



**HAL**  
open science

## Arcades: A deep model for adaptive decision making in voice controlled smart-home

Alexis Brenon, François Portet, Michel Vacher

► **To cite this version:**

Alexis Brenon, François Portet, Michel Vacher. Arcades: A deep model for adaptive decision making in voice controlled smart-home. *Pervasive and Mobile Computing*, 2018, 49, pp.92-110. 10.1016/j.pmcj.2018.06.011 . hal-01829401

**HAL Id: hal-01829401**

**<https://hal.science/hal-01829401>**

Submitted on 4 Jul 2018

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# ARCADES: A deep model for adaptive decision making in voice controlled smart-home

Alexis BRENON<sup>a</sup>, François PORTET<sup>a,\*</sup>, Michel VACHER<sup>a</sup>

<sup>a</sup>*Univ. Grenoble Alpes, CNRS, LIG, F-38000 Grenoble France*

---

## Abstract

In a voice controlled smart-home, a controller must respond not only to user's requests but also according to the interaction context. This paper describes ARCADES, a system which uses deep reinforcement learning to extract context from a graphical representation of home automation system and to update continuously its behavior to the user's one. This system is robust to changes in the environment (sensor breakdown or addition) through its graphical representation (scale well) and the reinforcement mechanism (adapt well).

The experiments on realistic data demonstrate that this method promises to reach long life context-aware control of smart-home.

*Keywords:* Smart-home, Decision system, Context-aware, Reinforcement learning, Deep learning

---

---

\*Corresponding author

*Email addresses:* alexis.brenon@univ-grenoble-alpes.fr (Alexis BRENON), francois.portet@univ-grenoble-alpes.fr (François PORTET), michel.vacher@univ-grenoble-alpes.fr (Michel VACHER)

## 1. Introduction

One of the applications in which Ubiquitous Computing [56] generates great expectations is *smart-home*. Smart-home is an application domain which brings together home automation and *ambient intelligence* to ease life of dwellers and to provide support to people in loss of autonomy. The development of smart-homes is not only a cultural and technological evolution but is also recognized as one way to address the challenges created by an aging population in developed countries [42]. If home automation is concerned with sensing (sensors, actuators, middle-ware) and low-level automation (heating control, lighting control), *Ambient Intelligence* should provide perception and reasoning capabilities into the smart-home ecosystem.

However, although the development of smart-homes is supported by a large amount of research and industrial projects, it has not reached a large public since many challenges are still to be addressed. One of the main challenges is due to the complexity of setting up the smart-home system in case of new situations (devices, house, dwellers, after an accident, etc.). Furthermore, smart-homes often need to perform inferences from data in order to make decision so as to provide services that enhance users' experience and control over their environment [3]. To make this enriched control possible, these systems perceive their environment and decide which actions to apply to the environment. This decision can be made after some specific events such as a user's request, a scheduled event (e.g., programmed task) or a detected risky situation. This perception is not only useful to trigger a decision but also to adapt the decision making to the current circumstances in which such decision must be performed. These circumstances are called *the context* and systems that explicitly take the context into account, are said *context-aware* [35]. However, defining explicitly what pieces of information is playing an important role in the context of a decision is a difficult and tedious task. Indeed, if a general definition of context exists in the field [14], the large amount of existing models [6, 21] demonstrates not only the importance of this notion for research but also the lack of a unified way of modelling context in intelligent environments.

Even though a model could be defined for a specific home, successful interaction must match users' preferences. To do so, explicit user configuration is a way to handle preferences but this have been proven to reduce the acceptance of the system [61]. More subtle ways of adapting system to users' preferences must be found. Moreover, as shown by Zaidenberg et al. [61], users ask for a way to monitor the system to better understand what it does and why. Indeed, smart-home systems must let the user keep control of the house even when she is in loss of autonomy [43]. All these factors make the development of an end-to-end smart-home system challenging.

In this paper, we propose a new system called ARCADES (Adaptive Reinforced Context-Aware Deep Decision System) to perform adaptive context-aware decision for voice command in a smart home that does not use an explicitly defined context. Rather, the context information is directly learned from data using deep neural network [47, 28]. Indeed, one of the main advantages of Deep Learning (DL) is to learn attributes from raw data at the same time as the classification model is trained. This ability of DL has been demonstrated in numerous classification tasks reaching superhuman accuracy in the case of image classification [28].

Another aspect of context awareness is to be able to capture a changing environment over time. Indeed, sensors, activity places, dwellers, etc. can change and any system that targets a lifelong support must be able to take these changes into account. [Scenario 1](#) illustrates this need. In this simple scenario, it is impossible to know how to act without considering the context since the two voice commands are semantically the same. Here the activity or the location influence the choice of the lamp to light. Changes in the daily living routine (changing a lamp of place, moving the table, etc) must be accounted for in the configuration of the system. If static rule based systems are employed, any change must be impacted in the rule based system, making this impossible to modify by non-technical person such as the elderly users. To adapt continuously to a changing environment, the decision model we present in this paper uses reinforcement learning to update its decision policy.

**Scenario 1** *The inhabitant is washing up dishes and utters the voice command "Turn on the light". The lamp above the sink is lighted. Then she goes to the kitchen table and utters "Give me some light". The lamp above the table is lighted.*

Finally, to make the smart home system easy to interact with we rely on speech interaction as well as on the sensor data capturing the interaction of the user with her own home. This makes the set-up accessible since the user does not have to learn a complex jargon or to deeply understand the internal mechanisms of the system. Relying on machine learning paradigms, ARCADES is able to build a model relevant for the given task, and then to adapt itself, in a non-intrusive manner. Moreover, we advocate the use of a new graphical approach to represent the environment state that brings two advantages: it handles multiple heterogeneous sensor data (making it resilient to smart-home evolution) and it allows users to check what the system perceives (cf. [61]). Thus, the work presented in this paper brings the following contributions.

