Sharing Knowledge to Promote Proactive Multi-environments in the WoT[*]

Daniel Flores-Martin[1,†], Rubén Rentero-Trejo[2], Jaime Galán-Jiménez[1], José García-Alonso[1], Javier Berrocal[1] and Juan Manuel Murillo[3]

[1]*University of Extremadura, Cáceres, Spain*
[2]*Global Process and Product Improvement S.L., Cáceres, Spain*
[3]*Computing and Advanced Technologies Foundation of Extremadura (COMPUTAEX), Cáceres, Spain*
E-mail: dfloresm@unex.es; rrenterot@unex.es; jaime@unex.es; jgaralo@unex.es; jberolm@unex.es; juanmamu@unex.es
[†]*Corresponding Author*

Received 30 November 2022; Accepted 30 December 2022;
Publication 16 June 2023

Abstract

The main goal of the Web of Things (WoT) is to improve people's quality of life by automating tasks and simplifying human–device interactions with ubiquitous systems. However, the management of devices still has to be done manually, which wastes a lot of time as their number increases. Thus, the expected benefits are not achieved. This management overhead is even greater when users change environments, new devices are added, or existing devices are modified. All this requires time-consuming customization of configurations and interactions. To facilitate this, learning systems help manage automation tasks. However, these require extensive learning times to achieve customization and cannot manage multiple environments so new approaches are needed to manage multiple environments dynamically. This

[*]R. Rentero-Trejo, J. García-Alonso, J. Galán-Jiménez, J. Berrocal, J. M. Murillo, Knowledge sharing in proactive WoT multi-environment models. To appear in the *Proceedings of the 2nd International Workshop on Big Data Driven Edge Cloud Services (BECS 2022), Jul 5, 2022.*

Journal of Web Engineering, Vol. 22_2, 327–356.
doi: 10.13052/jwe1540-9589.2226

work focuses on knowledge distillation and teacher–student relationships to transfer knowledge between IoT environments in a model-agnostic manner, allowing users to share their knowledge each time they encounter a new environment. This work allowed us to eliminate training times and achieve an average accuracy of 94.70%, making model automation effective from the acquisition in proactive WoT multi-environments.

Keywords: Web of Things, knowledge distillation, mobile devices, context-aware.

1 Introduction

According to the State of IoT 2021 report [27], there are more than 12 billion connected Internet of Things (IoT) devices nowadays. Equipped with sensors, computing, and communication capabilities, these devices can be found everywhere working together to make our daily lives easier. Currently, controlling IoT devices is a simple task, as they are designed to execute simple and specific commands. However, in the near future, and due to the growth of this field, these devices will be found in homes, workplaces, vehicles, and so on, until everything becomes an IoT environment. People will move through these environments, interacting with their devices while changing their context, requirements, and behavior. This scenario demands IoT environments to proactively react to users' needs anytime, anywhere. Managing so many devices to make them personalized to users will become more complicated and time-consuming than ever, especially if the user needs to manually interact with every device whenever the context or their needs change, or whenever they arrive in an environment [30]. The situation becomes more complicated when users change environments, as they have no previous experience and customization cannot yet be established. Therefore, these additional efforts will allow users to perceive the benefits of IoT as insufficient, as the workload would not be reduced as much as expected.

There is a need for approaches that learn from users' behavior and the contextual information around them, automating routines and thus avoiding time wasted on trivial manual actions which follow predictable patterns. Systems that provide greater control over other devices already exist on the market, such as the Amazon Echo, Google Nest, or Apple HomePod voice assistants. However, these assistants do not support automated learning and serve as yet another programmable actuator [4] (e.g. voice assistants can be configured to automate a task, but cannot learn and assimilate changes by themselves). Some learning systems take advantage of the large amounts of data generated by IoT devices to learn behaviors and provide the desired automation. However, they often require long training times or periods of adaptation and are focused on a single environment [12, 23]. For example, learning systems designed for a particular smart-home or room are not usable "as is", in any other scenario. Given this downside, out-of-the-box automation and personalization approaches are needed, allowing devices to be used efficiently from the early stages, avoiding manual actions or configurations. At the same time, the knowledge learned must be reusable, providing the flexibility to face new situations and to always experience personalization, even if the environment, conditions, or preferences change.

This paper presents an approach to sharing knowledge among the participants of a federation. Thanks to this federation, the participants are able of leveraging individually generated knowledge to generate collective benefit. Also, it establishes as the main requirement that knowledge acquired by learning models can be reused by other users. Knowledge sharing focuses on enabling participants to leverage the knowledge acquired by other participants (i.e. past experiences from similar users in similar environments) and use them in their particular environment, therefore avoiding the need to learn everything from scratch. Knowledge distillation [10] techniques are used to extract specific portions of knowledge from complex models and transfer it to new, simpler models. This proposal allows models to effectively share their knowledge, achieving an overall accuracy of 94.70% with brief distillation while often improving the source model performance. To this extent, users can refine their knowledge and share it with others in the context of proactive IoT multi-environments. Thus, reducing the need for interaction with IoT devices.

The paper is structured as follows. Section 2 presents the background and motivations of this work. Section 3 presents our proposal to improve knowledge exchanges between IoT multi-environment models, and Section 4 presents a common scenario in IoT, the set-up and results. Section 6 presents related works in the field. Finally, discussion and concluding remarks are given in Section 8.

2 Motivation

Users demand automation and proactivity from IoT environments to fulfill their needs and preferences while models try to avoid the most trivial manual actions by knowing their usual behavioral patterns. However, users do not stay in one single place but move through different environments. The visited environments must be able to react to users, based on their preferences and actions performed in the past. Static nodes are not suitable for this task due to their lack of mobility. Therefore, mobile edge nodes are proposed as learning nodes. They serve as companion devices since they know their users better than anyone, can orchestrate the behavior of different environments as the main points of interaction, and their computing capabilities fit the automated learning necessities, reducing the dependency on the cloud. Related to this issue, different authors have proposed approaches like mobile-centric models to provide users greater control over their data [8], or an architecture to improve the integration of people within the IoT ecosystem [19]. To this end, mobile devices are proposed as mobile edge nodes to rule over the generated data. Their main role is to act as users' representatives and to be the main learning and decision-making nodes, interacting with IoT devices on their behalf.

