Received July 26, 2019, accepted August 17, 2019, date of publication August 26, 2019, date of current version September 17, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2937521

Human-Centric AI for Trustworthy IoT Systems With Explainable Multilayer Perceptrons

IVÁN GARCÍA-MAGARIÑO[1], RAJARAJAN MUTTUKRISHNAN[2], (Senior Member, IEEE), AND JAIME LLORET[3], (Senior Member, IEEE)

[1]Department of Software Engineering and Artificial Intelligence, Complutense University of Madrid, 28040 Madrid, Spain
[2]Department of Electrical and Electronic Engineering, City, University of London, London EC1V 0HB, U.K.
[3]Integrated Management Coastal Research Institute, Universitat Politècnica de València, 46022 València, Spain

Corresponding author: Jaime Lloret (jlloret@dcom.upv.es)

This work was supported in part by the U.K. Engineering and Physical Sciences Research under Grant EP/N028155/1, in part by the Programa Iberoamericano de Ciencia y Tecnología para el Desarrollo (CYTED) through the CITIES: Ciudades inteligentes totalmente integrales, eficientes y sotenibles under Grant 518RT0558, and in part by the Spanish council of Science, Innovation and Universities from the Spanish Government through the Diseño colaborativo para la promoción del bienestar en ciudades inteligentes inclusivas under Grant TIN2017-88327-R.

ABSTRACT Internet of Things (IoT) widely use analysis of data with artificial intelligence (AI) techniques in order to learn from user actions, support decisions, track relevant aspects of the user, and notify certain events when appropriate. However, most AI techniques are based on mathematical models that are difficult to understand by the general public, so most people use AI-based technology as a black box that they eventually start to trust based on their personal experience. This article proposes to go a step forward in the use of AI in IoT, and proposes a novel approach within the Human-centric AI field for generating explanations about the knowledge learned by a neural network (in particular a multilayer perceptron) from IoT environments. More concretely, this work proposes two techniques based on the analysis of artificial neuron weights, and another technique aimed at explaining each estimation based on the analysis of training cases. This approach has been illustrated in the context of a smart IoT kitchen that detects the user depression based on the food used for each meal, using a simulator for this purpose. The results revealed that most auto-generated explanations made sense in this context (i.e. 97.0%), and the execution times were low (i.e. 1.5 ms or lower) even considering the common configurations varying independently the number of neurons per hidden layer (up to 20), the number of hidden layers (up to 20) and the number of training cases (up to 4,000).

INDEX TERMS Explainable artificial intelligence, human-centric artificial intelligence, Internet of Things, multilayer perceptron, smart kitchen, emotion detection.

I. INTRODUCTION

Artificial intelligence (AI) can provide different functionalities to systems composed with Internet of Things (IoT) objects, such as (a) providing an intelligent communication between two devices using different IoT objects [1], (b) smart collaboration of IoT elements in the context of smart cupboards [2], and (c) smart sensing by composing information from different sensors like in the case of prediction where to fumigate in precision agriculture [3].

The existing AI techniques such as neural networks (NNs), support vector machines (SVM), k-nearest neighbours (KNN) and random forests, and the newest ones like deep learning (based on NNs but with high numbers of layers) are able to learn from IoT sensor inputs when these are associated with certain target features. For example, in agriculture, the AI algorithms can learn from the pictures captured by IoT objects associated with the known need of irrigation or pesticide, and then the learned models can estimate the need of irrigation or pesticide when capturing new pictures [4]. In this particular case and other similar ones, the human user uses the intelligence of the IoT system as a black-box fashion. They check that it works with a reasonable accuracy without understanding the reasons behind each decision, and eventually the user starts trusting the intelligent IoT knowing that there is an admissible percentage of errors.

Nevertheless, in some domain applications in fields such as medicine, health, safety and security, the intelligent

The associate editor coordinating the review of this article and approving it for publication was Sherali Zeadally.

IoT systems can assist humans in taking decisions, but some actions cannot be directly taken without human supervision [5]. In some occasions, the AI percentage of errors is not tolerable due to the repercussion in other human beings. On the one hand, unsupervised decisions would probably violate human rights or raise serious ethical implications. On the other hand, supervising decisions would require almost as much effort as taking decisions from scratch giving the black-box nature of the common existing AI algorithms.

In this context, the current work proposes to apply human-centric AI (HAI) in IoT systems, so that IoT systems cannot only learn from users but also provide easy-to-understand explanations about decisions or estimations. In this way, users have the opportunity to understand the reasons behind AI decisions, to fasten the supervision process confirming or refuting the decision, based on whether the highlighted reasons make sense in the given context. In particular, this work proposes a novel HAI approach for the auto-generation of explanations based on the analysis of discriminant inputs in the training cases and artificial neuron weights in the multilayer perceptron (MLP) trained with backpropagation.

The remainder of this paper is organised as follows. The next section introduces related works highlighting the literature gap covered by the current approach. Section III presents the proposed HAI approach composed of three HAI auto-generation explanation techniques, determining their algorithms among other aspects. Section IV illustrates the current approach in the context of a smart IoT kitchen for detecting depression with a simulator developed for this purpose. Section V evaluates the current approach with the smart IoT kitchen simulator, assessing the explanations and the performance of the HAI techniques in terms of execution time. Section VI discusses the most relevant aspects about this research and its results, and section VII mentions the conclusions and our most relevant future research lines.

II. RELATED WORK

HAI is a starting science field that is creating great expectations and is promoted in the European Union, with the goal of achieving transparency and trustworthy in the already existing AI techniques. The community expects that this field reinforces and facilitates the collaboration between human beings and AI. In this research line, [6] recently proposed 18 guidelines for human-AI interaction. Among these the guidelines, they proposed that the AI system (a) shows contextually relevant information that explains certain decisions or estimations, (b) makes clear why the system did what it did, and (c) supports efficient correction by users. They conducted a study with both human-computer interaction (HCI) specialists and employees from a software company to determine these 18 guidelines, extracted from 150 AI-related design recommendations and three rounds of evaluations. This work supports the relevance of achieving HAI and, consequently, the relevance of the current work, but it did not propose any specific implementation as the current work does.

Several works have applied AI techniques in IoT systems. For instance, [7] presented an IoT agricultural system that used a NN for smart irrigation with a proper schedule, based on the data received from a sensor information unit. Farmers could use the system in admin mode and then the system could continue functioning in an automatic monitoring mode with the learned knowledge. This system achieved an overall water savings of 67% over the traditional irrigation mechanism. In this line of research, [8] proposed to use deep NNs (DNNs) in IoT-based hydroponic systems. Thanks to machine-to-machine interaction, they proposed a solution in which the hydroponic system was able to work autonomously and intelligently. They illustrated their approach with a prototype for tomato plant growth using Tensor Flow for implementing the DNN and a Raspberry Pi3 among other processing units. In addition, [9] developed an IoT fitness system that collected information from IoT sensors tracking exercisers, and this system used AI for providing them guidance to build their bodies. Reference [10] introduced a special issue that applied AI algorithms for the interoperability of IoT, such as NNs, swarm intelligence and genetic algorithms. Reference [8] presented a self-evolving and self-adapted approach that applied unsupervised self-learning for achieving data interoperability in IoT systems. Their approach was tested with dynamic self-organising maps in the context of real data from a fire department in Australia. Nevertheless, none of these works developed a system that used HAI for providing easy-to-understand auto-generated explanations of the AI decisions.

The field of eXplainable AI (XAI) is very related to HAI, and focuses on the generation of explanations for AI algorithms. In particular, [11] proposed to use neural logic networks (NLN) to implement XAI. In NLNs, a NN structure with the appropriate weights is used to represent a set of flexible operations. They introduced a supervised incremental learning algorithm with neural logic. They provided an illustrative example based on a decision tree, to describe how an explanation could be generated from a NLN. The survey about XAI of [12] discussed how much XAI was necessary highlighting the needs of (1) explaining for justifying AI decisions, (2) explaining for later controlling the system, (3) explaining for improving the system, and (4) explaining for discovering new knowledge. They also introduced the categories of XAI for respectively (a) global interpretation of the learning model, and (b) understanding of specific cases estimated with a learned model for concrete input values. Both of these categories have been considered in the current approach. They also mentioned some useful applications in the fields of healthcare, law, finance and military, and indicated XAI still needs of novel solutions for overcoming the technical challenges around XAI.

