with TheONE Simulator [1]. This tool aims to simulate message routing between nodes using Delay-Tolerant Networking routing algorithms. However, although its objective is different, it has been chosen because it can simulate movement patterns. The relevant patterns for this work are those simulating the movements of a person in his day-to-day life through different places. By modifying its core, this tool allowed us to simulate interactions with IoT devices in different environments.

A set of different scenarios were defined following the directions presented in Section 3.1, the details are: 4 different environments per user, each environment has a maximum of 10 devices, and 10 different actions. Additional context has been provided by specifying different work, leisure and sleep schedules over the day. Finally, human movement is implemented by TheONE. The main goal is to simulate scenarios similar to the one given in Section 2. For the sake of completion, 165 user datasets have been generated, being No-IID and unbalanced, up to a maximum of 1.500 tuples each.

4.1.2 Set-up

After an exhaustive data processing, each dataset has a total of 64 columns, generated from the source data by *One-Hot Encoding*, 54 of them will be taken as input features for NNs. Since we are dealing with a classification problem, the output data are 10 classes. To find the best configuration for our models, different tests were performed, following a specific process. First, we tested different neural network definitions, changing the depth and complexity of the hidden layers, including variations in the number of layers, nodes, and usage of techniques like Batch Normalization (BN) or Dropout (DO). The 4 most satisfactory configurations are shown in Figure 3(a) description. All layers have *ReLu* activation function, except for the output one that has a *Softmax*. Iterations are $epochs = 5$ and $n\_rounds = 50$.

Once the best NN configuration has been found and, being aware there are local models unable of learn from their users (i.e. outliers), 3 different quality filters have been tested to prevent GM poisoning: the *None* filter which allows any LM to be aggregated, followed by *Low* and *High* filters, which restrict aggregations to models with accuracies greater than 65% and 75% respectively. Additionally, SGD and ADAM optimizers have also been tested, since they are essential elements in FL. Their details are a *learning rate* of 0.015, and, specifically for SGD, a $momentum = 0.9$ and a $decay = \frac{lr}{n\_rounds}$.

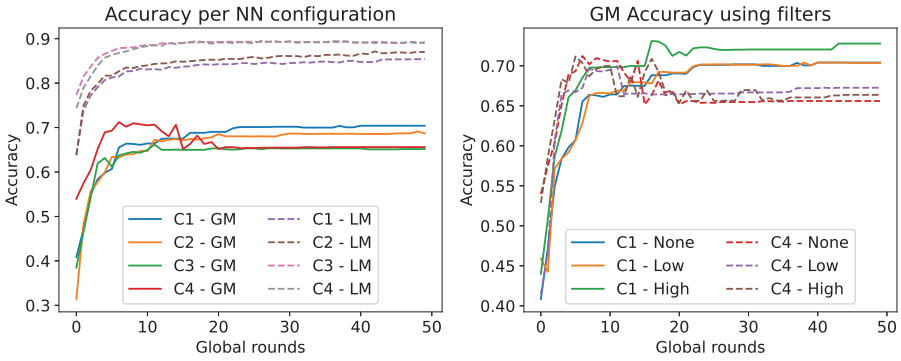
Finally, the 3 different situations defined at the end of Section 3.1 were tested. The goal is to give additional context to the obtained accuracies and show the real potential of this approach.

4.1.3 Results

The tested configurations have been evaluated under a standard evaluation (SE), where 50% of randomly taken data has been reserved for tests, regardless of the situation. Figure 2(a) shows how C1 (Configuration 1) performs better in the GM, being more stable with a superior convergence. In contrast, C4 (Configuration 4) poorly performs in GM in the long term, but shows promising results for personalization. From now on, C1 and C4 will be the focus of our study.