Since the main goal is personalization, dealing with multiple IoT devices, the way users operate them, and the conditions around them, give shape to an IoT environment. Learning models need to learn from the user's previous actions in other environments. The use of a common point, such as a mobile device, simplifies this task. However, the first problem to overcome is the lack of data. New users and new environments (i.e. mostly those consisting of devices the user has not previously used) experience a lack of previous data and models can not be trained. As a consequence, users need to generate data based on devices' usage, causing models to be inactive in their early stages and penalizing users' experience due to these adaptation periods [36].

To overcome this limitation, the knowledge of existing users needs to be transferred to other users and leveraged to avoid the initial training phases. Existing models can be exploited in situations where users do not have previous data, or when facing an environment that is different from their usual experience. However, the second problem arises when trying to select specific portions of knowledge. Performing an effective knowledge transfer in situations with ever-changing conditions like IoT environments and ever-evolving models is a challenging task due to the models' heterogeneity. For example, suppose a model that manages four different environments with many devices each. Now, if a user wants the functionality of only one environment, relationships between the subset of input features and the subset of output labels of interest cannot be decoupled from the rest of the network. Due to the nature of neural networks (NNs), layers can be deleted or added [11] and their neuron weights can be modified or transferred [18,25], but extracting specific end-to-end relationships is still a challenge (i.e. it cannot be known which specific set of neurons among the whole network produces a specific output).

To solve both issues, the concept of knowledge reusability needs to be extended to every mobile device. By reusing the knowledge assimilated by existent models, users will take advantage of the actions performed by previous users in situations where data is non-existent or insufficient.

If the behaviors of *similar users* who have previously visited a *similar environment* are selected, a model capable of managing new environments can be generated from the properties of the environment. The knowledge distillation [6] is the main driver of this approach. It allows distillation, extraction, and sharing of knowledge (i.e. specific fragments). In exchange for a small training phase, a teacher model trains a student model. This process allows the extraction of end-to-end relationships by looking at model outcomes from a model-agnostic perspective while maintaining the original performance. The model-agnostic perspective refers to the fact that the internal logic of the model does not need to be accessed and is based only on the input instances and output results [35]. Complementary information from other models is useful to generate similar and preliminary environments for the user. Thus, environments based on similar characteristics can be generated and can be useful for decision-making in the user's real environments, until, over time, the user's model acquires information about his or her habitual behavior. Finally, the student model acquires the processed knowledge and will mimic its teacher. Although this technique has promising potential, it is usually used for model reduction, which converts a complex model into a simpler, faster, and cheaper one, that does the same tasks. However, this process can be modified to make students mimic only specific tasks (i.e. each specific user needs).

Recovering the previous example, this approach allows newcomers to ask for a specific part of knowledge from a similar user. In this way, users acquire a trained and ready-to-use model. Even if some further refinement is needed to reach full personalization, the model will be ready from the early stages without heavy training phases while providing acceptable performance. This perspective avoids traditional learning necessities where time was needed to generate data to train on later, especially for models that need to achieve personalization.

This paper places knowledge distillation in the IoT context and extends its traditional use to allow the selection of knowledge fragments from complex models and create new, smaller and task-specific models. First, knowledge distillation techniques are applied to reduce the complexity of a larger and intensively trained model and distill it into a simpler one. Second, available data and the training algorithm are modified, so input features and output labels are reduced to fit the new model and user needs. This avoids providing an overly complex model to a user who is only interested in some specific functionalities, such as the behavior of a single environment or a subset of devices, and avoids long waits for users when there is a deficit of data.

3 Knowledge Distillation in WoT Environments

This paper applies knowledge distillation (KD) to complex and heavily trained models to extract specific fragments of knowledge and transfer them to new and simpler models which fit other users' needs. The goal is to provide users with a personalized and ready-to-use model whenever they discover a new environment. The main components are as follows. (1) A complex, trained model that takes the role of the teacher. (2) A reduced, simpler model that takes the role of the student. (3) A distiller that implements the training algorithm and manages the distillation from teacher to student. Finally, by *complexity* we mean both the configuration and structure of the internal layers of an NN, as well as the number of input features and output labels.

Although distillation involves much less training compared to training a model from scratch, a small amount of data is still required. However, this data can be provided by both the teacher (e.g. a small history of recently performed actions) or the student (e.g. generated data from known preferences or usage data from the actual environment). They are needed to identify the internal relationships from the teacher inputs to the outputs. For privacy reasons, the distiller process takes place on the student's mobile device using its data, avoiding the risk of data exposure. As the distiller needs both models, the teacher model is sent to the student (either compiled or as a list of neuron weights) to complete the process. Once the creation of the student model is done, the teacher model is discarded from the student's device. This also implies the student carries the distiller computation, as it will be the main beneficiary at the end of the process.

The process starts with a dataset of information (i.e. tuples of device usage). These tuples collect the information associated with the environments and the states associated with them. To generate heterogeneous environments, different elements have been considered such as the technology used, location, devices of different types, and time. The generation of environments has been performed for a different number of users. For each user, N

Table 1 Characteristics considered for the environments in the datasets

Characteristic	Variable	Possibilities
Communication technology	BLE	AA1, ALT1, ALT2
	WIFI	
Location	Car	E1, E2, E3
	Home	
	Home parents	
Smart devices	Plug	TV1, TV2, TV3, 18C, 20C, 70W, Antena 3, FDF, Telecinco, ON, OFF, Pop, Rock
	Air conditioner	
	Speaker	
	TV	
	Bulb	
Time	01:00	0, 1
	02:00	
	...	
	23:00	
	00:00	

environments are generated where each of them has different states for the elements considered. In this way, the simulation is enriched with heterogeneous environments to consider a wide range of possibilities when training the model and to suggest possibilities to new users. Table 1 details the elements examined in the datasets, which are specified as follows:

- **Characteristic:** aspects considered in the environment. The technology used, the location where the user is located, the devices that can be used, and the time at which they are used are considered.
- **Variable:** each element can take a different value. For example, WIFI for the technology, Home for the location, Bulb for the smart device, and 17:00 for the time when the environment is stored.
- **Possibilities:** each variable can be associated with different devices and take different values. For example, different WIFI points such as ALT1 and ALT2, the environment related to Home, E1, or setting the Bulb to ON.