- A system which represents the decision-making task as a reinforcement learning one, to adapt a decision making to the user’s behaviour and her environment in an on-line setting (i.e., data are processed only with a limited history of past and no information about the future) in which the task is to perform voice command execution.
- A unified representation of the smart-home heterogeneous data sources, namely image, which is readable by machine and human ; hence, the model does not depend on intermediate representation layers but only on the input data. This is a strong strength of our approach as well as an original aspect of the research presented here (we are not aware of any work using this original data representation in this field).
- A deep learning model able to build and extract relevant contextual features from (semi-)raw sensors input (i.e. images) as well as solving the reinforcement learning task.
- An evaluation of the system on a realistic smart-home data set available to the community.

The paper is organised as follows. The next section ([section 2](#)) summarises the related work. The method is detailed in [section 3](#) which describes the overall approach and the data representation. In [section 4](#) we present an experiment to train and evaluate ARCADES on realistic data. Complementary experiments are summed up in [section 5](#). Finally, the results and some possible improvements are discussed in [section 6](#).

## 2. Related works

Automatic adaptive decision-making in smart-homes has received far less attention than more widespread research on context or situation recognition. However, it had recently regained interest, particularly in the domain of energy management [31, 5]. In the literature, a decision-making system depends on two main abilities: (1) the ability to represent knowledge perceived from the environment and (2) the ability to make decision from this perception given a specific user and situation.

With respect to the first ability, research on context aware systems had generated a vast amount of knowledge representations. Logical approaches are generally applied for representation in pervasive environments because they can easily model relations among entities and perform automatic inference. For instance, Mileo et al. [36] applied a logic programming approach, Answer Set Programming, in order to implement a system of prediction of risky situations. Event calculus, a method to model situations using first order logic, has also been applied in the development of smart-homes [11, 25].

The main trend for knowledge representation in smart environments has become ontology, in particular those based on Description Logic (DL). The Ontology Web Language (OWL) is the main implementation of DL, and it has been largely applied in pervasive environments [46, 32]. Liao and Tu [32] employed an ontology to organize temporal events and the concepts about the smart-home. Context-aware systems have also been developed using ontologies to model the current context for mobile applications [2, 59]. Fuzzy ontologies also received attention because of their ability to model imprecision and allow inference with vague concepts such as “hot weather”. Rodríguez et al. [45] presented an application of fuzzy ontologies for activity recognition in pervasive environments. Another important contribution to make ontologies capable of dealing with uncertainty is the use of probabilistic models in ontology inference. A notable application of such approach has been presented by Helaoui et al. [18]. In Chahuara et al. [9] a Markov Logic Network together with an ontology in Description Logic (DL) is employed to perform context recognition. However, the diversity of approaches also shows the lack of consensus regarding the different concepts to be considered in a smart-home and their organization. Hence, there is a need for alternative approaches to handle information being inferred from the sensor network of the smart-homes.

In all of these approaches, context modelling is considered from a top-down perspective, where elements of the context are identified by expertise. In this paper, we propose a different approach: a data-driven constitution of the context where all the sensor data are unified into a common representation: an image. The closest work to this approach is the recent one by Singh et al. [48] which transforms a pressure sensor matrix image to perform gait identification using deep neural network. However, in their work the data were already sequences of 2-dimensional values while in the case of smart home we must deal with a set of monodimensional sensor data of variable type (Boolean, categorical, numerical) whose position is only loosely known and changing.

Regarding the decision-making process, several logic-based approaches are presented in the literature. Moore et al. [38] developed a system that exploit a set of fuzzy rules in order to find the most appropriate action under a certain condition given by the context. Similarly, Kofler et al. [26] and Gómez-Romero et al. [17] used Description Logic to define the behavior of a context-aware system which models context elements (activities) in an ontology. There are several works applying ECA (Event-Condition-Action) rules based systems into pervasive environments [30, 58]. However, in these proposals the system is set *a priori* to execute an action given a specific configuration (the condition) and consequently the system does not adapt its behavior to a changing situation. Bayesian networks, for instance, are among the most important methods for decision-making, as exemplified by Lee and Cho [29]. They have modeled uncertainty on context-aware system by means of Bayesian networks but the method does not include formal elements of decision-making. Influence Diagrams (ID, [20]) is a method based on Bayesian networks that includes special variables to appropriately model the decision process. Some research work on context aware systems [13, 41] have relied on ID in order to model the decision process, treating the uncertainty and measuring the expected utility of possible actions. However, the expressiveness of this probabilistic approach is limited since only propositional variables and conditional dependencies among variables can be represented in the model, besides the fact that it is less human readable than logical approaches. In Chahuara et al. [10] a Markov Logic Network (MLN) is proposed to perform automatic decision-making in smart-home. MLN

make it possible to benefit from a formal representation of the decision process using first order logic while modeling uncertainty by weighting the logic formulae. The other advantage is that weights can be learned from data. The approach has been tested in a realistic setting and showed the strength of the approach. However, it does not include mechanism to adapt the decision model to new situations. Furthermore, if part of the model can be learned from data, the logic part is acquired by human expertise which has the same drawback as the logic based knowledge representations in smart-home.