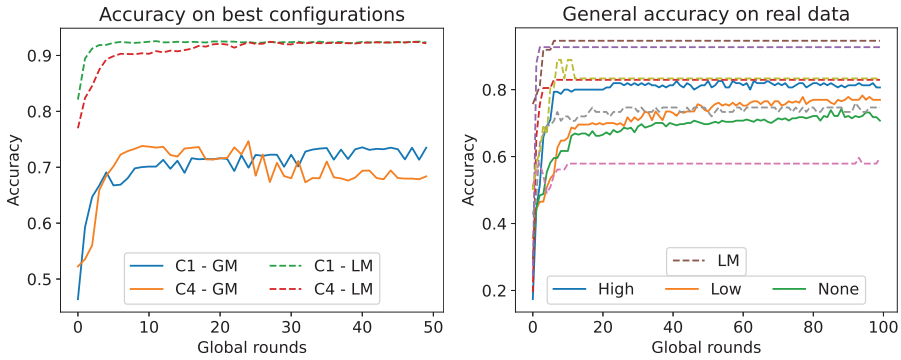
In terms of filters, Figure 2(b) shows little difference between the different filters. Nevertheless, a higher filter is slightly better and benefits the GM not only in accuracy, but also in security, as stated in 3.2.

To improve performance, SGD and ADAM optimizers have been compared, as ADAM often performs better than SGD. Taking the highest filter as basis, Figure 3(a) shows configurations 1 and 4 with ADAM, both performing better than their versions with SGD, in Figure 2(b). Finally, the best possible



(a) Accuracy per NN configuration. (b) GM Accuracy per filter.

Figure 2 Models performance on synthetic data: (a) Shows GM and LM accuracies for the selected configurations, based on SGD: C1 (2 hidden layers * 54 nodes), C2 (3 hidden layers * 108 nodes), C3 (2 hidden layers * 54 nodes, BN, DO), and C4 (3 hidden layers * 108 nodes, BN, DO). (b) Shows GM accuracy: All clients aggregated, *Low* and *High* requirements filter (65% and 75% min. accuracy).



(a) Performance on synthetic data. (b) Performance on real data.

Figure 3 Definitive models: Phase 1 (a) shows GM accuracy and the average accuracy from 165 LMs using *Adam* optimizer, the 2 best NN configurations and the *High* filter; Phase 2 (b) GM accuracy for each filter, and the accuracy of 6 LMs from real users. Configuration 1 from Section 4.1 is being applied.

model is the one made with configuration 1, Adam optimizer and *High* filter (C1 in Figure 3(a)), providing a final accuracy of 75% and 94% in GM and LM.

These accuracies are just an overview and do not show the full potential of the models, since they were obtained by randomly selecting 50% of the

Table 2 Detailed accuracies per model configuration, according to the different situations. LM accuracies are the average of all LMs in the federation

		UE		NE		NU	
		GM	LM	GM	LM	GM	LM
SGD	**C1**	0.72	0.84	0.82	0.46	0.86	–
	C4	0.85	0.96	0.94	0.41	0.87	–
ADAM	**C1**	0.88	0.96	0.95	0.57	0.95	–
	C4	0.80	0.96	0.95	0.42	0.93	–

samples for each dataset. Therefore, Table 2 presents the 3 possible situations of the case study and the behaviour of each model. First, UE situations are where the LM shines, showing how good it predicts user's behaviour with an accuracy of 96%. As expected, the GM cannot reach such levels of accuracy for a well known user, since it is made to generalize for the entire federation. Second, NE situations are where most of the competition between GM and LM takes place. Since the test was performed using complete new environments, the LM cannot give acceptable predictions at first, so the GM will be used until the LM is sufficiently trained and the NE becomes another UE. Finally, the last situation is NU, which is monopolized by the GM since new users do not have a LM yet. The outcome will be similar to NE, where the GM takes the lead in the first steps.

4.2 Phase 2: Real Data

4.2.1 Scenario

A case study has been proposed with situations similar to the example given in Section 2. A set of real users who regularly interact with IoT devices have been selected. Notable differences between them are work and leisure schedules (hours and weekdays), device type preferences, usage, environment changes and occasional patterns. They have been provided with an Android app to specify the device, performed action, and environment. This app only serves as an information-gathering tool and does not perform any real action with IoT devices.