After the pre-processing steps, two different datasets are obtained as input data (Figure 1]. Although representing the same information, they are processed differently to feed the two models since the feature space and the label space from both models are different (e.g. if a teacher manages three environments but the student is interested in one, as Figure 2 shows, the student's dataset will not have any data related to other environments or

Table 2 Environments example for a user

Environment	BLE	WIFI	Car	Home	Home Parents	Office	Plug	A/C	SPK	TV	Bulb	1:00	…	23:00	00:00
30	0	1	0	0	1	0	0	0	0	1	0	0.0	…	0.000000	0.0
0	1	0	0	0	0	1	3	0	0	0	0	0.0	…	0.000000	0.0
22	1	0	0	1	0	0	0	1	0	1	1	0.0	…	0.651613	0.0
31	1	0	0	0	0	1	1	1	0	0	1	0.0	…	0.000000	0.0
18	1	0	1	0	0	0	0	1	1	0	0	0.0	…	0.000000	0.0
28	1	0	0	1	0	0	0	1	0	1	1	0.0	…	0.761062	0.0
…															

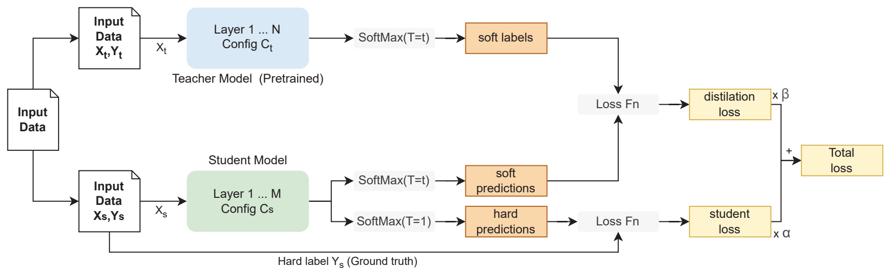


Figure 1 Distiller workflow. Input data is processed into two datasets to fit the networks' input and output. Note that Y_t is never used. Configurations C_t and C_s can be different to make the distiller model agnostic and allow model heterogeneity.

devices, modifying the feature space and the target output labels). For each user, a dataset is obtained from the aspects examined above. An example is given in Table 2. This example contains the different environments generated (*Env*), as well as the factors that have been produced for these environments (*BLE, WIFI, Car, 13:00, Rock,…*). These factors are identified as 1 if they have occurred, and as 0 if they have not occurred yet. In this way, we can identify how the different environments are generated for all users when training the model. More details about inputs are given in Section 4.3.

The distiller coordinates the training. It performs the operations with the different functions and optimizers to update the weights of the neurons appropriately. The following process is carried out in each training step. First, the input data X_t is used to perform a forward pass on the trained teacher to get the soft labels from a *SoftMax* function and temperature (T). The purpose of the temperature parameter is to soften the logits and smooth the probability distribution to reveal the internal relationships learned from the teacher. The same process is performed with the input data X_s on the student to get soft predictions. To get the *distillation loss* (DL), Kullback–Leibler divergence has been applied to minimize both teacher and student probabilistic outputs, Equation (1).

A parallel process takes place to calculate the student loss (SL), being the difference between student predictions and ground-truth using the *softmax* function where $T = 1$ to get the original output, and the *categorical cross entropy* (CCE) loss function, Equation (2), as it is a classification problem. Finally, the total loss, Equation (3), is obtained by weighting the two different losses (SL, DL) with an α value that ranges between 0 and 1. This total loss is the one used to get the gradients and update the trainable weights of the student NN with the optimizer.

$$DL = KLDivergence(Softmax(pred_t/T), Softmax(pred_s/T)) \tag{1}$$

$$SL = CCE(Y_s, pred_s) \tag{2}$$

$$TotalLoss = \alpha * SL + (1 - \alpha) * DL \tag{3}$$

In our study, the most common situation involves models with different feature spaces and label spaces. Thus, some modifications need to be made to be able to operate with the different probabilistic outputs. The main restriction is operations that involve logits from different models, like DL, Equation (1). To make these operations viable, logits that are of no interest to the student need to be discarded to fit teacher and student shapes (e.g. if a student is interested in a specific device like smart TV, data from other devices is meaningless). This way, specific labels can be selected and transferred from one model to another.

In short, this process allows cumbersome and heavily trained models to share a specific part of their knowledge with third parties. It allows knowledge reusability and avoids intense training periods for newcomers and users when visiting new IoT environments since they acquire immediate and adequate device personalization from the early stages of the model.

4 Validation

The validation of this work is structured as follows. Section 4.1 introduces the studied scenario and the source of our data. Section 4.2 describes the implementation. Section 4.3 deals with the algorithm's technical details and model configurations. Finally, Section 4.4 presents our results and analyzes the proposal performance.

4.1 Scenario

A case study has been proposed with situations similar to the one illustrated in Figure 2: Suppose a user (U1) that has many IoT devices (e.g. light bulbs, air conditioners, speakers, smart TV) in different IoT environments (e.g. home, office, car). This user can automate them with a learning model installed on his mobile device. Somewhere, another user (U2) enters an environment with the same settings that one specific environment from U1 while having similar preferences. Instead of training from scratch, U2 asks for U1's model, as it is already trained. As U2 is interested in just one environment, the mobile device extracts the knowledge of interest to then automate the newly discovered environment.

To measure the performance of our approach many datasets are needed. To do so, semi-synthetic data has been prepared following a specific process. First, real data was obtained from a set of real users who regularly interact with IoT devices. Notable differences between them were work and leisure schedules (hours and weekdays), device type preferences, usage, environment changes, and occasional patterns. For 3 weeks, they were provided with an Android app to specify the device, performed action, and environment. This first phase with real data gave us an accurate overview of how users interact with IoT devices and how their daily circumstances affect their behavior in environments. However, the obtained data were not large enough to train a sufficient amount of learning models. Therefore, this real data was extended to get substantial data for training and testing purposes. TheONE Simulator [13] has been used to simulate human 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. Some examples are daily home-to-work trips and vice versa and occasional trips for leisure, visits, or shopping purposes. By modifying its core, this tool allowed us to simulate interactions

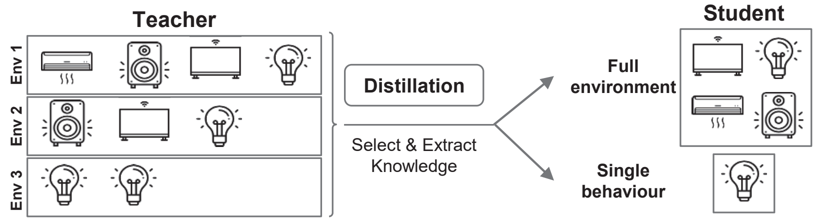


Figure 2 Situation example where a teacher manages multiple environments (monolithic model) and a student that decides to learn a full environment or just the behavior of a single device.

with IoT devices in different environments. For example, human-device interactions like turning on lights in the morning or at night, setting up alarms, setting up the desired temperature, and choosing specific channels and radio stations on TV and speakers. For the sake of completion, 165 user datasets have been generated, being No-IID and unbalanced data, up to a maximum of 1.500 tuples each.