In this content, the current work proposes several solutions for advancing HAI and XAI in the context of MLP in IoT environments, which are novel to the best of the authors' knowledge. We illustrated our HAI solutions with a smart IoT kitchen simulator, towards partially covering the

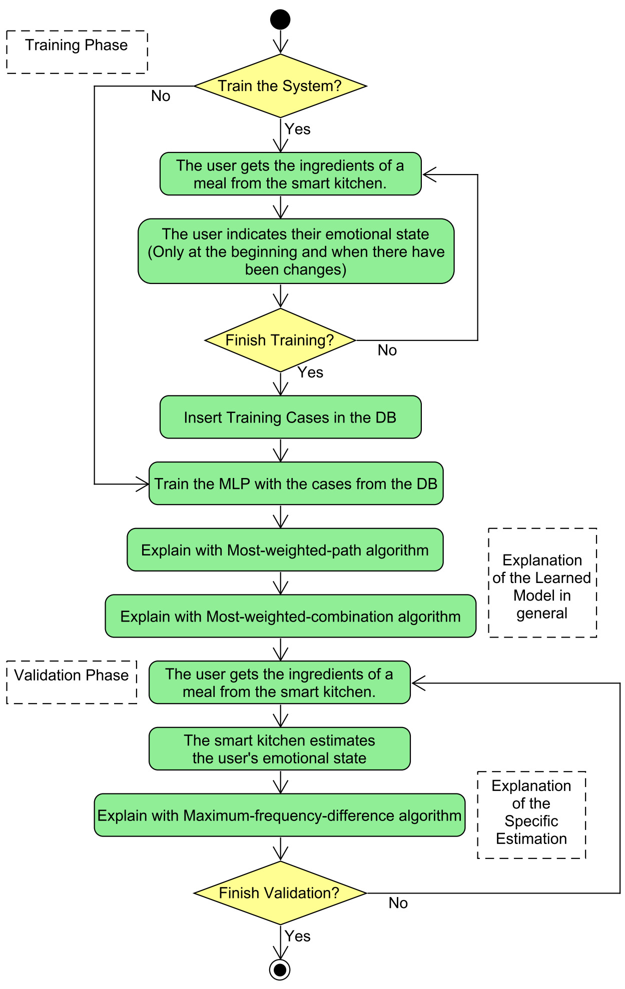

Training Phase
Train the System?
No
Yes
The user gets the ingredients of a meal from the smart kitchen.
The user indicates their emotional state (Only at the beginning and when there have been changes)
Finish Training?
No
Yes
Insert Training Cases in the DB
Train the MLP with the cases from the DB
Explain with Most-weighted-path algorithm
Explain with Most-weighted-combination algorithm
Explanation of the Learned Model in general
Validation Phase
The user gets the ingredients of a meal from the smart kitchen.
The smart kitchen estimates the user's emotional state
Explain with Maximum-frequency-difference algorithm
Explanation of the Specific Estimation
Finish Validation?
No
Yes

FIGURE 1. **Block diagram of the smart IoT kitchen simulator.**

literature gap about empowering HAI techniques in IoT systems for making users understand and trust AI decisions in IoT systems.

III. HAI APPROACH FOR IOT

The proposed HAI approach is composed of a training phase of the IoT system and a phase of normal use (referred also as the validation phase to align with the AI and machine learning literature). Figure 1 provides an overview of this approach with a block diagram. Although the approach is generic for different IoT systems, we have preferred to illustrate this overview with the particular case study of a smart IoT kitchen simulator (later presented in section IV), to avoid continuously using abstract terms such as inputs and outputs, and facilitate the understanding for a wider audience. The user can decide either (a) to train the IoT system to be properly customised or (b) to use some pre-loaded knowledge from a database (DB). In both cases, the IoT system provides an explanation of the AI behaviour (implemented with a MLP), by describing the most relevant aspects, with two algorithms (''Most-weighted-path'' and ''Most-weighted-combination'') that are later introduced respectively in sections III-A and III-B. The user can use the IoT system and get an estimation for each usage case, whose most relevant aspect is explained with the ''Maximum-frequency-difference'' algorithm, later presented section III-C.

In its conception, HAI could be applied to any AI technique. The presented HAI approach proposes to use a MLP (a common type of NN) for training the IoT system with backpropagation [6], which is a common learning algorithm for this kind of NN. For the training, the inputs are the sensed information from the sensors of the IoT system, and the outputs for these inputs should be assigned by a human based on a known truth. The outputs can be either labels in classification problems or numeric values in regression problems. Most common labels of IoT systems are related to the estimation of the user's states, for example the detection of (a) whether the user has fallen or not, (b) whether the user is getting depressed, or (d) whether the user is sleeping.

After MLP is trained, the learned MLP model can be used to estimate the output such as certain user's state for new IoT sensor inputs. The learned model is mainly the input weights of each neuron. The output is obtained by calculating each neuron's output from the weighted inputs and applying a sigmoid function, each neuron layer after the previous one starting from the MLP input, until the MLP provides an output.

The novel contribution of this approach is to add an explanation after training the MLP and for each prediction of the IoT system, so the user can understand the reason behind the learned model. This approach proposes three alternative techniques for automatically generating HAI explanations.

The first two proposed HAI techniques analyse the MLP learned model for explaining its most relevant features in general, right after the training phase. These techniques are based on the analysis of weights of artificial neuron inputs (also referred as dendrites). The third proposed HAI technique explains the MLP estimation for each input case based on the analysis of training cases. The following three subsections respectively present the three proposed HAI techniques.

A. *MOST-WEIGHTED-PATH EXPLANATION*

We denote our first technique as the ''Most-weighted-path'' explanation, and it is generated with the algorithm 1. It basically starts from the output neuron, and selects the most weighted input. Then, it analyses the neuron connected to this input in the same manner, going again to the neuron of the previous neuron layer connected with the highest weight. It continues this process recursively until it reaches to an IoT input through the most weighted dendrite of the corresponding neuron of the input layer of neurons. Then, it creates an auto-generated explanation indicating that one of the most relevant features considered in this training model for estimating the positive output category is the IoT input reached by the algorithm.

An example of Most-weighted-path explanation is ''*In the learned model for the smart kitchen, the most relevant input*

Algorithm 1 Most-Weighted-Path Explanation: It Provides a HAI Explanation Based on the Path From the Output to the Most Relevant Input Based on the Selection of the Most Weighted Dendrites

```
1: function explainMostWeightedPath(mlp, names)
2:     current ← mlp.outputNeuron
3:     while mlp.isNeuron(current) do
4:         dendrite ← mostWeightedDendrite(neuron)
5:         current ← mlp.connectedTo(dendrite)
6:     input ← current
7:     inputName ← names.inputs[input]
8:     explanation ← 'In the learned model for the
       '+names.IoTsystem+', the most relevant input for esti-
       mating that you are '+ names.highestOutputValue+' is
       that '+names.userAction+' '+ inputName+'.'
9:     return explanation
```

for estimating that you are depressed is that you are eating chocolate."

In order to further formalise this HAI approach, we define equations about the analysis of dendrite weights of neurons in the MLP using the following notation:

- L_i is the layer of the MLP in the level i, where 0 is the level of inputs, $\{1, \ldots, N-1\}$ are the hidden layers where the first one is the closest to the inputs, and N is the layer with the neuron that provides the output;
- $s(L_i)$ is the size of a layer L_i, indicating either the number of inputs or the number of neurons
- N is the number of layers in the MLP;
- n refers to a particular neuron (or an input in case of layer 0), normally referred as belonging to a particular layer;
- $n_{i,j}$ is the neuron (or input in the case of layer 0) in the j position in the layer L_i;
- x and x_i respectively refer to any MLP input and the input in the i position;
- X refers to the set of all the inputs;
- w is the weight of a dendrite, normally referred as belonging to a neuron; and
- $w_{i,j,k}$ refers to the weight of the dendrite belonging to a neuron $n_{i+1,j}$ that is in the k position, so this dendrite connects the input/neuron $n_{i,k}$ with neuron $n_{i+1,j}$.

In order to formally define the most weighted path, we define a path with the tuple P composed with an input and $N-1$ neurons when the following condition is satisfied:

$$P(<p_0, p_1, \ldots, p_n>) \Leftrightarrow (\forall i \in [0, N] : p_i \in L_i) \quad (1)$$

We define that a path is the most weighted with the M predicate, by means of the following equation:

$$\begin{aligned} &M(<p_0, p_1, \ldots, p_n>) \\ &\quad \Leftrightarrow (P(<p_0, p_1, \ldots, p_n>) \wedge \\ &\quad (\forall i \in [0, N-1] : (\exists w_{i,j,k} \in n_{i+1,k} : \\ &\quad (p_i = n_{i,j}) \wedge (p_{i+1} = n_{i+1,k}) \wedge \\ &\quad w_{i,j,k} = (Max.y \in [0, s(L_i)-1] : w_{i,j,y})))) \quad (2) \end{aligned}$$

An input x is classified as relevant with the R predicate, which is defined as follows:

$$\begin{aligned} R(x) \Leftrightarrow (\exists <x, p_1, \ldots, p_n> \in L_0 \times L_1 \times \ldots \times L_n : \\ M(<x, p_1, \ldots, p_n>)) \quad (3) \end{aligned}$$

The auto-generated explanation for the most-weighted-path is generated for the x input that satisfies the $R(x)$ relevance condition.