The system has been implemented with TensorFlow[1] (TF). However, since TFLite only performs inference on Android, clients have been implemented with the DeepLearning4j library.[2] Apart from validating the previous obtained results, the aim is to study device computational and

[1] https://www.tensorflow.org/

[2] https://deeplearning4j.org/

battery consumption. This phase uses the best configuration from phase 1: Configuration 1, *Adam* optimizer and *High* filter.

4.2.2 Set-up

During data gathering, users had all possible options within the following limits: 4 environments, 5 different device types, 12 devices, 2 connection modes, and 10 actions in total. This phase lasted 3 weeks. In total, there are 6 users and 1,313 tuples. For the testing phase, 50% of the data obtained from each user were randomly taken and used to request predictions from the two models. The monitoring of mobile devices has been carried out with Android Profiler [7] and Battery Historian [8].

4.2.3 Results

The progress of both GM and the personalized LMs have been monitored. In Figure 3(b) we can see the differences between LMs: Some models are able to learn quickly the behavior of their users, reaching accuracy rates up to 94%. However, there are some models that do not learn properly. The main cause are users with no apparent behavioral patterns or who interact with IoT devices in an occasional and random way, cases already considered in the data collection phase and classified as outliers. As for the GM, performed tests show a general accuracy of 83% when the best filter is active (Figure 3(b)).

Like Section 4.1, accuracies are divided according to the different situations to be faced by models. Starting with UE, the GM has an accuracy of 85% while the LM has an accuracy of 94%, showing the control of the LM when it comes to already known situations. For NE situations, GM and LM show 83% and 73% accuracy, proving how difficult it is to provide a prediction in new situations where the LM has not yet acquired the necessary knowledge, and it is why the GM is needed in the meantime. Moreover, it is an example of the competition between them. Finally, in NU situations the GM keeps its accuracy up to 83%, as stated before.

Apart from the system's performance, we are particularly interested in the impact this learning process has on mobile devices. Data obtained are listed below. The Android Profiler report shows a CPU usage of 19% and a RAM usage of 130 MB on average. As for network usage, each update package is 25 KB in size, sent in 10 rounds (250 KB data sent and 250 KB data received), for a total of 0.52 MB of network traffic taking network headers into account, and bandwidth highest peak of 110 KB/s. The execution time has an average duration of 11 sec. Regarding the battery impact, we obtained results of 0.04% usage of device's battery, classified by Battery Historian as

a low-level cost. As for the inference cost (i.e. a prediction), we consider it insignificant as it is so small that it is not even recorded in the history.

4.3 Related Works Comparison

The proposed approach has been compared with two different related works. The testing methodology is a standard evaluation with synthetic data like in Section 4.1. This validation will not dive into the situation comparison.

First, the approach in [18] presents a centralized solution to study air quality. Since they detected different behaviors, the architecture is composed by a general model to provide generic predictions and two specific models, for summer and winter behaviours. A similar architecture has been recreated with slight changes to fit our case study. The generic model provides an accuracy of 78%, having slightly better performance than our option, due to the centralised training. For the specific models, the users have been split according to their work schedule, resulting in two models: one for users who work in the morning and one in the evening. The model for mornings provides an accuracy of 92% since most behaviours are similar, while the model for evenings provides only a 76%, since evening works are more heterogeneous. Both models are outperformed by the presented approach (FL-Dual) thanks to its personalized approach, which performs even better in specific situations.

Second, the approach in [27] focuses on human activity recognition and uses the standard architecture of FL with a deeper NN (3NN) and a convolutional NN (CNN), very different configurations compared with those tested in this paper. Results show the 3NN provides 68% and 85% in GM and LM, while the CNN approach provides 67% and 87%. Both approaches are far from the 73% and 96% obtained with our FL-Dual model. The differences lie in the use of a local model that is never independent of the global one, and in the use of NNs that do not fit with the presented domain and case study.