4.2 Implementation

Next, the implementation performed is described for the use case. Firstly, the tasks performed are indicated:

1. **Encapsulate the recommender in a module:** The recommender must be located on some kind of cloud server because it needs to have information from all the participants in the federation and act as the main database. The main task of the recommender is to receive a search request in the form of an environment definition, and use it as a basis to generate a list of environments similar to that one, with the top one being the environment that most resembles it. Subsequently, from that list, the selected environment will become a teacher, which will train the environment we have used as input and which will be the student.
2. **Dataset selector:** This is used to locate the user to which the environment (or environments) obtained from the distiller belongs. Knowing the definition of the environment, the user to which it belongs is searched. Then, the action dataset of this user is searched. This dataset of actions will then be used to train the teacher.
3. **Teacher trainer:** This takes the teacher's dataset and trains the teacher's model. In a real situation and for a new user of the federation, a veteran user is asked to share his model, because this model is already trained and works correctly.
4. **Encapsulate the distiller in a module:** The distiller is intended to be placed on the cell phones of all users (especially new ones) and to serve as a service that takes a teacher model (copy of the original), its dataset (copy of the original) and the environment to be learned, and extracts the knowledge to obtain a trained student model. Distillation is in charge of training a student model using valid answers (also called "ground truth"), the answers that the teacher model would give for the same situations. A fundamental aspect is that this distillation only takes into account the environment that the student wants to learn, ignoring all other environments that the teacher may know. This means that, from

the teacher's dataset, all the rows that do not belong to the environment to be taught are eliminated, as well as all the columns that have nothing to do with the environment to be taught. For example, if the student does not have any television in the environment he wants to learn, all columns containing televisions or actions of televisions are ignored.

Secondly, we show the procedure followed. Before starting to distill, it is necessary to clarify that to train the models we need a dataset (input data). Since it will be the teacher who teaches the student, the data to be used are those of the teacher. Logically, this should be the case since the teacher is the only one who has experience with the environment that the student wants to learn. The idea is an action history dataset or something similar. This input data dataset, apart from the curing, cleaning, and pre-processing that is done with any dataset, is used to generate two datasets:

- First, the dataset for the teacher. In the distillation, the teacher will only be asked questions (inference) about the environment the student wants to learn about; therefore, all rows of actions that are not within that environment are removed. This provides us with an input data dataset X_t, Y_t, where:
  - X_t: a set of characteristics of the teacher dataset.
  - Y_t: a set of labels of the teacher dataset.
- Second, the dataset for the student. In the distillation, the student must learn the environment he has chosen; for this, its training dataset must contain all the rows of actions of the environment to be learned, as well as all the columns of features needed for that environment (they will be, at most, the same as the teacher's). This leaves us with an input data dataset X_s, Y_s, where:
  - X_s: a set of characteristics of the student dataset.
  - Y_s: a set of labels of the student dataset.

Next, each dataset is provided to its respective model:

- Teacher: It will have N layers and a configuration C_t. For each tuple, it infers its prediction (soft labels: probability, values from 0 to 1).
- Student: Will have M layers and C_s configuration, both can be the same or different from the teacher's since the distillate can handle other models. For each tuple, it trains his model, obtaining:
  - Soft predictions: predictions with probability values from 0 to 1 according to the student.
  - Hard predictions: predictions with boolean values 0 or 1.

With the predictions already made, the loss function is run to determine how much the student model still has to learn (or how far it is from the ideal situation). This calculation is done by weighting two different losses:

1. Difference between the teacher's prediction (soft labels) and the student's prediction, giving rise to lose in the distillation process (distillation loss).
2. Difference between the student's prediction (hard predictions) and the absolute truth Y_s (ground truth) that appeared as labels in the original dataset. This results in student loss.

The total loss is a weighting of the two previous losses. This loss will be the one used internally by the model at each iteration to make the necessary corrections to the neuron weights. In addition, we should note that the distillation loss represents the distance between the student's and the teacher's model. Also, the student loss represents the distance between the student model and the actual predictions found in the original dataset. Therefore, the main idea is to keep the weight that favors learning through the teacher high, to take advantage of the knowledge already acquired.

And thirdly, part of the implementation conducted to perform the tasks specified through the procedure described above is specified. The first step is to load the data. For this purpose, the previously pre-processed data and the different environments that can be produced (Listing 1) are loaded:

```
[...]
# Loading data
defs_pre = pd.read_csv('data/definitions_dataset/preprocessed.csv')
defs_raw = pd.read_csv('data/definitions_dataset/
    environment_definitions.csv')
[...]
```

Listing 1 Data loading.

The next step is the creation of the recommender and its training to provide the model that most interests the users according to their characteristics (Listing 2):

```
[...]
# X = Original data
rec_x_vals = defs_pre.copy()

# Y = Tuples index
rec_data = defs_pre.reset_index()
rec_y_vals = rec_data['index'].copy()

# Train / Test split
rec_x_train, rec_x_test, rec_y_train, rec_y_test = train_test_split(
    rec_x_vals, rec_y_vals, test_size=0.25, random_state=42)
```

```python

# Creating and training the recommender
rec_model = NearestNeighbors(n_neighbors=2, algorithm='ball_tree').
    fit(rec_x_train)
[...]
```

Listing 2 Recommender training.

Next, one of the tuples belonging to the user is selected in order to find the best possible trainer, and in this way, the training is simulated for a new environment (Listing 3):

```python
[...]
# Selecting data from dataset
new_envs = rec_x_test[seleccion-1:seleccion]

# Retrieving user environment
user_id, env_id, env_name, user_dataset = retrieve_definition_origin(
    new_envs, defs_raw)
[...]

# Search for a teacher
k=5 # Neighbors
distances, indices = rec_model.kneighbors(new_envs, k,
    return_distance=True)

# Get DF index
similar_envs_indexes = rec_x_train.index[indices[0]]

# Get tuples having that index in the df
similar_envs = rec_x_train[rec_x_train.index.isin(rec_x_train.index[
    indices[0]])]
[...]
```

Listing 3 Selecting user environment.