B. MOST-WEIGHTED-COMBINATION EXPLANATION

The second proposed HAI technique is denoted as the "Most-weighted-combination" explanation, and algorithm 2 implements this technique. It is based on the selection of the most weighted dendrite between the input layer of neurons and the first hidden layer of neurons. From the input neuron connected to it, the algorithm selects the two most weighted IoT inputs as a relevant combination if their weights surpass certain threshold. Otherwise, it continues with the neuron of the input neuron layer connected through the most weighted dendrite from the first hidden neuron layer, and so on. It presents the reached combination of two inputs as one of the most relevant ones to the user properly explained.

An example of most-weighted-combination explanation is "*In the learned model for the smart kitchen, the most relevant input combination for estimating that you are depressed is that you are eating pasta and a spicy condiment.*"

To precisely determine the meaning of a relevant pair combination, we start by defining the predicate B as a pair of inputs x_y and x_z connected to a neuron $n_{1,j}$ of the first hidden layer with the most weighted connections, considering all the inputs connected to this neuron, with the following equation:

$$\begin{aligned} B(<x_y, x_z, n_{1,j}>) \Leftrightarrow (\forall i \in [0, s(L_0)-1] : \\ (x_i = x_y) \vee (x_i = x_z) \vee ((w_{0,i,j} \leq w_{0,i,y}) \\ \wedge (w_{0,i,j} \leq w_{0,i,z}))) \quad (4) \end{aligned}$$

We define the S surpass predicate for a pair of inputs x_y and x_z and a neuron $n_{1,j}$ if all these satisfy the previous B predicate and the weights related to these inputs and the neuron surpass a certain threshold T:

$$\begin{aligned} S(<x_y, x_z, n_{1,j}>) \Leftrightarrow B(<x_y, x_z, n_{1,j}>) \wedge \\ (w_{0,y,j} > T) \wedge (w_{0,z,j} > T) \quad (5) \end{aligned}$$

The most relevant combination C predicate is defined as the most weighted combination of inputs that are connected to the neuron that has the most weighted connection with the second hidden layer, considering in the first layer only the neurons that have pairs of inputs that surpass the T threshold:

$$\begin{aligned} &C(<x_y, x_z>) \\ &\Leftrightarrow (\exists n_{1,j} \in L_1 : S(<x_y, x_z, n_{1,j}) \wedge \\ &\quad (\forall <x_a, x_b> \in X \times X : (\nexists n_{1,q} \in L_1 : S(<x_a, x_b, n_{1,q}) \wedge \\ &\quad ((Max.k \in [0, s(L_2)-1] : w_{1,q,k}) > \\ &\quad (Max.k \in [0, s(L_2)-1] : w_{1,j,k})) \quad (6) \end{aligned}$$

Algorithm 2 Most-Weighted-Combination Explanation: It Provides a HAI Explanation Based on the Most Relevant Combination of Two Inputs Based the Most Weighted Dendrites of the First and Second Neuron Layers

```
1: function explainMostWeightedCombination(mlp, names)
2:     layer ← 1                    ▷ Second layer, as count starts on 0
3:     numInputs ← 2
4:     dendrites ← mlp.getDentritesLayer(layer)
5:     quicksortByWeightDescendentOrder(dendrites)
6:     found ← false
7:     i ← 0
8:     while i<dendrites.length and not found do
9:         dendrite ← dendrites[i]
10:        inputNeuron ← mlp.connectedTo(dendrite)
11:        inputDendrites ← mlp.mostWeightedDendrites(
12:                          neuron, numInputs)
13:        found ← true
14:        for j ∈ [0, numInputs) do
15:            if inputDendrites[j].weight>threshold then
16:                combination[j] ← mlp.connectedTo(
17:                          inputDendrites[j])
18:            else
19:                found ← false
20:        i ← i + 1
21:    if found then
22:        explanation ← ‘In the learned model for the ‘+names.IoTsystem+’, the most relevant input combination for estimating whether you are ‘+names.highestOutputValue+’ is that ‘+names.userAction+’ ‘+ names.inputs[combination[0]]+’ and ‘+names.inputs[combination[1]]+’.’
23:    else
24:        explanation ← ‘No combination of two inputs is especially relevant.’
25:    return explanation
```

The most-weighted-combination explanation is generated for the pair of inputs that fulfil the C predicate.

C. MAXIMUM-FREQUENCY-DIFFERENCE EXPLANATION

The last technique is to store the whole training dataset in a database, and then retrieve the most similar cases for each IoT system input case. This technique then analyses which sensor inputs have been the most discriminative ones in favour of the output provided by the trained MLP. In particular, this technique is designed assuming that each IoT sensor provides binary values (either true/false or zero/one regarding the preferred encoding) that are used as inputs of the MLP. The MLP output is also assumed to be binary (for instance distinguishing between two user states). For each activated input (meaning its value is true) of the IoT system, this technique calculates the percentage of cases that have the same activated input for separately (a) the cases with the same output as the one predicted by the MLP, and (b) the cases with a different output from the predicted by MLP. After this, it calculates the difference of percentage as the former percentage minus the latter one. This is performed with the algorithms 3 and 4.

Algorithm 3 It Calculates the Difference of Percentages of Cases That Have an Activated Input Between the Ones that Match and the Ones That Mismatch a Predicted Output Value

```
1: function diffPercen(inputName, prediction)
2:     $match$ ← percentage(inputName,prediction,true)
3:     $mismatch$ ← percentage(inputName,prediction, false)
4:     return $match - mismatch$
```

Algorithm 4 It Calculates the Percentage of Cases That Have an Activated Input That Match/Mismatch a Given Output

```
1: function percentage(nameInput, outputValue, match)
2:     if match then
3:         $op$ ← ‘$=$’
4:     else
5:         $op$ ← ‘$<>$’
6:     $total$ ← ‘SELECT COUNT(*) FROM training ‘ +
7:             ’WHERE training.output’+$op$ + $outputValue$
8:     $selected$ ← ‘SELECT COUNT(*) FROM training’+
9:             ’WHERE training.’+$nameInput$+’ = true’+
10:            ’AND training.output’+$op$ + $outputValue$
11:    return $100 * selected/total$
```

The last proposed HAI technique is implemented with algorithm 5, and we denote it as the ‘‘Maximum-frequency-difference’’ explanation. This algorithm explores all the differences of percentages, and it provides an explanation based on the input that has the highest difference of percentages.

An example of explanation generated with this HAI algorithm is the following: ‘‘*The smart kitchen estimates that you are depressed because among other reasons you are eating spicy condiments, which is 41.3% more frequent in people in this emotional state than in people with other emotional states*’’.

In order to formally define the maximum-frequency-difference, we will use the following notations:

- D is the dataset for training, which is a set of cases represented as tuples $< X, o >$ where X are the values of IoT system inputs and o is the boolean correct output;
- $X[i]$ is the boolean input value in i position of the IoT system input X; and
- $s(X)$ is the number of inputs.

The following equation defines the f function for calculating the frequency of cases that have a given input in position i with a *true* value, considering only the ones with

Algorithm 5 Maximum-Frequency-Difference Explanation: It Provides a HAI Explanation Based on the Most Discriminative Input, Measured as the One With the Highest Difference of Frequency Percentage for the Given Prediction

```
1: function explainMaxFreqDiff(caseInputs, prediction, names)
2:     maxDiff ← minIntValue
3:     maxInputName ← ‘’
4:     for i ∈ [0,names.inputs.length) do
5:         if caseInputs[i] then
6:             inputName ← names.inputs[i]
7:             diff ← DiffPercen(inputName,prediction)
8:             if diff>maxDiff then
9:                 maxInputName ← inputName
10:                maxDiff ← diff
11:    explanation ← ‘The ‘+names.IoTsystem+’ estimates that you are ‘+prediction+’ because among other reasons ‘+names.userAction+’ ‘+maxInputName+’, which is “+maxDiff+’% more frequent in people in this ‘+names.state+’ than in people with other ‘+names.states+’.’
12:    return explanation
```

a certain o output:

$$f(i, o) = \frac{(Count. < X, o' > \in D : X[i] \wedge o = o')}{(Count. < X, o' > \in \wedge o = o')} \quad (7)$$

This approach calculates the difference of the frequencies between the ones that have a particular input in the i position, when comparing the ones with the o output and the ones with the opposite output, with the d function determined below:

$$d(i, o) = f(i, o) - f(i, \neg o) \quad (8)$$

The E predicate determines that for a given input X and estimation o, the most discriminative input is the one in the position i, and is defined as follows:

$$E(X, o, i) \Leftrightarrow (\forall j \in [0, s(X) - 1] : d(j, o) \leq d(i, o)) \quad (9)$$

In this way, given an input case X and an estimated output o, the maximum-frequency-difference explanation is provided with the name of the input in i position where $E(X, o, i)$ is satisfied. The difference of frequencies is provided by the value calculated with $100 \times d(i, o)$, in which the purpose of the multiplication by 100 is to present a percentage instead of a ratio.