4.4 Validation Discussion

This section presents a brief analysis after synthetic and real data tests. In phase 1, synthetic data provides better results in general terms, specially in already known environments. In NE, LM's performance is low, as the model does not yet have sufficient knowledge of the new situation, hence the need for the GM. The performance will increase with subsequent actions in this environment, until it becomes a UE and the LM replaces the GM. Also, the GM provides an exceptional accuracy on NE and NU, which would be hard to maintain in real scenarios due to the wide range of users' preferences.

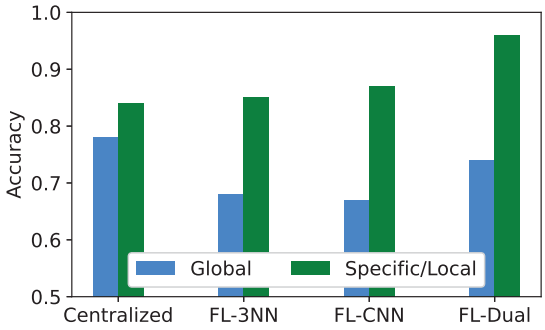


Figure 4 Accuracy comparison between the different approaches. LM acc. is the mean of the average of every model.

In phase 2 the models were tested with real data, and shows how the accuracies are softened to a more realistic degree. However, there is an exception to this rule, the LM in NE situations, where its accuracy increased to 73%. The reason for the increase in accuracy is that in real environments, users tend to use the same type of devices, as they feel more familiar with them.

Finally, there is a small percentage of cases where models do not give an accurate prediction, caused by situations where it is difficult to provide one. For example, loudspeakers, if a user listens to pop and rock music with the same frequency and at the same time.

5 Related Work

There are plenty of statistical heterogeneity works [13, 14, 29] who study the impact of No-IID data and propose new algorithms like *SCAFFOLD*, but they usually are not focused on achieving a personalized behavior. In [28], FL is used to improve the quality of GBoard word suggestions, making them closer to users' habits, and it is a constructive example of its applications in production environments. In [16], an intermediate approach between single global model and local models is proposed, by suggesting a *user clustering* to group similar users. *Transfer Learning* can also be used to learn some or all parameters in the global model and re-learn them in local models [25], but those models cannot be re-trained for too long to avoid knowledge forgetting. *FedPer* is proposed in [4] to mitigate statistical heterogeneity in federated scenarios. It presents a NN architecture where the base layers are trained centrally with *FedAvg* and the top (or personalized) layers are trained locally

and independently. However, this approach might not throw good results with small networks like the ones presented in this paper. Hanzely et al. [9] propose a formulation for a trade-off between the global model and the local ones. To solve it, the author develops a new variant of gradient descent called *Loopless Local Gradient Descent* under the claim that standard *FedAvg* might be too aggressive. In addition, FL techniques to mitigate device heterogeneity issues in IoT environments are discussed in [27]. In this work, a framework named *PerFit* is developed that, through a globally shared model, advocates for improving device integration while preserving data privacy.

Cook et al. [5] propose an architecture based on a hierarchy of rational agents that cooperate to meet the goals of the environment. Unfortunately, the architecture is designed to work on fixed environments. Also, Kabir et al. [12] proposes another architecture for different areas at home. This architecture provides services according to users' choices using machine learning (ML) techniques. However, there is no discussion of the possibility of detecting other environments automatically or how to handle situations that occur with different people. In addition, Think Home [22] is presented as a deployable solution on inexpensive hardware such as Arduino boards, whose main objectives are to minimize energy consumption and ensure user comfort. This is achieved by developing intelligent control strategies and representing the knowledge of the home through ontologies. However, this work does not address the possibility of automating behaviors or making predictions in changing environments. Nascimiento et al. [18] also proposed an architecture that uses a ML model to analyze a dataset or interact with an environment, and also monitors changes in the context. The comparison of a general model for all contexts with a context-specific system is particularly interesting. However, they are only based on analyzing one feature, so this system would have to be modified for contexts where more than one feature is involved.