In this step the models for the teacher and the student are created (Listing 4):

```python
[...]
# Create the teacher
# Config 4 (3 capas + 108 Nodos + BN + DP)
teacher = Sequential()
teacher.add( Dense(54, input_shape=(n_features,)) )
teacher.add( Activation("relu") )
teacher.add( BatchNormalization() )
teacher.add( Dropout(0.25) )

[...]

# Instantiate an optimizer and loss function
teacher_optimizer = keras.optimizers.Adam(learning_rate=0.01)
teacher_loss_fn = keras.losses.CategoricalCrossentropy(from_logits=
    True)
```

```

# Create the student
student = Sequential()
student.add( Dense(54, input_shape=(n_features ,)) )
[...]
```

Listing 4 Defining the models for the teacher and the student.

Once the models are defined, they can be prepared for knowledge distillation. This process is shown in Listing 5. Note that, for this experiment, after several tests, the hyperparameters alpha = 0.1, temperature = 3, and 10 epochs have been selected to train the network. Part of the training procedure is also shown.

```
[...]
# Distillation Hyper—parameters
alpha = 0.1
temperature = 3
epochs = 10

#Train loop
for epoch in range(epochs):
    print("\nStart of epoch %d" % (epoch ,))

    # Iterate over the batches of both datasets at the same time
    for step, ((x_batch_train_t, y_batch_train_t),(x_batch_train_s,
    y_batch_train_s)) in enumerate(zip(student_train_dataset,
    student_train_dataset)):

        # Forward pass of teacher
        teacher_predictions = teacher(x_batch_train_t, training=False
    )

        with tf.GradientTape() as tape:
            # Forward pass of student
            student_predictions = student(x_batch_train_s, training=
    True)

            # Compute losses
            student_loss = student.loss(y_batch_train_s,
    student_predictions)
            distillation_loss = distillation_loss_fn(
                tf.nn.softmax(teacher_predictions/temperature, axis
    =1),
                tf.nn.softmax(student_predictions/temperature, axis
    =1),
            )
            loss = alpha * student_loss + (1 — alpha) *
    distillation_loss
        # Compute gradients
        trainable_vars = student.trainable_variables
        gradients = tape.gradient(loss, trainable_vars)

        # Update weights
        student.optimizer.apply_gradients(zip(gradients,
    trainable_vars))
[..]
```

Listing 5 Training the model to select the appropriated knowledge.

From this we obtain the loss produced and the model reduction achieved. In addition, it has also been tested to train the trainee from scratch. This is shown in Listing 6.

```
[...]
student_scratch.compile(
    optimizer=keras.optimizers.Adam(learning_rate=0.01),
    loss=keras.losses.CategoricalCrossentropy(from_logits=True),
    metrics=[keras.metrics.CategoricalAccuracy()],
)

# Train and evaluate student-scratch on data.
hist_student_scratch = student_scratch.fit(student_x_train,
    student_y_train, epochs=epochs)
[...]
```

Listing 6 Training the student from scratch.

Using the same hyperparameters, the accuracy of the student model varies from 56.76% from epoch 2 to 97.84% at epoch 10. This is a relevant accuracy gain for the model used.

Finally, in the last step, we evaluate the accuracy of the generated models (Listing 7).

```
[...]
# Teacher accuracy
eva_t = teacher.evaluate(teacher_x_test, teacher_y_test)

# Distilled student accuracy
eva_s = student.evaluate(student_x_test, student_y_test)

# Student from scratch accuracy
eva_ss = student_scratch.evaluate(student_x_test, student_y_test)
[...]
```

Listing 7 Accuracy obtained after the process.

The results obtained are shown in Table 3. It is observed how the teacher model obtains an accuracy of 96.20%, the distilled student 94.74%, and the student from scratch 84.21%. Thus, a gain of 10.53% is obtained when distilled knowledge is used.

Table 3 Final results after training and select the teacher and the student

	Loss	Accuracy	Accuracy (%)
Teacher	0.0919	0.9620	96.20%
Student distilled	0.5275	0.9474	94.74%
Student from scratch	0.6457	0.8421	84.21%

4.3 Set-up

After exhaustive data processing, each dataset has a maximum of 64 columns, generated from the source data by *One-Hot Encoding*, 54 of which will be taken as input features for NNs. Since we are dealing with a classification problem, the output data are 10 at the most. To set up students' datasets (X_s, Y_s), the unused columns and tuples are removed and only the environment of interest for the student remains in the dataset. Teachers' configurations vary from just a couple of layers to cumbersome models according to the user needs and their development over time.

For starters (i.e. students), in a previous work [24] we identified that the best performance relies on simple models with 2 hidden layers with 54 nodes each and an ADAM optimizer, which are capable of managing single environments of any kind. In summary, to find the best configuration for our models, different tests were performed on a high heterogeneity federation. This heterogeneity is generated thanks to the multiple elements that are considered when simulating the environments and explained above, such as the number and type of devices, number of users, spatio-temporal properties, etc.

The tests consisted of an exhaustive comparison of different NN definitions, changing the depth, complexity, number of layers, nodes, and use of techniques like Batch Normalization or Dropout. The four most relevant configurations were selected and studied in detail before choosing the best one. Additionally, to improve the models' quality, a custom filter was designed for teachers, limiting their ability to teach on low-performance stages. Finally, after the distillation is complete, the students were tested with new data unrelated to teachers and generated in the users' particular environment to check the performance of the reused knowledge under the conditions of new users. For completeness, 50 teachers were selected, which taught one of their environments to 50 students in a 1:1 ratio.

4.4 Results

The teacher models have been evaluated for their ability to transfer knowledge to other models. First, models that were still in the early stages of their development (i.e. less than 85% accuracy on their environments) were discarded by the quality filter and are therefore not eligible for teaching. Second, the models selected as teachers achieved an average accuracy of 94.04%, and therefore well placed to share their knowledge effectively. Figure 3 shows the fast progress of models when teaching. Also, it is important to note that brief training is needed to achieve good learning. Although it is represented in steps, these data have been obtained in roughly five epochs.