IV. CASE STUDY WITH THE SIMULATION OF A SMART KITCHEN

In order to illustrate this approach, we applied it in the context of a smart kitchen where the different shelves of the fridge, the compartments of cupboards and drawers of other pieces of furniture are tracked with sensors. The kitchen constitutes an IoT system that is assumed to know the content of the different spaces, organised according to a pre-established classification. The purpose is to train the IoT system about the relation of the food the user eats and whether they are depressed.

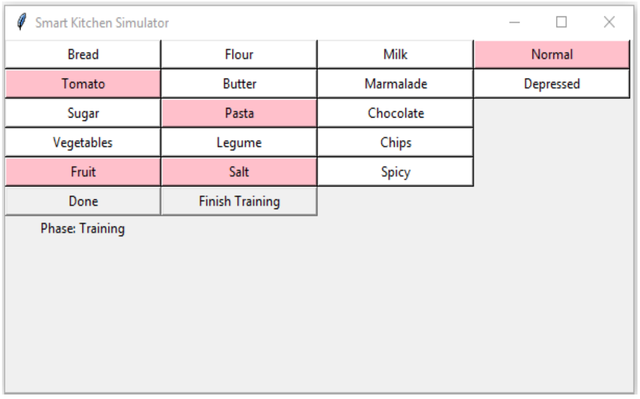


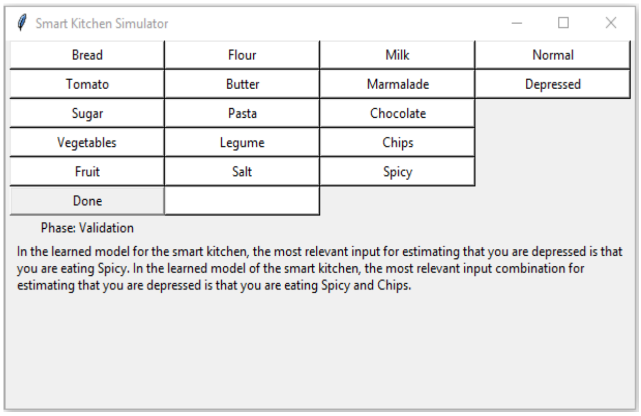
FIGURE 2. Smart IoT kitchen simulator in the training phase.

This smart-kitchen scenario is relevant because once the IoT system is trained it could detect the depressions without requiring any specific action from users besides their daily activities. This could be useful for people suffering diseases such as different cancer types where their emotional states are relevant for surviving them.

In order to simulate this scenario, we have developed a Python application, in which the user can select the compartments/shelves/drawers represented as buttons with their food category written on it, in a user interface (UI). In this way the user can simulate all the ingredients that they would take for preparing a meal. Figure 2 presents the UI of this smart kitchen simulator in the training phase. The selected items are represented by changing the background colour to pink. When they have stopped selecting ingredients for a meal, the user clicks the “Done” button to tell this to the system. In the real-life scenario, the system would detect the end of the selection of ingredients of a meal by a timeout. For example, when the user would not take any food for more than 30 min, the system would assume that the user has finished taking ingredients for a meal.

The smart kitchen simulator can assist both phases of HAI, the training of the system and its validation. The phase is indicated in the bottom of the UI, and the user can change from training phase to validation phase by clicking on the “Finish Training” button.

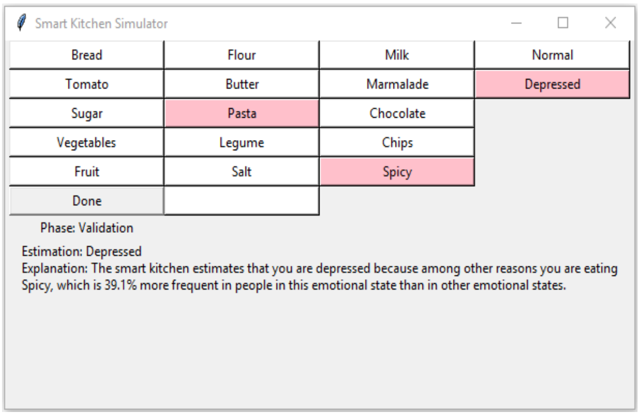
In the training phase, the user has to select their emotional state in the right side of the UI, indicating their state between normal and depressed, besides indicating the ingredients for their meal. When the user clicks on the Done button, the system stores the information locally associating the ingredients of each meal with the user is depressed or not. Once the user clicks on the button to finish the training phase, all the collected data are uploaded to a table called “training” in a database in MariaDB (a fork of MySQL) database management system (DMBS) within a XAMPP environment. Alternatively, if the user just wants to use the training dataset previously stored in the database, they can do it by just



Phase: Validation

In the learned model for the smart kitchen, the most relevant input for estimating that you are depressed is that you are eating Spicy. In the learned model of the smart kitchen, the most relevant input combination for estimating that you are depressed is that you are eating Spicy and Chips.

FIGURE 3. Smart IoT kitchen simulator explaining the learned model by means of the analysis of neuron weights.



Phase: Validation

Estimation: Depressed
Explanation: The smart kitchen estimates that you are depressed because among other reasons you are eating Spicy, which is 39.1% more frequent in people in this emotional state than in other emotional states.

FIGURE 4. Smart IoT kitchen simulator explaining the estimation for one specific meal.

clicking on the Finish Training button without storing any meal before.

The application creates a MLP with the implementation provided with a Scikit-learn [13] library with the class ''MPLClassifier of the ''sklearn.neural_network'' package. In particular, we used a 'lbfgs' solver (an optimizer in the family of quasi-Newton methods), a 10^{-5} alpha, 15 and 5 hidden layer sizes and a random state of 1. We trained the MLP using the common backpropagation algorithm implemented in the ''fit'' method of this classifier.

The IoT smart kitchen simulator applies the novel technique most-weighted-path introduced in the previous section about the analysis of weights of neurons inputs in order to detect the highest priority path from the neuron to one input, presenting this input as one of the highest priority. In addition, it also analyses whether there is any combination of IoT inputs of special relevance with most-weighted-combination technique with a weight threshold of 0.5 (see previous section to remind how this threshold is used). The smart IoT kitchen simulator shows the auto-generated explanations to the user in one paragraph describing the most relevant aspects of the trained model, as one can observe in the example of Figure 3. In this way, the user can understand the most relevant reasons behind the model learned from the training.

In the validation phase, the user simulates taking ingredients from the smart kitchen by clicking on the corresponding buttons. In the real-life scenario, once the system is properly trained, the validation phase would be the normal usage in which the user would use the smart kitchen and this would estimate the user state between normal and depressed. In the simulator application, when the user finishes taking food (i.e. simulated by clicking on Done), the system uses the MLP trained model with the corresponding food inputs, and estimates the user's emotional state between normal and depressed. The UI displays this by setting the corresponding state with pink background colour, and showing a message in the bottom of the application.

The smart IoT kitchen simulator provides an explanation for each estimation of the user state, as one can observe in the example of Figure 4, using the novel HAI maximum-frequency-difference technique introduced in the previous section. The system indicates which ingredient has been one of the most decisive indicating the difference of frequency percentages of this ingredient between the estimated user's emotional state and the other possible emotional state.