To tackle the problem of adaptation, some researches in smart-home have designed the decision problem as a reinforced decision-making problem. The most famous of these works is *The Neural Network House* [39]. It is a house equipped with more than 75 sensors and actuators monitoring the environmental conditions — such as temperature, light and sound levels, motions, door and windows openings — and controlling some devices of the house such as heaters or lights. The devices were controlled by the ACHE system [40], which commanded the lights in order to reduce the overall energy consumption without impairing the inhabitant comfort. Reinforcement Learning (RL) is the base of the ACHE system. A state estimator builds a high-level environmental state from the sensor data. This state contains the current activity of the user and the level of natural light at the user’s location. Then, a controller makes a decision about which light to act on, and decides on the intensity level to set. To evaluate this decision, two kinds of cost are defined and returned to the controller for its learning. On the one hand, the energy cost, directly indexed on the actual energy cost of electricity in the area of the smart-home. On the other hand, the comfort cost, which is induced when the inhabitant have to modify the decision made by the system. Although this project showed promising results, it seems to be no longer active but has given rise to a number of researches by other groups [60, 23] which showed that RL allows some flexibility that traditional machine learning paradigms do not. Nevertheless, if RL is particularly suitable for restricted environment (with a few possible states and actions), it poorly scales to large real cases sets.

This drawback, is mainly caused by the classical implementations used for reinforcement learning tasks, based on an associative array whose size depends on the number of variables and their domain values to represent the state of the environment. A way to solve this, is to change the data structure, relying for example on a deep neural network [28, 47]. A notable example of *deep reinforcement learning* for decision-making is provided by Mnih et al. [37] who proposed a new approach to train a deep neural network using reinforcement learning. The proposed model, the Deep Q-Network (DQN), has been applied to a decision-making task regarding classic Atari2600 games. Roughly speaking, given a pre-processed image frame from the Atari display, the model chooses the best action out of 18 (17 combinations of a joystick plus 1 no action) to maximize its reward based on the game score. A convolutional neural network (CNN) processes the frames, while a fully connected network approximates the value of the states of the RL problem. After few millions training iterations, the model is compared to random and human decision-making and shows that the DQN reaches or surpasses human performances in the majority of tasks. This work proves that RL and a deep convolutional network can be combined. Thus, RL can be used in situations approaching real-world complexity. This inspired us to use a similar agent to make context-aware decision in a smart-home.

Table 1 lists the main attributes of the papers considered in this state of the art. It shows the variety of models that have been applied for decision-making in contrast to the lack of experimental real-time tests in real environments. Even when several approaches have been proposed to handle knowledge representation, inference or adaption, most of these works have been applied to a particular problem of context awareness, such as activity recognition, and did not consider the problem as a decision-making one. In this work, we propose an approach using deep-learning to learn the features/concepts that comes into play during the decision at the same time as the decision model. This learning is performed using the reinforcement learning paradigm so that on-line adaptation is made possible. Furthermore, we propose an original knowledge presentation, raw image, to benefit from the recent improvement in image processing using deep-learning. The remaining of this paper presents the method and its application to decision-making in a smart-home.

Table 1: Summary of the studies related to context representation and decision-making in smart homes

Publication	Inferred contextual information	Decision-making method	Data representation	Voice interface				Use case
				Continuous adapt.	On-line	Smart-home	Multi-room	
<b>No decision-making</b>								
Mileo et al. (36)	Location, identity, objects	–	Answer set programming	✗	✗	✗	–	Synthetic data
Katzouris et al. (25)	Activity	–	Logical	✗	✗	✗	–	Synthetic data
Rodriguez et al. (45)	Activity	–	Fuzzy ontology	✗	✓	✗	–	Synthetic data
Heloui et al. (18)	Activity	–	Probabilistic ontology	✗	✓	✗	–	Synthetic data
<b>Decision-making</b>								
Liao and Tu (32)	Predefined situations	Dangerosity level	Ontology	✗	✗	✗	✗	Synthetic data
Kofler et al. (26)	Predefined situations	Ontology reasoning	Ontology	✗	✗	✗	–	None
Gómez-Romero et al. (17)	Activity, predefined situations	Situation recognition	Ontology	✗	✗	✓	✗	Video sequence from one person
Nishiyama et al. (41)	Predefined situations	Influence diagram	Bayesian model	✗	✗	✗	–	Synthetic data
Chahuaara et al. (10)	Location, activity, agitation, predefined situations	MLN based influence diagram	OWL2 and MLN	✓	✗	✓	✓	37 naive users
<b>Reinforcement</b>								
Mozzer (40)	Location, activity	Reinforcement learning	Ontology	✗	✓	✓	✓	Daily life use
Karami et al. (23)	Activity	Reinforcement learning	Ontology	✗	✓	✗	✓	Synthetic data
Muñh et al. (37)	None	Deep reinforcement learning	Graphical	✗	✓	✓	✗	Game simulation
<b>This study</b>	None	Deep reinforcement learning	Graphical	✓	✓	✓	✓	15 naive users