New students, taught from scratch by their respective teachers, obtained an average accuracy of 94.70%. This demonstrates that, overall, all of the resulting models were able not only to maintain the accuracy of their teachers but also to improve it, as seen in Figure 4. However, since both environments and users are heterogeneous, there are models that improve the accuracy of their teachers by more than 6%, while some cannot keep up with the teacher and their accuracy worsens by a similar amount in the worst case. Although this last case has rarely occurred in the study, the worst student had a performance of 88%, always comfortably meeting the minimum requirements of the established quality filter. It is an expected outcome, as not every user will ever perfectly match the needs and preferences of another user. At that point, the brief personalization period to fine-tune the model in its entirety would start from this solid base.

The results obtained are quite promising from the point of view of the training time required and the accuracy achieved by the model. Given these

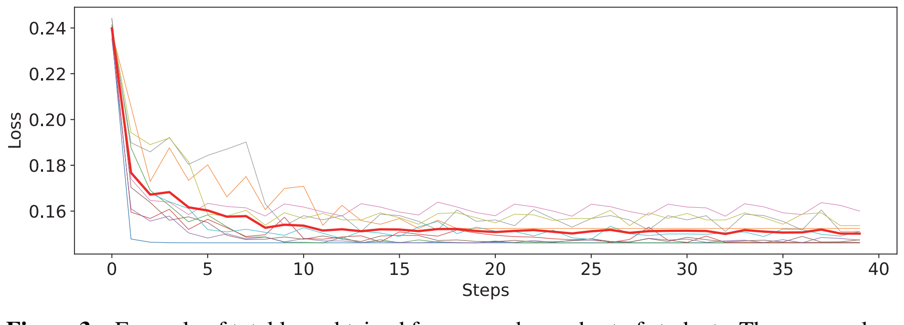


Figure 3 Example of total loss obtained from a random subset of students. The average loss of all models is shown by the bold red line. For readability, each step represents five actual train steps in the algorithm (i.e. 200 steps in total).

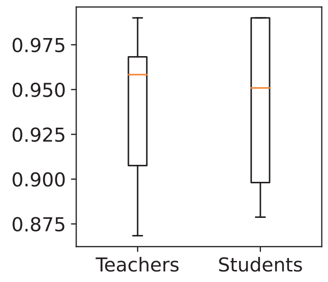


Figure 4 Teachers and students final accuracy (%) comparison. White boxes represent 50% of the models. The inner line is the central value (median).

results, there are several aspects to be considered when using KD. In the following section, we discuss the main concerns to be considered when using KD in multi-device WoT environments.

5 Knowledge Distillation in Practice

Knowledge distillation provides many benefits. However, there are quite a few concerns surrounding the use of this technology. In this section, we review these concerns and also analyze some work similar to this proposal.

In [32], the authors conducted an extensive review on knowledge distillation and student-teacher learning, where different aspects to be taken into account are evaluated. Some of these concerns are:

- **Size of the models:** Logic invites us to think that as the teacher's model grows, it will be more robust and will have more information to provide to the student. According to the works analyzed in [32], this is not always true, as there are times when the student cannot follow the teacher in the learning model, or even if the student is able to follow the teacher, the student can not absorb useful knowledge.
- **Teacher training:** An over-trained teacher will make it difficult for the learner to learn. In this case, it is also advisable that training can be done by pairs of students to encourage cooperation and enrich their models while relieving the teacher of this responsibility.
- **Multiple teachers:** The student can combine learning from multiple teachers to establish his own version of knowledge. This method collects individual predictions and aggregates them providing a more robust method for the learner.

Table 4 Comparatison with other works

Reference	Utilized Data	Teacher Accuracy	Student Accuracy	Reduction in Accuracy Compared to Teacher	Improvement in Accuracy Compared to to Baseline
Li et al. (2019) [15]	UIUC-Sports	94.20%	74.60%	7.43%	16.89%
Lopes, Fenu and Starner (2017) [16]	MNIST	95.70%	91.24%	4.80%	5.70% (decrease)
Gao et al. (2018) [5]	CIFAR-100	90.36%	91.09%	2.89%	7.81%
This work	Gathered from users for this work	94.04%	94.70%	6.04%	0.66%

- **Data-free distillation:** One of the concerns picked up in our proposal is the need to provide an initial model for the learner. This is also considered in [32] when obtaining relevant learning models. Although there are works that advocate getting preliminary results without a consistent data source, they tend to be models that are based on simple classification and environments, so data can be considered necessary to generate a first approximation of the learner model.
- **The network:** importance of students and teachers being within the same network to perform knowledge distillation has also been demonstrated. The transfer of information between different networks has hindered this process, which in some cases has not been directly possible to achieve.

Regarding the technical aspect, in [2] a study on KD techniques applied to deep learning models is carried out. Among the most relevant techniques, we find the use of weak labels and the use of feature maps with or without labeling. Both possibilities can be carried out both offline and online, depending on the requirements of the problem. Taking as a reference the works analyzed in [2], we can make a comparison between teacher and student learning models. Table 4 shows this comparison, where the works most similar to ours have been selected for the type of architecture used and the type of distillation performed:

The table shows how the selected proposals have a performance similar to ours. This means that, on the one hand, the teacher–student model is one of the most widely used techniques in knowledge distillation and, on the other hand, that it offers more than acceptable results to generate smaller knowledge networks with similar or even better performance than the source network from which the knowledge is extracted.

In addition, the authors propose a metric for distillation to compare different models and select the most appropriate one depending on the type of problem to be addressed. This metric is based on the ratio between the size of the teacher's model, the learner's model, and the score ratio between both models. In addition, the metric allows assigning a weight to indicate the importance of each model.

6 Related Works

Although not focused on the IoT field, there are many works dealing with the concept of knowledge reusability, focusing their efforts on making cheaper and faster models capable of maintaining the performance of the original models. In the natural language processing field, Sanh et al. [26] applied distillation to the large-scale model BERT and improved the overall performance in a lighter model called DistilBERT. Aguinaldo et al. [1] studied model compression and its possibilities on generative adversarial networks through KD. Proposed two novel methods and validated them on popular datasets such as MNIST, CIFAR-10, and Celeb-A. In the image processing field, works like Yu et al. [34] studied how to efficiently compute image embeddings with small networks. Xiang [33] revisited the KD fundamentals and also proposed a new framework called evolutionary embedding learning for open-set problems, which deepens on the model acceleration problem avoiding the trade-off in model accuracy. In Tang et al. [29], KD is first introduced into the recommender systems. Their proposal focuses on learning compact ranking models and is called ranking distillation because the recommendation is expressed as a ranking problem. Polino et al. [21] studied the impact of combining distillation and quantization when compressing deep NNs and demonstrates how students can reach similar accuracy levels to state-of-the-art full-precision teacher models. Finally, Guo et al. [9] focused their study on collaborative learning, dropping the traditional teacher–student scheme and treating all models as students. It enables parallel computation, fast models, and interesting generalization ability.