V. EVALUATION

In order to evaluate this approach with the presented case study, we used 30 meals from the first Google entries retrieved by the words 'depression meals' that actually provided a list of ingredients that mainly matched our categories, and the 30 meals from first Google entries retrieved by the words 'healthy meals' with the same conditions. We randomly shuffled the meals of each category. We split the data in three groups, so that each one had 20 depression meals and 20 healthy meals. We randomly shuffled again each group. We performed a 3-fold cross validation, in which in each round, we trained the system with two groups and validated with the remaining one. We repeated this three times so that the validation group was different in each group. Thus, we had the data from 60 predictions about both depression and healthy meals, in which each prediction was performed with a meal unknown by the trained system, so in this manner we could assess the capacity of the HAI approach in predicting and explaining unknown data. In this way, cross-validation avoided bias of the results because of overfitting as commonly done in the literature [14].

After training the model in each round with two thirds of the dataset, the system provided the explanations indicated in table 1, as the concatenation of most-weighted-path and most-weighted-combination explanation.

The human user indicated that all the three pairs of explanations made sense given the training datasets used in each round, and were easy to understand.

In the estimations of the cases, we collected the 60 explanations for the 60 estimations resulting from the 3-fold cross validation. Table 2 presents the explanations provided the smart IoT kitchen simulator for the different

TABLE 1. HAI explanations provided for the three learned models in the 3-fold cross validation.

Explanation	Made sense
In the learned model for the smart kitchen, the most relevant input for estimating that you are depressed is that you are eating Sugar. In the learned model of the smart kitchen, the most relevant input combination for estimating whether you are depressed is that you are eating Legume and Salt.	Yes
In the learned model for the smart kitchen, the most relevant input for estimating that you are depressed is that you are eating Pasta. In the learned model of the smart kitchen, the most relevant input combination for estimating whether you are depressed is that you are eating Pasta and Fruit.	Yes
In the learned model for the smart kitchen, the most relevant input for estimating that you are depressed is that you are eating Spicy. In the learned model of the smart kitchen, the most relevant input combination for estimating whether you are depressed is that you are eating Spicy and Fruit.	Yes

validation cases indicating their absolute frequencies. The human experimenter read these explanations, and indicated which explanations made sense. The ''made sense'' column indicates the number of cases in which each explanation made sense, or ''All' if it was in all cases. For the explanations that were provided only once, we just indicate whether it made sense with ''Yes'' or ''No''. Figure 5 shows the frequencies of the explanations graphically indicating in the legend which of these made sense (with the reasonable and non-reasonable adjectives). Notice that we have gathered explanations that only slightly varied in the difference of frequency in eating certain food element for the sake of brevity, and these differences appeared because cross-validation used different training subsets.

As one can observe, the explanations made sense in 58 out of the 60 estimation cases, which represented 96.67% of these cases. The human experimenter mentioned that he found easy to understand all the explanations that made sense. Each of the two explanations that did not made sense only appeared once considering a pull of 60 cases. Thus, each of the two non-sense explanations only appeared in the 1.67% of the cases. We checked the estimations in these two cases, and in both cases the estimations were wrong, meaning that they did not match the real classification. Thus, a non-reasonable explanation of an estimation may be a good indicator of the non-reliability of this estimation.

The accuracy of the system in determining whether meals were related with depression was 81.67%, the sensitivity (also called true positive rate) was 86.67%, and the specificity (also called the true negative rate) was 76.67%, considering unknown cases that were not used in the training, as common done in cross-validation. Figure 6 graphically shows these classification measurements. The results of these classification metrics show that our proposed approach was able to explain an AI model that relatively worked properly in a particular domain, in this case a MLP applied to detect depressions by analysing the combinations of common meal components.

TABLE 2. HAI explanations provided for the validation cases.

Frequency	Explanation	Made sense
25	The smart kitchen estimates that you are normal because among other reasons you are eating Vegetables, which is 65.0% more frequent in people in this emotional state than in other emotional states.	All
18	The smart kitchen estimates that you are depressed because among other reasons you are eating Sugar, which is 55.0% more frequent in people in this emotional state than in other emotional states.	All
5	The smart kitchen estimates that you are depressed because among other reasons you are eating Spicy, which is 15.0% more frequent in people in this emotional state than in other emotional states.	All
3	The smart kitchen estimates that you are depressed because among other reasons you are eating Bread, which is 10.0% more frequent in people in this emotional state than in other emotional states.	All
3	The smart kitchen estimates that you are depressed because among other reasons you are eating Pasta, which is 5.0% more frequent in people in this emotional state than in other emotional states.	All
2	The smart kitchen estimates that you are depressed because among other reasons you are eating Flour, which is 45.0% more frequent in people in this emotional state than in other emotional states.	All
1	The smart kitchen estimates that you are depressed because among other reasons you are eating Salt, which is 0.0% more frequent in people in this emotional state than in other emotional states.	No
1	The smart kitchen estimates that you are depressed because among other reasons you are eating Tomato, which is -20.0% more frequent in people in this emotional state than in other emotional states.	No
1	The smart kitchen estimates that you are normal because among other reasons you are eating Bread, which is 5.0% more frequent in people in this emotional state than in other emotional states.	Yes
1	The smart kitchen estimates that you are normal because among other reasons you are eating Tomato, which is 20.0% more frequent in people in this emotional state than in other emotional states.	Yes

To further present the classification performance of the IoT system, we calculated the three metrics accuracy, sensitivity and specificity in the same cross-validation as described before, but changing the number of neurons. Figure 7 shows the results with the MLP with two layers of hidden neurons, both of which with the same number of neurons in each trial. One can observe that the classification metrics had high variations with low numbers of neurons (e.g. up to three). From five neurons forward, the variations were smaller. For larger numbers (i.e. from 15 neurons forward), all the classification metrics were mainly in the range 80 to 90% with some exceptions. Generally, the sensitivity was higher than specificity, and the accuracy was the average of these two.

We also assessed different configurations of numbers of layers. Figure 8 shows the results of accuracy, sensitivity and specificity in the cross-validations previously introduced but with different numbers of hidden layers, each of which with 10 neurons. It is worth highlighting that it achieved relatively appropriate results up to 5 layers, but from 6 layers forward the system started getting very low results in at least one of

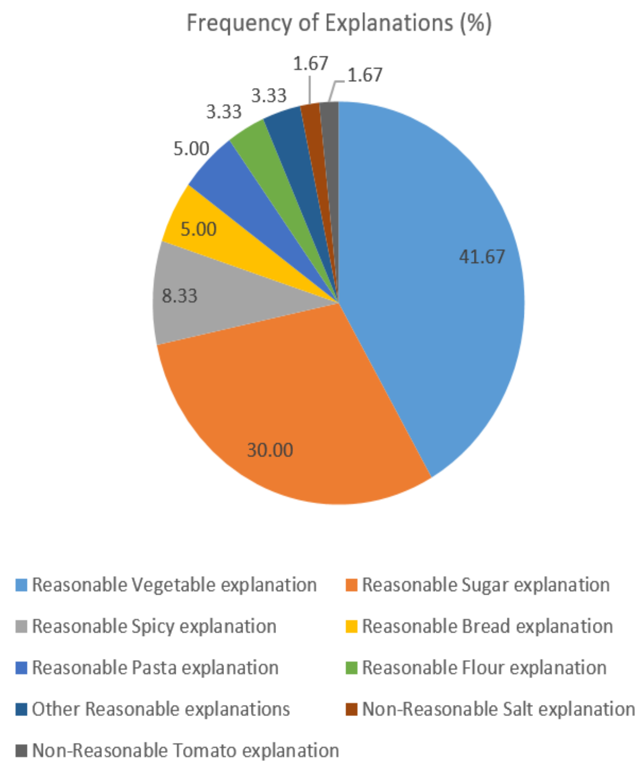


FIGURE 5. Frequency of the explanations for particular estimations indicating whether they were reasonable.

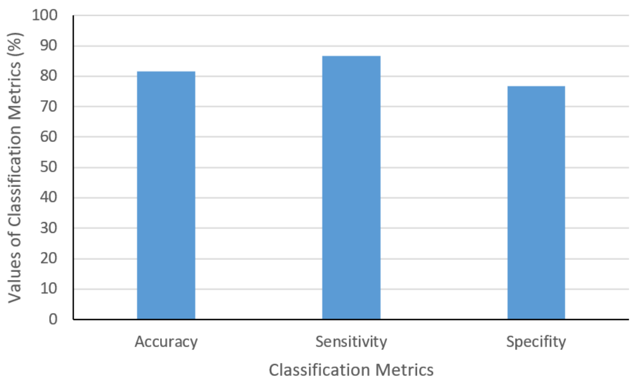


FIGURE 6. Measurements with classification metrics.