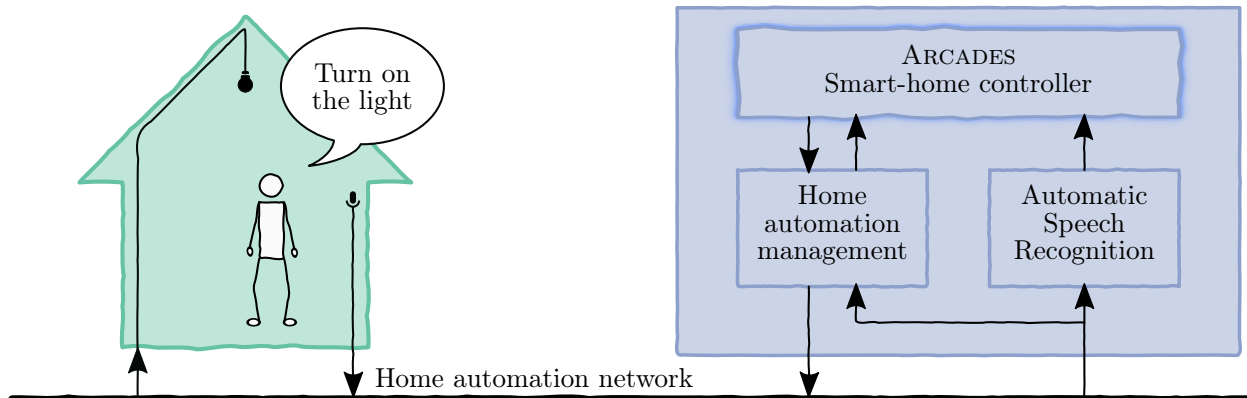


Figure 1: A typical smart-home system equipped with a voice-user interface: use case of a user asking to turn on the light.

### 3. Methods

Typical smart-homes considered in the study are the ones that permit voice based interaction. Voice-User Interfaces (VUIs) in domestic environments recently gained interest in the speech processing community. The rising number of smart-home projects that consider Automatic Speech Recognition (ASR) in their design [22, 15, 27, 12] exemplifies this trend. Adapted to homes containing multiple rooms which are fitted with sensors and actuators — such as infra-red presence detectors, consumption meters, multimedia servers, etc. — this kind of interface is particularly suitable to people in loss of autonomy [43, 52]. Smart-homes aim at providing daily living context-aware decision using the perception of the situation of the user, as previously illustrated in [Scenario 1](#).

The interactions between the user, her home and the control system is depicted in [Figure 1](#). An underlying home automation network, composed of typical home automation sensors and actuators (switches, lights, blinds, etc.) and several microphones per room allows a controller to interact with the home and the user.

As emphasized in [section 1](#), the main challenges in such a context are to be able to handle heterogeneous information in a unique representation space and to adapt the decision-making to an evolving environment. Thus, the remaining of this section focuses on the description of ARCADES, the Adaptive Reinforced Context-Aware Deep Decision System, and its components. We start with the reinforced adaptation loop ([subsection 3.1](#)), then the data representation ([subsection 3.2](#)) and finally the decision model acquired using deep learning ([subsection 3.3](#)). Last section ([subsection 3.4](#)) explains how all these components interact with each other to allow the system to learn.

#### 3.1. Adaptation of the decision model through reinforcement learning

A Reinforcement Learning (RL) task relies on the Markov Decision Process (MDP) formalism which considers two components: the *environment* and the *agent*. [Figure 2](#) shows the interactions between each component: the agent (here the decision-making system) is fed with the current environment state  $s_t \in \mathcal{S}$ . It chooses an action  $a_t \in \mathcal{A}$  based on this state. The chosen action affects the environment (here the house and the user) whose state changes (from  $s_t$  to  $s_{t+1}$ ) and which generates a reward. This reward  $R(s_t) = r_t$  can be positive if the action is appropriate, negative if the action is not adequate, or null. Based on this reward, the agent adapts its decision policy. In [Scenario 1](#), the voice command and context of the interaction compose the state, while the decision made by the system is the action to execute. Then the user will be able to evaluate/reward this decision by different means not discussed here.

In a fully qualified MDP, dynamic programming [4] is used to compute the optimal policy. However, it cannot be used in real world problem due to missing information about the environment model and the reward function. In that case, *Q-Learning* [55, 49] can be used to compute and update a policy which will converge toward the optimal one.



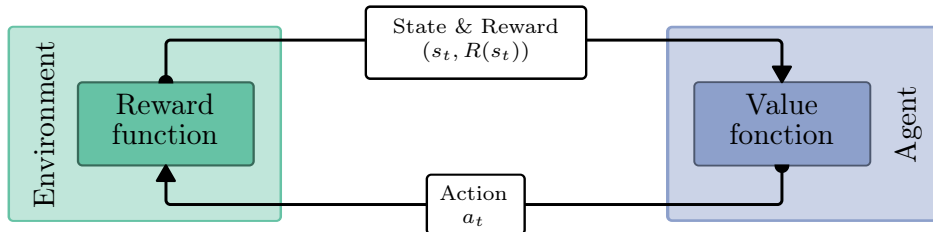


Figure 2: The main components of ARCADES and their interactions

In  $Q$ -Learning, a function, named  $Q$ -function, associates a value, named  $Q$ -value, to each possible pair of state-action, named  $Q$ -state:  $Q : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ , where  $\mathcal{S}$  is the set of possible states of the environment, and  $\mathcal{A}$  is the set of possible actions to perform. Beginning with an initial  $Q$ -function ( $Q_0$ ), it can be updated after each interaction  $\langle s_t, a_t, r_t, s_{t+1} \rangle$ , following Equation 1 (where  $\gamma$  is the discounting factor of the reward function and  $\alpha$  is a learning rate). This updating rule has been proven to converge toward the actual  $Q$ -value of each  $Q$ -state, allowing to compute the optimal policy to solve the task.

$$Q_{k+1}(s_t, a_t) = \alpha(r_t + \gamma \max_a Q_k(s_{t+1}, a)) + (1 - \alpha)Q_k(s_t, a_t) \quad (1)$$

The following sections describe how the states of the environment can be acquired, and the model used for the  $Q$ -function, before presenting how both components interact during the learning process.