For the specific field of IoT environments, there are very different approaches. In previous works [24], the authors of this paper focused on federations, leveraging individually generated knowledge to generate a collective benefit. However, as it relies on federated learning (FL), federation models' structures and internal details are limited to the one imposed by the server and do not allow the extraction of fragments of knowledge. Incurring situations where a user ends up with an unnecessarily complex model to solve simple situations and vice versa. For example, Savazzi et al. [28] uses FL [18] to collaboratively train a global model that generalizes a federation behavior, and applies it to massive IoT networks to leverage the mutual cooperation of devices. *Transfer learning* can also be used to learn some or all parameters in the global model and re-learn them in local models [31], but those models cannot be re-trained for too long to avoid knowledge forgetting. Nascimiento et al. [20] also proposed an architecture that uses an ML model to analyze a dataset or interact with an environment and also monitors changes in the context. Sangsu et al. [14] propose an approach where mobile devices are also the main learning nodes. It extends FL as an egocentric collaboration in opportunistic networks where devices learn new information from encounters with other devices. Nevertheless, this approach is tied to opportunistic encounters that may or may not occur. Kabir et al. [12] propose an architecture that provides services according to users' choices using machine learning. However, there is no discussion about detecting other environments automatically and the multi-environment concept is kept inside the same home. Bahirat et al. [3] present a data-driven approach to improve privacy. By applying machine learning techniques, default smart profiles are generated and then recommended to other users. Zipperle et al. [37] propose a framework to describe task-based IoT services and their settings in a semantical manner and decouple them from their physical environments. However, it still has limitations like the increasing complexity and deteriorating accuracy over time. Finally, recommender systems are a widely used option. Although presenting limitations of mobility between environments and usually focused on a single environment, they take the concept of IoT and smart-home to a higher level such as smart-cities [22].

7 Discussion

Throughout this paper it has been seen how KD pursues, on the one hand, to extract expert knowledge from the teacher and, on the other hand, to provide this knowledge to the student in a simplified form to serve as a guide in their learning process. This has been used to provide a methodology to take advantage of the benefits of knowledge distillation within WoT environments. However, within this process, it is necessary to take into account the methods used and the quality of the knowledge, two of the main concerns also commented on in this paper.

In [7], the authors reviewed some of the most relevant works on knowledge distillation. The authors concluded that current KD techniques focus mainly on new types of knowledge, leaving aside somewhat the design of teacher–student architectures, such as the simplification of data structures, or the design of communication networks between them. This may negatively influence the performance of distillation algorithms because these architectures mainly consider the teacher–student relationship when extracting knowledge. Therefore, further progress in refining existing architectures remains an open challenge in KD.

Also, some of the related work most relevant to this proposal has been discussed in Section 6. However, existing review works claim that research in KD is still insufficient from the point of view of theoretical explanations of models and model evaluation [17, 32]. This causes one of the open technical challenges to perform a reflection on KD to measure its quality and applicability from the currently proposed architectures.

8 Conclusions

During the development of this work, the following limitations have been identified: (1) As seen, smart environments can be very different and heterogeneous, same as the needs of the users who frequent them. This fact makes the teacher–student matching a challenging task in some cases. (2) Although KD is a popular technique, it is necessary to make sure that teachers do not train students for too long. Otherwise, the students' performance will be negatively affected. (3) The distillation process can be also carried out by teachers. However, this would have disastrous scalability within a federation, since many students can request knowledge from the same teacher, saturating it. In addition, teachers are not rewarded for training students, so it is fairer for the student to make the effort.

Currently, we are improving the association and clustering of similar users and environments to ensure that students can take full advantage of the knowledge gained from teachers. In future works, we will apply the proposed approach in social environments, where many users need to be considered in the same environment, and conflicts of interest must be addressed.

Acknowledgement

This work was supported by the projects RTI2018-094591-B-I00 and PID2021-124054OB-C31 and the FPU17/02251 grant (MCIU/AEI/FEDER, UE), by the 4IE+ project (0499_4IE_PLUS_4_E) funded by the Interreg V-A España-Portugal 2014-2020 program (POCTEP), by the Department of

Economy, Science and Digital Agenda of the Government of Extremadura (GR21133, IB18030), and by the European Regional Development Fund. All authors contributed equally to this work.

References

[1] Angela Aguinaldo, Ping-Yeh Chiang, Alex Gain, Ameya D. Patil, Kolten Pearson, and Soheil Feizi. Compressing gans using knowledge distillation. *ArXiv*, abs/1902.00159, 2019.

[2] Abdolmaged Alkhulaifi, Fahad Alsahli, and Irfan Ahmad. Knowledge distillation in deep learning and its applications. *PeerJ Computer Science*, 7:e474, 2021.

[3] Paritosh Bahirat, Yangyang He, Abhilash Menon, and Bart Knijnenburg. A data-driven approach to developing iot privacy-setting interfaces. In *IUI'18*, page 165–176. Association for Computing Machinery, 2018.

[4] Caddy Becca, Pino Hick, and St Leger Henry. The best smart speakers 2021, Mayo 2021.

[5] Mengya Gao, Yujun Shen, Quanquan Li, Junjie Yan, Liang Wan, Dahua Lin, Chen Change Loy, and Xiaoou Tang. An embarrassingly simple approach for knowledge distillation. *arXiv preprint arXiv:1812.01819*, 2018.

[6] Jianping Gou, Baosheng Yu, Stephen J. Maybank, and Dacheng Tao. Knowledge Distillation: A Survey. *International Journal of Computer Vision*, 129(6):1789–1819, jun 2021.

[7] Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6):1789–1819, 2021.

[8] Joaquín Guillén, Javier Miranda, Javier Berrocal, José García-Alonso, Juan Manuel Murillo, and Carlos Canal. People as a service: A mobile-centric model for providing collective sociological profiles. *IEEE Software*, 31(2):48–53, 2014.

[9] Qiushan Guo, Xinjiang Wang, Yichao Wu, Zhipeng Yu, Ding Liang, Xiaolin Hu, and Ping Luo. Online knowledge distillation via collaborative learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[10] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *ArXiv*, abs/1503.02531, 2015.