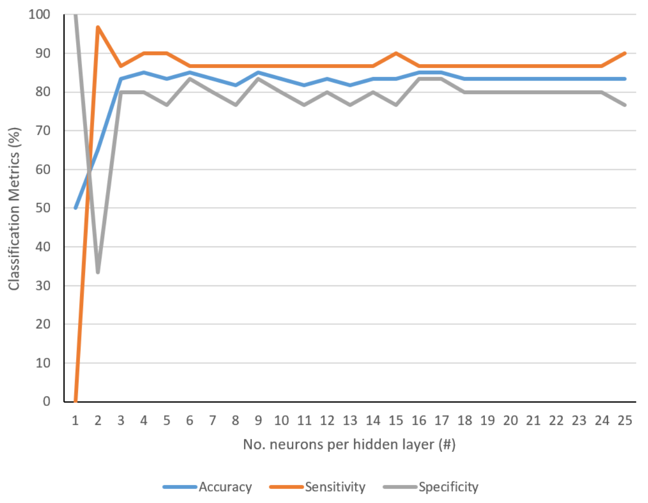


FIGURE 7. Classification metrics with different numbers of neurons per hidden layers.

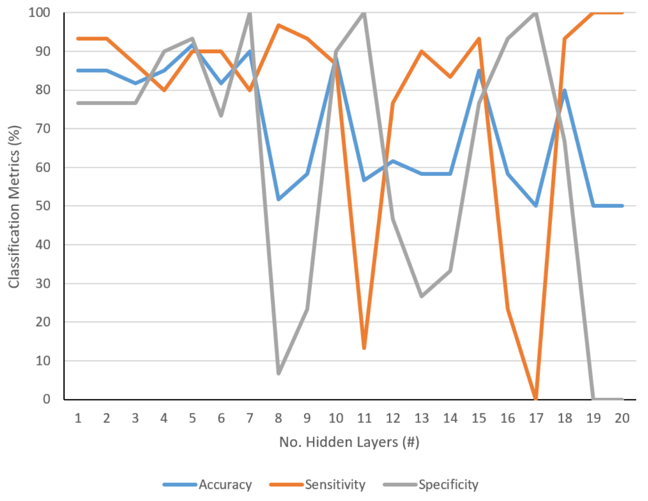


FIGURE 8. Classification metrics with different numbers of hidden layers.

the metrics (varying which one was the lowest) specially with 8 layers and forward. This worsening of the metrics for high numbers of neurons might have been due to overfitting [15], because the MLP could have been excessively customized for the training dataset and have not performed well in unknown data. In this analysis, the sensitivity had values higher than the specificity for some numbers of neurons, and some other times was the other way around.

In order to estimate the performance of each explanation type, Figure 9 shows the execution time of the three proposed HAI explanation techniques for different numbers of neurons per hidden layer, with a MLP of two hidden layers, trained with 40 meals associated with the corresponding outputs. In particular, we measured all the possible number of neurons from 1 to 200. In order to be able to measure the necessary short times (in the range of microseconds) and to obtaining representative results, each measurement of time was taken by repeating each explanation method 1,000 consecutive times, and calculating the average execution time of each technique by dividing the total time by the number of repetitions.

As one can observe, the most-weighted-path is the technique that obtained the quickest explanations, in this configuration in the range from 1 to 200 neurons. In the MLPs up to 200 neurons per hidden layer, the maximum execution time was 109 μs. It increased the execution time with the number of neurons but relatively slow, compared to the most-weighted-combination explanation. This latter explanation type increased linearly in respect to number of neurons but more steeply than the former. The maximum-frequency-difference explanation type for each case had

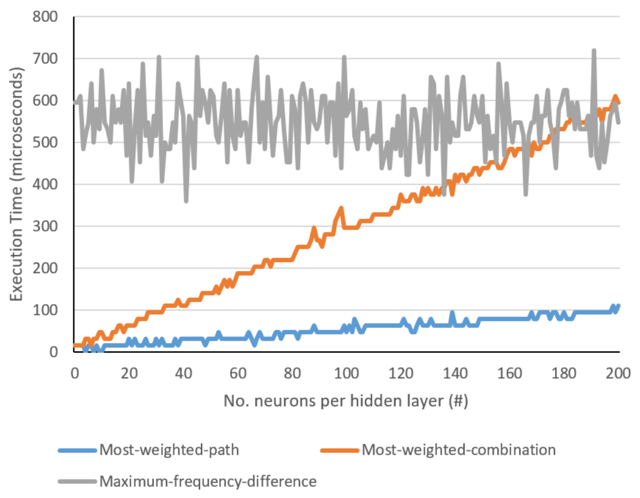


FIGURE 9. Processor execution time of the three HAI explanation approaches for different numbers of neurons per hidden layer.

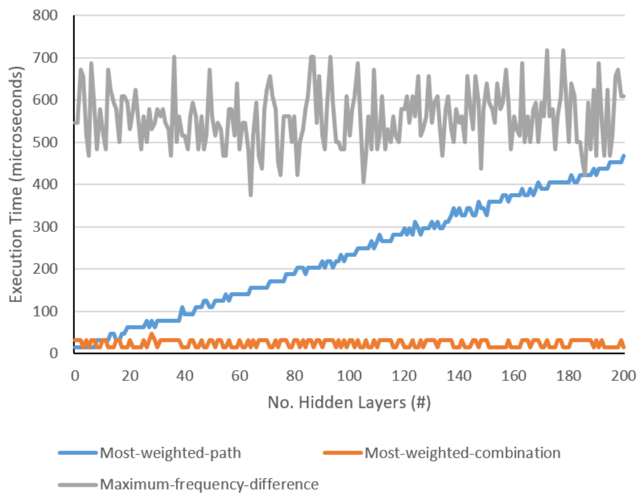


FIGURE 10. Processor execution time of the three HAI explanation approaches for different numbers of hidden layers.

longer responses for most of the range of neurons from 1 to 200 compared with the other two ones, but had the advantage of not increasing in respect to the number of neurons. This constant computational cost makes it the most robust for MLPs with very high numbers of neurons per hidden layer.

Moreover, we analysed the execution times using from 1 to 200 hidden layers, with a fixed amount of five neurons per hidden layer. We used all the same parameters as in the previous analysis of execution time. Figure 10 shows the execution times of the three proposed HAI explanation techniques. The execution times of both the most-weighted-combination and the maximum-frequency-difference explanations did not depend on the number of layers. The most-weighted combination was quite faster than maximum-frequency difference. The execution time of the most-weighted-path increased linearly with the number of hidden layers, although the absolute value was low (469 μs) even for 200 hidden layers.

Furthermore, we also measured the execution times using different sizes of training datasets going from 20 to 4,000 cases, increasing 20 cases in each step. In all the remaining parameters we used the same values as the ones indicated for the case study in section IV. Figure 11 shows the execution times for the three proposed HAI techniques.

One can observe that the execution time of most-weighted-path and most-weighted-combination explanations did not increase with the size of the training datasets. This makes sense as both methods analyse the learned weights of the MLP once this NN has already been trained. However, the execution time of the maximum-frequency-difference explanation increases with the training dataset size. This is reasonable as it analysed the cases of the used training dataset for giving explanations to specific estimations. This increase was not really steep although the execution time had a high range of variation (a SD=243μs for an average of 845μs). The app still kept low absolute execution times (the maximum was only 1.50 ms) for up to 4,000 cases, maybe because the app used MariaDB DBMS, and DBMSs

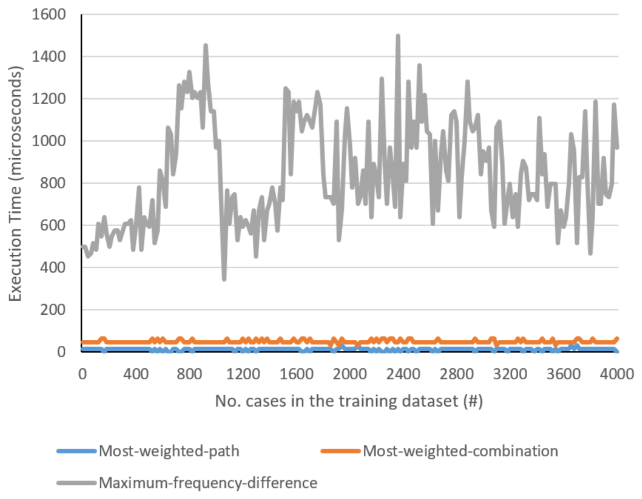


FIGURE 11. Processor execution time of the three HAI explanation approaches using different training dataset sizes.

generally include optimizations for ameliorating the negative performance impact of large datasets.