6 Discussion and Conclusions

The increasing number of IoT devices and smart environments will make this paradigm difficult for users to manage due to the high number of interactions and configuration that they require. This will result in users spending too much time on devices' management and the final benefits of this paradigm will be reduced. Users need tools reducing the workload and the interaction level with their devices. In this paper, a dual model to predict interactions with IoT devices in different environments, based on users preferences, has been

proposed. This proposal makes use of contextual information and previous actions of federated users and has a low impact on mobile devices.

During the development of this work, the following key aspects have been identified: (1) Intelligent environments can be very different and heterogeneous, which makes developing a multipurpose and effective model for any context a challenging task. Besides, interoperability is crucial to allow easy management of any IoT device using a mobile device; (2) Context and properties of environments shape their fingerprint. Each environment is defined by the devices in them and how they are used, assisting the model in its task of environment identification; (3) The more contextual properties and information to be analyzed, the more appropriate the behavior will be, and more similar environments can be identified. A better understanding of context leads to smarter models and better predictions.

In future works, we plan to apply the proposed approach in social environments, where different users need to be taken into account and conflicts of interests need to be addressed.

Acknowledgement

This work was funded by the project RTI2018-094591-B-I00 and the FPU17/02251 grant (MCI /AEI/FEDER,UE), the 4IE+ Project (0499-4IE-PLUS-4-E) funded by the Interreg V-A España-Portugal (POCTEP) 2014-2020 program, by the Department of Economy, Science and Digital Agenda of the Government of Extremadura (GR18112, IB18030), and by the European Regional Development Fund.

References

[1] Information – The ONE. https://akeranen.github.io/the-one/.

[2] Musaab Alaa, A. Zaidan, Bilal Bahaa, Mohammed Talal, and Miss Laiha Mat Kiah. A review of smart home applications based on Internet of Things. In *JNCA*, nov 2017.

[3] H. Arasteh, V. Hosseinnezhad, V. Loia, A. Tommasetti, O. Troisi, M. Shafie-khah, and P. Siano. Iot-based smart cities: A survey. In *IEEE-EEEIC'16*, pages 1–6, 2016.

[4] Manoj Ghuhan Arivazhagan, V. Aggarwal, A. Singh, and S. Choudhary. Federated learning with personalization layers. *ArXiv*, abs/1912.00818, 2019.

[5] Diane J. Cook, Michael Youngblood, and Sajal K. Das. *A Multi-agent Approach to Controlling a Smart Environment*, pages 165–182. Springer Berlin Heidelberg, 2006.
[6] Y. Deng, M. M. Kamani, and M. Mahdavi. Adaptive personalized federated learning. *ArXiv*, 2020.
[7] Google Inc. Measure app performance with Android Profiler. https://developer.android.com/studio/profile/android-profiler, oct 2020.
[8] Google Inc. Profile battery usage with Batterystats and Battery Historian. https://developer.android.com/topic/performance/power/setup-battery-historian, jan 2021.
[9] Filip Hanzely and Peter Richtárik. Federated learning of a mixture of global and local models. *ArXiv*, abs/2002.05516, 2020.
[10] Juan Luis Herrera, Paolo Bellavista, Luca Foschini, Jaime Galán-Jiménez, Juan M Murillo, and Javier Berrocal. Meeting stringent qos requirements in iiot-based scenarios. In *IEEE GLOBECOM 2020*, pages 1–6. IEEE, 2020.
[11] Kevin Hsieh, Amar Phanishayee, Onur Mutlu, and Phillip B. Gibbons. The non-iid data quagmire of decentralized machine learning. In *ICML*, 2020.
[12] M. Kabir, M. R. Hoque, and Sung-Hyun Yang. Development of a smart home context-aware application: A machine learning based approach. *IJSH*, 9:217–226, 2015.
[13] S. P. Karimireddy, S. Kale, M. Mohri, Sashank J. Reddi, Sebastian U. Stich, and A. T. Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *ICML*, 2020.
[14] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the convergence of fedavg on non-iid data. In *ICLR*, 2020.
[15] Knud Lasse Lueth. State of the IoT 2020: 12 billion IoT connections. https://iot-analytics.com/state-of-the-iot-2020-12-billion-iot-connections-surpassing-non-iot-for-the-first-time/, nov 2020.
[16] Y. Mansour, M. Mohri, Jae Ro, and A. T. Suresh. Three approaches for personalization with applications to federated learning. *ArXiv*, abs/2002.10619, 2020.
[17] H. McMahan, E. Moore, D. Ramage, and B. Agüera y Arcas. Federated learning of deep networks using model averaging. *ArXiv*, abs/1602.05629, 2016.
[18] N. Nascimento, P. Alencar, C. Lucena, and D. Cowan. A context-aware machine learning-based approach. In *CASCON*, 2018.