[11] Hengyuan Hu, Rui Peng, Yu-Wing Tai, and Chi-Keung Tang. Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. *CoRR*, abs/1607.03250, 2016.

[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] Ari Keränen, Teemu Kärkkäinen, Mikko Pitkänen, Frans Ekman, Jouni Karvo, and Jörg Ott. Information – The ONE. https://akeranen.github.io/the-one/.

[14] Sangsu Lee, Xi Zheng, Jie Hua, Haris Vikalo, and Christine Julien. Opportunistic federated learning: An exploration of egocentric collaboration for pervasive computing applications. In *(PerCom 2021)*, pages 1–8, 2021.

[15] Hao-Ting Li, Shih-Chieh Lin, Cheng-Yeh Chen, and Chen-Kuo Chiang. Layer-level knowledge distillation for deep neural network learning. *Applied Sciences*, 9(10):1966, 2019.

[16] Raphael Gontijo Lopes, Stefano Fenu, and Thad Starner. Data-free knowledge distillation for deep neural networks. *arXiv preprint arXiv:1710.07535*, 2017.

[17] David Lopez-Paz, Léon Bottou, Bernhard Schölkopf, and Vladimir Vapnik. Unifying distillation and privileged information. *arXiv preprint arXiv:1511.03643*, 2015.

[18] H. McMahan, E. Moore, D. Ramage, and B. Agüera y Arcas. Federated learning of deep networks using model averaging. *ArXiv*, abs/1602.05629, 2016.

[19] Javier Miranda, Niko Mäkitalo, Jose Garcia-Alonso, Javier Berrocal, Tommi Mikkonen, Carlos Canal, and Juan M. Murillo. From the internet of things to the internet of people. *IEEE Internet Computing*, 19(2):40–47, 2015.

[20] N. Nascimento, P. Alencar, C. Lucena, and D. Cowan. A context-aware machine learning-based approach. In *CASCON*, 2018.

[21] Antonio Polino, Razvan Pascanu, and Dan Alistarh. Model compression via distillation and quantization. In *International Conference on Learning Representations*, 2018.

[22] Lara Quijano-Sánchez, Iván Cantador, María E. Cortés-Cediel, and Olga Gil. Recommender systems for smart cities. *Information Systems*, 92:101545, 2020.

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

[24] Rubén Rentero-Trejo, Daniel Flores-Martín, Jaime Galán-Jiménez, José García-Alonso, Juan Manuel Murillo, and Javier Berrocal. Using federated learning to achieve proactive context-aware IoT environments. *Journal of Web Engineering*, nov 2021.

[25] Sudipan Saha and Tahir Ahmad. Federated transfer learning: concept and applications. *Intelligenza Artificiale*, 15:35–44, 2021.

[26] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *ArXiv*, abs/1910.01108, 2019.

[27] Satyajit Sinha. State of IoT 2021, sep 2021.

[28] Stefano Savazzi, Monica Nicoli, and Vittorio Rampa. Federated learning with cooperating devices: A consensus approach for massive iot networks. *IEEE Internet of Things Journal*, 7(5):4641–4654, 2020.

[29] Jiaxi Tang and Ke Wang. Ranking distillation: Learning compact ranking models with high performance for recommender system. In *KDD '18*, page 2289–2298. Association for Computing Machinery, 2018.

[30] Wolfgang Thieme. Why It Is Time to Prioritize IoT Network and Device Management, Mayo 2020.

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

[32] Lin Wang and Kuk-Jin Yoon. Knowledge distillation and student-teacher learning for visual intelligence: A review and new outlooks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[33] Xiang Wu, Ran He, Yibo Hu, and Zhenan Sun. Learning an evolutionary embedding via massive knowledge distillation. *Int. J. Comput. Vis.*, 128(8):2089–2106, 2020.

[34] Lu Yu, Vacit Oguz Yazici, Xialei Liu, Joost van de Weijer, Yongmei Cheng, and Arnau Ramisa. Learning metrics from teachers: Compact networks for image embedding. In *CVPR 2019*, pages 2902–2911, 2019.

[35] Jiawei Zhang, Yang Wang, Piero Molino, Lezhi Li, and David S Ebert. Manifold: A model-agnostic framework for interpretation and diagnosis of machine learning models. *IEEE transactions on visualization and computer graphics*, 25(1):364–373, 2018.

[36] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76, 2021.

[37] Michael Zipperle, Achim Karduck, and In-Young Ko. Context-aware transfer of task-based iot service settings. In *Intelligent Systems and Applications*, pages 96–114. Springer International Publishing, 2021.

Biographies

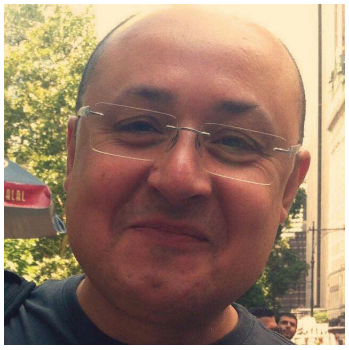
Daniel Flores-Martin received his Ph.D. degree in Computer Science at the University of Extremadura, Spain, in 2023 (International mention). His research interests are the Internet of Things, Context-Awareness, Machine Learning, Mobile Computing, and Crowd Sensing. He is currently working at the Department of Informatics and Telematics System Engineering, University of Extremadura.

Rubén Rentero-Trejo received his Bachelor's degree in Software Engineering (2019) and his MSc in Computer Science (2021) from the University of Extremadura. He is currently working at Gloin as a software developer and AI researcher, applying deep learning techniques for sentence matching and semantic alignment. He is also interested on IoT and mobile computing.

Jaime Galán-Jiménez received the Ph.D. in computer science and communications from the University of Extremadura in 2014. He is currently with the Computer Science and Communications Engineering Department, University of Extremadura, as Associate Professor. His main research interests are UAV-based networks, Software-Defined Networks, 5G network planning and design, and mobile ad-hoc networks.

José García-Alonso is an Associate Professor at the University of Extremadura, Spain, where he completed his Ph.D. in software engineering in 2014. He is the co-founder of Gloin, a software consulting company, and Health and Aging Tech, an eHealth company. His interests include quantum software engineering, mobile computing, pervasive computing, eHealth, and gerontechnology.

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.



Juan Manuel Murillo (IEEE Member) is a full professor at the University of Extremadura and general director at Computing and Advanced Technologies Foundation of Extremadura (COMPUTAEX). His research interests include software architectures, mobile computing, cloud computing, and quantum computing.