We also compared the execution times of the most-weighted-path and the most-weighted combination explanations with the time in training the MLP with backpropagation, so we can evaluate whether how much our HAI techniques increase the time after the training process. For the measuring backpropagation, we only used 10 repetitions for calculating the average time, as these times had a complete different order of magnitude. Figure 12 shows the execution of the configuration of the MLP introduced in the previous section, but changing the number of training cases from 20 to 300, in steps of 20 cases.

The execution time of our HAI algorithms were negligible in comparison to the backpropagation time. The sum of both explanation times were only the $6.77*10^{-7}\%$ of the execution time of backpropagation in average. The most-weighted-path

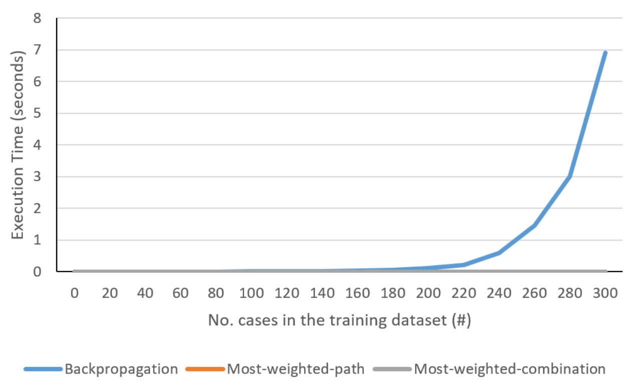


FIGURE 12. Processor execution time of the three HAI explanation approaches using different training dataset sizes.

and most-weighted path times were visually overlapped in the x-axis because of this difference of magnitude.

VI. DISCUSSION

The current work has presented three novel HAI auto-generation explanation techniques in the context of IoT for explaining the behaviour of a MLP. These techniques cover both (a) the explanation of the learned model in general with the concatenation of two different explanations types describing respectively the most relevant IoT sensor input and the most relevant combination of two sensor inputs, and (b) the explanation of each single prediction based on the most discriminative shared IoT input according to the training dataset. In the illustrated case study with a smart IoT kitchen, all the six explanations about the model made sense (presented in three pairs). In the 60 explanations for specific predictions, 58 of these made sense. In total 64 out of 66 explanations made sense, which represents a 96.97% of reasonable explanations. Therefore, the proposed HAI approach automatically generated mainly reasonable explanations in natural language, fulfilling one of the main goals of HAI established in the literature [6], which is the justification of AI decisions.

It is worth highlighting that the two non-reasonable auto-generated explanations were explaining predictions that did not match the correct classification. Thus, the proposed HAI approach may be useful so that human users can have hints about non-reliable estimations when the explanations do not make sense. This can be considered a step forward towards another property highlighted as desirable and challenging in the XAI literature [16], which is detection of non-reliable AI estimations by properly generating explanations so humans can revise them.

The execution times of the proposed techniques are considered efficient as each of the most-weighted-path, most-weighted-combinations and the maximum-frequency-difference explanation techniques only depend on one variable from respectively number of layers, number of neurons, and the number of training cases. However, the approximate backpropagation training depends on all these variables. The processing time for generating explanations about the whole training model was normally much less than the time necessary for the backpropagation training (a percentage of $6.77 * 10^{-7}\%$ in average in our experiments), and consequently the former time was unnoticed by the user. Regarding the time execution for generating the explanations may also be unnoticed, as even for large training datasets (e.g. up to 4,000 meals for training depression detection), the maximum execution time was 1.5 ms, which is much lower than the minimum duration time for being perceived by human eye (i.e. 200 ms) according to the common standards of HCI field [17]. Thus, users will perceive the generation of these explanations as instantaneous. In all the tests performed with the other explanations based on the analysis of the weights of the MLP for explaining the whole trained model, the executions times were 0.61 ms or lower considering MLPs up to 200 neurons in two hidden layers, and MLPs of 200 layers with five neurons per hidden layers. Thus, in this context, all the explanation generations were perceived as instantaneous and are far away from the human-noticeable duration time.

We observed that the combinations obtained by the most-weighted-combination technique were not necessarily related with the highest value but with the relevance in predicting any value. Thus, we changed the generated explanation to highlight this fact, and repeated the experiments, so that in the explanation about the learned model made sense. This fact occurred because if a relevant combination was detected based on the analysis of dendrites weights between layers 0 and 2 (i.e. from the input to second neuron layers), then this values could be used for detecting either true or false outputs (i.e. depressed or normal in our case study), depending on the signs and values of the weights between layer 2 and the output.

The classification metrics in average of 81.67% was relatively high if compared to the recent works in the field of emotion detection, such as the ones about emotion detection from touch interactions when typing on smartphones [18] and emotion detection from body poses [19]. Notice that in the context of our case study, emotions are very subjective as well as the opinions about what meals are usually eaten or recommended in certain emotional states.

It is worth noting that the proposed maximum-frequency-difference explanation used the maximum difference of frequency percentage considering only the active inputs. Another possible explanation could have been to also consider the negative difference of frequencies in the non-activated inputs. This could have provided explanations such as that the system has selected a certain output because the system did not detect a certain input and this input was a certain percentage more frequent in the opposite output. In the particular case study, a possible explanation could be "The system predicted you were normal because among other reasons you did not eat sugar, and depressed people eat sugar 32% more frequently than people in a normal emotional state." We avoided this kind of explanations because of the known difficulty of understanding when using multiple negations [20], but we will explore this in the future to assess whether this kind of explanations can be useful and easy-to-understand by users.

We also performed some initial tests using the approach of selecting different combinations of input and graphically representing the output of the MLP. We found that this was intuitive in two numerical inputs and one numeric output. For a higher number of inputs, the system could have obtained projections. However, we did not find this easy-to-understand for users, and for a high number of inputs (e.g. 15 like in the presented case study), the number of possible projections would have been too high. We plan to further experiment this option by for instance automatically selecting the most relevant projections with a reasonable computational cost, for example using the two IoT sensor inputs of the most-weighted-combination explanation.

In the proposed most-weighted-combination explanation, we only considered combinations of two inputs. In the future, we plan to also highlight combinations of more than two inputs, considering for example a threshold weight in the dendrites connected to the IoT sensor inputs and a higher limit of possible inputs. We think that even this limit can be higher than two, it should not be too high to avoid that explanations are too difficult to be understood by end users.

In the case study of this work, we have mainly used a MLP for classification of certain binary inputs coming from IoT sensors. However, this approach could probably be applied for inputs with more states or a continuous range of values. In addition, MLP could also be used as regressor. We plan to further evaluate all these scenarios with future case studies.

VII. CONCLUSION AND FUTURE WORK

The current work has introduced three novel techniques for auto-generating HAI explanations from MLPs in IoT environments, and has presented their corresponding algorithms so that other researchers can reproduce these experiments or enhance them. Two of these techniques focus on explaining the whole learned model in general, by analysing weights of artificial neurons. Another technique explains each prediction by analysing similar cases used in the training of the MLP. In the case study about a smart IoT kitchen simulator trained and validated with different meals associated with either depression or healthy meals, the proposed approach generated explanations, 97.0% of which made sense according to a human experimenter. The two explanations that did not make sense were explaining two wrong predictions, so these explanations could help humans in detecting non-reliable predictions. The computational cost of each proposed HAI explanation technique depended only from respectively one of the three variables (1) number of neurons per hidden layer, (2) number of hidden layers, and (3) number of training cases. Each proposed technique did not depend on the other two variables. The time in generating these explanations will keep unnoticeable, since its cost is lower than the one for training the MLP with backpropagation, which depends on all these three variables. In addition, the execution time was relatively low for the common configurations of the MLP used in the presented case study (1.5 ms or lower), which is much lower than human eye can notice (200 ms).

In order to fully achieve a HAI that can actually makes IoT further trustworthy, we plan to develop more HAI auto-generation explanation techniques based on other AI techniques such as SVM, KNN and random forest. We also plan to further assess the proposed HAI explanation techniques in other IoT systems and evaluate the explanations with large groups of users to determine whether the explanations are actually easy to understand. We also plan to implement the proposed approach in real smart cupboards, and test it with cancer survivors at their homes to track whether they are getting into a depression to warn contact people, like familiars, doctors or friends.

REFERENCES

[1] O. Bello and S. Zeadally, "Intelligent device-to-device communication in the Internet of Things," *IEEE Syst. J.*, vol. 10, no. 3, pp. 1172–1182, Sep. 2016.

[2] I. García-Magariño, F. González-Landero, R. Amariglio, and J. Lloret, "Collaboration of smart IoT devices exemplified with smart cupboards," *IEEE Access*, vol. 7, pp. 9881–9892, 2019.