### 3.2. A common representation for heterogeneous information: a raw image approach

One challenge in reasoning with raw data is to define a referential space where all kinds of information can be represented and manipulated. Indeed, real smart-home data are multi-modal [53]: from continuous time series, such as the overall water consumption or temperature, to event based semantic information such as voice commands or the state change of a device. Hence, finding a representation that can accommodate low-level sensor events (presence in a room) and high-level events (user current activity) is a problem that has not found a consensual solution. Here, we are proposing the use of a **graphical representation**, projecting any information on a two-dimensional map of the smart-home any time the environment state changes. Then, we will rely on the ability of Convolutional Neural Networks (CNN) to build relevant features from this raw representation. This approach provides some advantages listed below.

1. The images generated can be used by the user to monitor what the system sees or understands.
2. The image size is only loosely linked to the number or type of sensors (adding, removing, or changing some sensors will not change the size of the image), hence images accommodate a variable number of data sources (sensors can come and go) a property that classical input vector machine learning approaches do not *naturally* handle.
3. Images *naturally* convey spatial information (here an *a priory* information about the home room organisation).
4. The high redundancy of sensor values can be efficiently used by deep learning.

Example of generated image for the smart-home considered in the study are provided in Figure 3. In this image, as in the Scenario 1, the user asks to turn on the light while she is washing up dishes. The smart home walls are represented by black lines while icons represent raw sensor data. For instance usage of light is represented by a black lamp on white background while a turned off lamp is a white lamp on dark background. The highest part presents some gauges for electricity and water consumption. The big icon at the bottom left represents the uttered command. Finally, other icons match the last known state of each information provider (sensor or inference system) of the home, using a different icon for different information type. It exists a dedicated icon for various kinds of information such as:

- Binary sensors symbols: presence detector, door/window detector, blind state, light state, etc.;

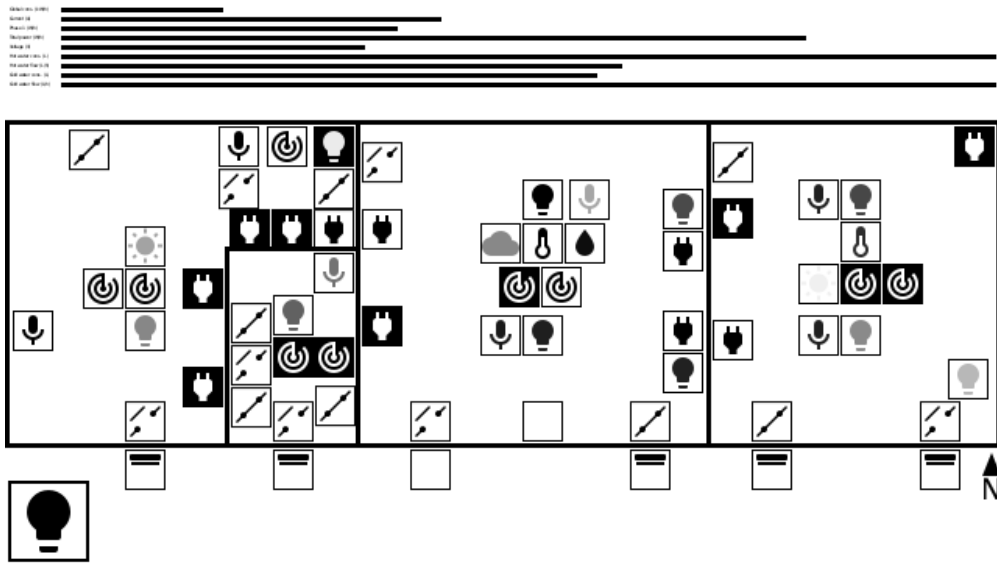


Figure 3: Examples of image generated for the agent input. From left to right : kitchen, bathroom, bedroom and study.

- Continuous sensors symbols: light level, sound level, temperature, humidity;
- Continuous sensors gauges: electricity consumption, water consumption, etc.;
- High level information symbols: uttered command.

The image and icons have been chosen to maximise the contrast and redundancy for the input of a CNN, to be understandable by human and to be easily generated on the fly. In fact, the implementation relies on the SVG<sup>1</sup> technology, to make it possible to update the image on-line to reflect the state of the smart-home.

### 3.3. Decision making model through deep-learning

As explained in subsection 3.1, the decision policy rely on the  $Q$ -function of the agent (the simplest policy being: “Given  $s$ , choose the action with the highest  $Q$ -value”) and we are going to present two ways to represent it.

Classical implementations choose a tabular representation. A two-dimensional array, indexed by states in one dimension and by actions in the other, records the  $Q$ -values for each  $Q$ -state. However, this approach does not scale to big or continuous sets of states or actions and cannot capture redundancy in states.

Another recent implementation relies on a deep neural network to model the  $Q$ -function. This representation has been introduced by Mnih et al. [37] who introduced the Deep Q-Network (DQN), which has been successfully tested on decision-making task related to Atari2600 games (cf. related work section 2). Although, their application was different from our, the voice command problem can be seen as a general problem of decision-making in a changing environment. We thus adapted the approach to smart-home decision-making.