[19] D. C. Nguyen, M. Ding, P. N. Pathirana, A. Seneviratne, J. Li, and H. V. Poor. Federated learning for internet of things: A comprehensive survey. *IEEE COMST*, pages 1–1, 2021.

[20] Nitika Nigam, Tanima Dutta, and Hari Gupta. Impact of noisy labels in learning techniques: A survey. In *ICDIS 2019 LNNS*, pages 403–411. Springer Singapore, 2020.

[21] S. O'Dea. Smartphone users 2020. https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/, dec 2020.

[22] C. Reinisch, M. J. Kofler, and W. Kastner. Thinkhome: A smart home as digital ecosystem. In *IEEE-DEST 2010*, pages 256–261, 2010.

[23] Padraig Scully. Top 10 IoT applications in 2020. https://iot-analytics.com/top-10-iot-applications-in-2020/, jul 2020.

[24] R. Shokri and V. Shmatikov. Privacy-preserving deep learning. In *Allerton (2015)*, pages 909–910, 2015.

[25] K. Wang, R. Mathews, C. Kiddon, H. Eichner, F. Beaufays, and D. Ramage. Federated evaluation of on-device personalization. *ArXiv*, abs/1910.10252, 2019.

[26] Kai Wehmeyer. Assessing users' attachment to their mobile devices. In *ICMB 2007*, pages 16–16, 08 2007.

[27] Qiong Wu, Kaiwen He, and Xu Chen. Personalized federated learning for intelligent iot applications: A cloud-edge based framework. *IEEE OJ-CS*, 1:35–44, 2020.

[28] 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*, 2018.

[29] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. *arXiv*, 2018.

[30] Allen Zhu. Learning From Non-IID data. https://xzhu0027.gitbook.io/blog/ml-system/sys-ml-index/learning-from-non-iid-data, 2020.

Biographies

Rubén Rentero-Trejo. Received the Bachelor's degree in software engineering from the University of Extremadura in 2019, and the MSc in Computer Science Engineering in 2021. His main research interests include IoT, mobile computing and Deep Learning.

Daniel Flores-Martín. Is a Ph.D student at the University of Extremadura (Spain). His research interests are mobile computing, context-awareness, pervasive systems, crowd sensing and Internet of Things.

Jaime Galán-Jiménez. Received the Ph.D. in computer science and communications from the University of Extremadura in 2014. His main research interests are Software-Defined Networks, 5G network planning and design, and mobile ad-hoc networks.

José García-Alonso (IEEE Member) is an Associate Professor at the University of Extremadura. His research interests include software engineering, mobile computing, pervasive computing, eHealth, gerontechnology.

Juan Manuel Murillo (IEEE Member) is a full professor at the University of Extremadura. His research interests include software architectures, mobile computing, and cloud computing.

Javier Berrocal (IEEE Member) is an Associate Professor at the University of Extremadura. His main research interests are software architectures, mobile computing and edge and fog computing.