[3] I. García-Magariño, R. Lacuesta, and J. Lloret, "ABS-SmartComAgri: An agent-based simulator of smart communication protocols in wireless sensor networks for debugging in precision agriculture," *Sensors*, vol. 18, no. 4, p. 998, 2018.

[4] K. Jha, A. Doshi, P. Patel, and M. Shah, "A comprehensive review on automation in agriculture using artificial intelligence," *Artif. Intell. Agricult.*, vol. 2, pp. 1–12, Jun. 2019.

[5] N. Scarpato, A. Pieroni, L. D. Nunzio, and F. Fallucchi, "E-health-IoT universe: A review," *Int. J. Adv. Sci., Eng. Inf. Technol.*, vol. 7, no. 6, pp. 2328–2336, 2017.

[6] S. Amershi, D. Weld, M. Vorvoreanu, A. Fourney, B. Nushi, P. Collisson, J. Suh, S. Iqbal, P. N. Bennett, K. Inkpen, J. Teevan, R. Kikin-Gil, and E. Horvitz, "Guidelines for human-AI interaction," in *Proc. CHI Conf. Hum. Factors Comput. Syst.*, 2019, pp. 1–3.

[7] N. K. Nawandar and V. R. Satpute, "IoT based low cost and intelligent module for smart irrigation system," *Comput. Electron. Agricult.*, vol. 162, pp. 979–990, Jul. 2019.

[8] R. Nawaratne, D. Alahakoon, D. De Silva, P. Chhetri, and N. Chilamkurti, "Self-evolving intelligent algorithms for facilitating data interoperability in IoT environments," *Future Gener. Comput. Syst.*, vol. 86, pp. 421–432, Sep. 2018.

[9] B. Yong, Z. Xu, X. Wang, L. Cheng, X. Li, X. Wu, and Q. Zhou, "IoT-based intelligent fitness system," *J. Parallel Distrib. Comput.*, vol. 118, pp. 14–21, Aug. 2018.

[10] A. Ahmad, S. Cuomo, W. Wu, and G. Jeon, "Intelligent algorithms and standards for interoperability in Internet of Things," *Future Gener. Comput. Syst.*, vol. 92, pp. 1187–1191, Mar. 2019.

[11] L. Ding, "Human knowledge in constructing AI systems—Neural logic networks approach towards an explainable AI," *Procedia Comput. Sci.*, vol. 126, pp. 1561–1570, Jan. 2018.

[12] A. Adadi and M. Berrada, "Peeking inside the black-box: A survey on explainable artificial intelligence (XAI)," *IEEE Access*, vol. 6, pp. 52138–52160, 2018.

[13] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Oct. 2011.

[14] L. Prechelt, "Automatic early stopping using cross validation: Quantifying the criteria," *Neural Netw.*, vol. 11, no. 4, pp. 761–767, 1998.

[15] J. Rynkiewicz, "General bound of overfitting for MLP regression models," *Neurocomputing*, vol. 90, pp. 106–110, Aug. 2012.

[16] B. Mittelstadt, C. Russell, and S. Wachter, "Explaining explanations in AI," in *Proc. Conf. Fairness, Accountab., Transparency*, 2019, pp. 279–288.

[17] C. S. Miller, "Relating theory to actual results in computer science and human-computer interaction," *Comput. Sci. Educ.*, vol. 13, no. 3, pp. 227–240, 2003.

[18] S. Ghosh, K. Hiware, N. Ganguly, B. Mitra, and P. De, "Emotion detection from touch interactions during text entry on smartphones," *Int. J. Hum.-Comput. Stud.*, vol. 130, pp. 47–57, Oct. 2019.

[19] I. García-Magariño, E. Cerezo, I. Plaza, and L. Chittaro, "A mobile application to report and detect 3D body emotional poses," *Expert Syst. Appl.*, vol. 122, pp. 207–216, May 2019.

[20] F. Blanchette and M. Nadeu, "Prosody and the meanings of English negative indefinites," *J. Pragmatics*, vol. 129, pp. 123–139, May 2018.

IVÁN GARCÍA-MAGARIÑO received the Ph.D. degree in computer science engineering from the Complutense University of Madrid, in 2009. He is currently a Lecturer and a Contributor to the GRASIA Research Group, Complutense University of Madrid. Prior to commencing this position, in 2018, he was a Ph.D. Assistant Professor with the University of Zaragoza, from 2014 to 2018, and a Lecturer with Madrid Open University, from 2010 to 2014. He actively collaborates with the City, University of London (UK) on Human-centric AI and security-privacy projects, the EduQTech Research Group, University of Zaragoza, on m-health projects, the HCI Laboratory, University of Udine, Italy, on human–computer interaction projects, the Technological University of Dublin, Ireland, on datamining projects, and the Multicultural Alzheimer Prevention Program (MAPP) of the Massachusetts General Hospital and Harvard University, USA, on m-health projects. Among journals, book chapters, conferences, and workshops, he has over 125 publications (54 in journals with ISI Thomson JCR). His main research interests include artificial intelligence (AI), human-centric AI, agent-based simulators, multi-agent systems, and cryptocurrency/blockchain, all of these applied to different fields, such as the IoT, security, and health. He was a recipient of an FPI Scholarship, from 2006 to 2010. He has been a Guest Editor in several special issues in journals with impact factor. His website is http://grasia.fdi.ucm.es/ivan/.

RAJARAJAN MUTTUKRISHNAN received the B.Eng. and Ph.D. degrees from the City, University London, in 1994 and 1999, respectively, where he has been a Research Fellow, since 1999. In August 2000, he moved to Logica as a Telecommunication Consultant. After a few years in the industry, he is currently a Professor of security engineering. He is also the Program Director for the Engineering with Management and Entrepreneurship Program. He is also a member of IET and an Associate Member of the Institute of Information Security Professionals (IISP) and a member of the Technical Program Committees for PIERS 2010, eHealth 2010, SECURECOM2011, TrustBus 2011, Digital Economy 2012, IFIPTM 2012, and IFIP SEC 2012. He is also the General Chair of SECURECOMM 2011, London. He also sits on the Editorial Boards of *Wireless Networks* (Springer/ACM), *Health Policy and Technology* (Elsevier), and *Information Management and Computer Security* (Emerald).

JAIME LLORET (M'07–SM'10) received the B.Sc. and M.Sc. degrees in physics and electronic engineering, and the Ph.D. (Dr. Ing.) degree in telecommunication engineering, in 1997, 2003, and 2006, respectively. He worked as a Network Designer and an Administrator in several enterprises. He is currently an Associate Professor with the Polytechnic University of Valencia. He is also a Cisco Certified Network Professional Instructor. He is also the Head of the "Active and collaborative techniques and use of technologic resources in the education (EITACURTE)" Innovation Group. He is also the Director of the University Diploma "Redes y Comunicaciones de Ordenadores." He has been the Director of the University Master "Digital Post Production" for the term 2012–2016. He has authored 22 book chapters and has more than 480 research articles published in national and international conferences and international journals (more than 220 with ISI Thomson JCR). He is also a Senior Fellow of ACM and a Fellow of IARIA. He is also the Chair of the Integrated Management Coastal Research Institute (IGIC). He was the Vice-Chair for the Europe/Africa Region of Cognitive Networks Technical Committee of the IEEE Communications Society for the term 2010–2012 and the Vice-Chair of the Internet Technical Committee of the IEEE Communications Society and Internet Society for the term 2011–2013. He has been the Internet Technical Committee Chair of the IEEE Communications Society and the Internet Society for the term 2013–2015. He is also an IARIA Journals Board Chair (eight journals). He is also the Chair of the Working Group of the Standard IEEE 1907.1. He has been the General Chair (or Co-Chair) of 45 international workshops and conferences. He has been a Co-Editor of 40 conference proceedings and a Guest Editor of several international books and journals. He is also the Editor-in-Chief of *Ad Hoc and Sensor Wireless Networks* (with ISI Thomson Impact Factor), *Network Protocols and Algorithms*, and *International Journal of Multimedia Communications*. He is also an Associate Editor-in-Chief of *Sensors* in the Sensor Networks Section and an Advisory Board Member of the *International Journal of Distributed Sensor Networks* (both with ISI Thomson Impact factor). He is (or has been) an Associate Editor of 46 international journals (16 of them with ISI Thomson Impact Factor). He has been involved in more than 450 program committees of international conferences and more than 150 organizations and steering committees. He has led many local, regional, national, and European projects. Since 2016, he has been a Spanish Researcher with the highest h-index in the *Telecommunications* journal list according to Clarivate Analytics Ranking.