The decision model is based on a Convolutional Neural Network (CNN) to process the images followed by a fully connected neural network to choose the best action to perform. Figure 4 presents the architecture of our  $Q$ -network. While not discussed here, different architectures has been tested, taking into account base architecture proposed by Mnih et al. [37], technical limitation like memory consumption as well as some *a priori* knowledge like the 6 px based grid of the images. Finally, the neural network is composed of four

<sup>1</sup>Scalable Vector Graphics (SVG) is a widely-deployed, royalty-free, XML-based vector image format developed and maintained by the World Wide Web Consortium (W3C) (<https://www.w3.org/Graphics/SVG/>)

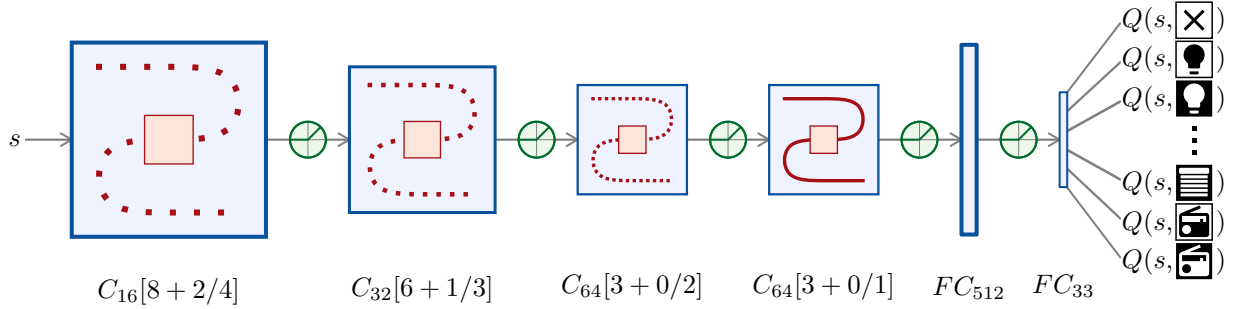


Figure 4: Architecture of the  $Q$ -network.

$C_n[f + p/k]$  denotes a convolution layer (using  $n$  filters of size  $f$ , with a padding of  $p$ , and using a stride of  $k$ ) and  $FC_n$  a fully connected linear layer (of width  $n$ ).

*convolutional layers*, reducing a  $256 \times 256$  input image to features maps of sizes:  $16 \times 64 \times 64$ ,  $32 \times 21 \times 21$ ,  $64 \times 10 \times 10$ , and  $64 \times 8 \times 8$ . The features maps of the last layer are then rearranged in a vector of dimension  $64 \times 8 \times 8 = 4096$ . This vector passes through *fully connected linear layers* of width 512 and 33 which is the number of possible actions. A *rectified linear unit* separates each layer.

Thus, this model provides a way to process the generated images presented in subsection 3.2. After some preprocessing (scaling and reducing the depth of the image), the image is passed forward in the  $Q$ -network. The network extracts relevant features used to classify it in the range of possible actions. It provides, as output, the  $Q$ -values for each action given the input state (and thus can be seen as a function  $Q\text{-network} : [0, 255]^{256 \times 256} \rightarrow \mathbb{R}^{33}$ ). Then, Equation 1 detailed earlier can be used as a loss function to update the weights of the  $Q$ -network, as in typical deep learning approaches.

While the work from Mnih et al. [37] is not the first attempt to use neural network as  $Q$ -function [50], it seems to be the first successful one. This is due to different optimization techniques as the use of a target network, experience replay [33], mini-batch learning or RMSProp gradient optimization [19]. All these techniques have been kept in our implementation and the reader is referred to the original paper [37] for detailed explanations.

### 3.4. Learning method

---

**Algorithm 1** Logic used to allow the agent and the environment to interact.

---

```

Require:  $epoch = 0$ 
while  $epoch < maxEpoch$  do
   $step \leftarrow 0$ 
  repeat
     $step \leftarrow step + 1$ 
     $obs \leftarrow \mathbf{environment.GETOBSERVABLESTATE}$ 
     $\mathbf{agent.INTEGRATEOBSERVATIONS}(obs)$ 
     $action \leftarrow \mathbf{agent.GETACTION}$ 
     $\mathbf{environment.PERFORMACTION}(action)$ 
     $reward \leftarrow \mathbf{environment.GETREWARD}$ 
     $\mathbf{agent.GIVEREWARD}(reward)$ 
  until  $step > maxSteps$ 
   $\mathbf{agent.LEARN}$ 
   $epoch \leftarrow epoch + 1$ 
end while

```

---

The overall learning of the decision-making module (the agent) is decomposed into two phases: a *pre-training phase* (i.e., initialization of the agent), during which the model is learned from scratch and an

*adaptation phase* during which the model is tuned toward the specific inhabitant and environment. In practice, the pre-training phase is performed off-line and is mostly devoted to the learning of the CNN and the subsequent neural network. It is well-known that CNN needs a high amount of data that is why, in our work, we used artificially generated data to feed the CNN with raw input to learn the CNN weights. In the adaptation phase the CNN is not supposed to evolve but the last part of the neural network (i.e., the decision part) is biased towards the adaptation data. This data can be either acquired on-line or performed on corpus if available. In any of these phases, the reinforcement learning method is the same and is described in the [Algorithm 1](#). Starting from an initial agent, the system makes *maxSteps* interactions, in which the observable state of the environment is passed to the agent, before it chooses an action applied back to the environment which releases a reward. All these interactions are recorded by the agent which uses them during a learning procedure. This training epoch (interactions plus training) is realized a number *maxEpoch* of times.

---

**Algorithm 2** Function used to simulate the evolution of the environment for synthetic data.

---

```

function PERFORMACTION(predictedAction)
  if predictedAction is trueAction  $\vee$ 
    numberOfTries > triesThreshold then
    MOVETONEXTDATASAMPLE
    numberOfTries  $\leftarrow$  0
  else
    numberOfTries  $\leftarrow$  numberOfTries + 1
  end if
end function

```

---

Since during the pre-training phase the agent explores artificially created data, the environment is not able to evolve consequently to a wrong action. Thus, to multiply the number of situations and explore wrong decisions, the function described in [Algorithm 2](#) simulates the environment evolution for synthetic data. In this function, as long as the action chosen is not the correct one, up to *triesThreshold* actions can be successively tried to simulate the patience of the user, repeating the same command until the system eventually behaves correctly.

Similarly, all the experiments presented in this work are realised on a previously recorded corpus. Thus, it is not possible for the user to interactively generate a reward. We defined a simple reward function able to simulate this reward, based on a ground truth (obtained from annotations of the corpus) and the predicted action: *if the chosen action is correct emit +1, else emit -1*.

---

**Algorithm 3** Agent policy

---

**Require:**  $\epsilon \in [0; 1]$

```

function GETACTION
   $x \sim \mathcal{U}([0, 1])$ 
  if  $x \leq \epsilon$  then
     $\text{action} \sim \mathcal{U}(\mathcal{A})$  ▷ Random action
  else
     $Q\text{-values} \leftarrow Q\text{-Network.FORWARD}(\text{state})$ 
     $\text{action} \leftarrow \text{argmax}(Q\text{-values})$ 
  end if
  return  $\text{action}$ 
end function

```

---

Fetching the action to perform requires the agent to execute its policy described in [Algorithm 3](#). We use a near greedy policy named  $\epsilon$ -greedy policy [49]. Given  $\epsilon \in [0; 1]$ , this policy returns a random action with probability  $\epsilon$  or the action with the highest  $Q$ -value given the current state. To compute the  $Q$ -value of each action, the agent uses its approximated  $Q$ -function, modeled by the deep neural network  $Q$ -Network.

---

**Algorithm 4** Agent learning procedure

---

Notation:  $\mathcal{B}_{s_t}$  is the vector composed of the  $s_t$  component of each sample.**function** LEARN $\mathcal{B} \leftarrow \text{GETSAMPLES}(\text{minibatchSize})$  $\triangleright$  Each sample is  $\langle s_t, a_t, r_t, s_{t+1} \rangle$  $\mathbf{Q}\text{-values} \leftarrow ( \mathbf{Q}\text{-Network.FORWARD}(\mathcal{B}_{s_t}) )[\mathcal{B}_{a_t}]$  $\triangleright$  Vector of  $Q$ -values of each  $Q$ -state in  $\mathcal{B}$   
 $\triangleright$  (0 for all  $Q$ -states not in the batch). $\mathbf{targets}\text{-values} \leftarrow \mathcal{B}_{r_t} + \gamma \max ( \mathbf{target}\text{-}Q.\text{FORWARD}(\mathcal{B}_{s_{t+1}}) )$  $\triangleright$  Target values computed with the target network $\mathbf{loss} \leftarrow \mathbf{targets}\text{-values} - \mathbf{Q}\text{-values}$  $\mathbf{gradient} \leftarrow \mathbf{Q}\text{-Network.BACKWARD}(\mathcal{B}_{s_t}, \mathbf{loss})$  $\mathbf{updatedGradient} \leftarrow \text{RMSPROP}(\mathbf{gradient})$  $\mathbf{Q}\text{-Network.parameters} \leftarrow \mathbf{Q}\text{-Network.parameters} + \mathbf{updatedGradient}$ **end function**

---

The learning of the agent follows the work of Mnih et al. [37]. In particular, to estimate the  $Q$ -value the learning does not use the latest experienced interactions. Instead, each interaction  $\langle s_t, a_t, r_t, s_{t+1} \rangle$  is recorded in a pool. Then, a set  $\mathcal{B}$  of interactions is randomly drawn from this pool and used as a (mini-)batch to train the network. An updated  $Q$ -value, more accurate than the current one (taking account of the reward provided by the user) and named target-value, is computed for each  $Q$ -state involved in  $\mathcal{B}$  according to:  $\text{target-value} = r_t + \gamma \max_a Q(s_{t+1}, a)$ . The difference between current  $Q$ -values and target-values is used as a loss and back-propagated through the  $\mathbf{Q}\text{-Network}$  to compute the gradient. The gradient obtained is optimized using the RMSprop optimization algorithms [19] before being applied to the weights of the neural network. This whole process is described in Algorithm 4.

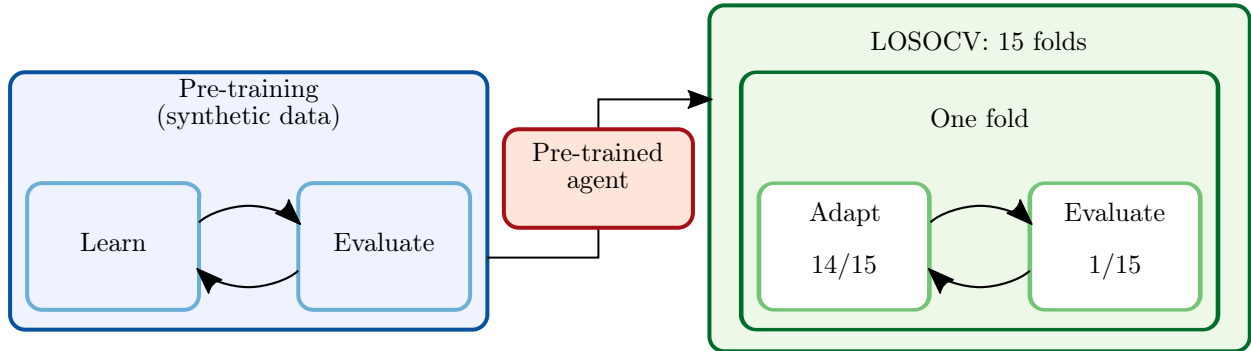


Figure 5: Experiment protocol. After pre-training with synthetic data, the agent is adapted with data from 14 of 15 subjects and evaluated on the left out participant’s data.

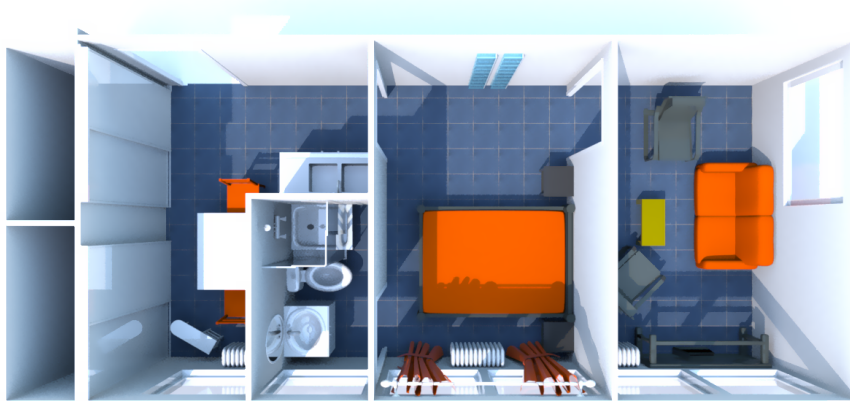


Figure 6: The Domus smart-home

## 4. Experiment

The experiment conducted in this study was decomposed in two phases as illustrated in Figure 5. The first phase, the **pre-training phase**, uses a generated corpus to train an agent and a similar corpus to test it. The **validation phase** then, performs a cross-validation on a real corpus. As data recorded for each participant is comparable, a **Leave One Subject Out Cross-Validation (LOSOVC)** method is used to avoid bias. It uses fourteen of fifteen participants’ data for adapting and tests ARCADES with the left out participant’s data.

### 4.1. Input data

#### 4.1.1. Sweet-Home Corpus

The data corpus was recorded in the DOMUS smart-home designed by Laboratoire d’Informatique de Grenoble (LIG) [16]. This 30 m<sup>2</sup> home includes a bathroom, a kitchen, a bedroom, and a study as show in Figure 6. All these rooms are equipped with sensors and actuators such as infrared motion sensors, tactil sensors, temperature sensors, power meter, etc. In addition, seven microphones in the ceiling capture the audio. The apartment is fully usable and can accommodate a resident for several days. More than 150 devices are managed in the apartment to provide different services (e.g. light, opening/closing shutters, media management, etc.).

To collect decision data in context, several participants were recruited to play scenarios of the daily life in the apartment. While enacting daily life scenarios, participants had to issue voice commands to

activate the actuators in the smart home. The objective of this experiment was to test a smart controller in real situations corresponding to voice commands spoken by the user. This study considered the following situations: to cleanup the apartment; to prepare and eat a meal; to converse via video conference; to do leisure activities (reading); to take a nap.

To guide the participants, the grammar voice command was provided with a scenario of everyday life. This scenario was designed to last about 45 minutes, however, there was no constraint on the execution time. Each participant received a list of actions to make and voice commands to utter that was interpreted by an intelligent system acting accordingly. Each participant had to use a voice command, repeating up to 3 times in case of system failure, before a Wizard of Oz was used.

A total of 15 people (9 women, 6 men) participated in the experiment to record 11 hours of data. The average age of participants was  $38 \pm 13.6$  (19 – 62). All experiments were video recorded but only for annotation purposes. This data set is part of the publicly available Sweet-Home corpus [53].

#### 4.1.2. State and action sets

The state space can be seen from different points of view. If the state space is represented only by the location and activity of the user (i.e., classical context variables), the number of states is 324 [7]. But these contextual information necessitate inference models and our aim is to directly make decision from raw data. When state space is represented by raw data issued by the 81 available sensors (49 binary sensors and 32 continuous sensors which can take values in different ranges – from  $[0; 100]$  up to  $[0; 2^{32} - 1]$ ), this leads to a big number (more than  $10^{12}$ ) of possible states. This shows again that decision making from raw data cannot be performed using explicit representation and that approximation of the kind presented here with neural network is the only way to make this decision making tractable.

The action set depends on the number of actuators and commands concerning them. Even if some actions were never realized during the corpus record, we chose to handle a large set of actions. Thus, any device used at least once has been incorporated, with all its possible commands even if some of them are not used. This ended up with a set of 33 possible actions listed in Table 2.

Table 2: Available actions for the agent to act on the environment.

Command	Actuator devices
turn on the light	all lights in the kitchen, light above the sink in the kitchen, ceiling light of the kitchen, all lights in the bedroom, bedside lamp of the bedroom, ceiling light of the bedroom, ceiling light of the study
turn off the light	all lights in the kitchen, light above the sink in the kitchen, ceiling light of the kitchen, all lights in the bedroom, bedside lamp of the bedroom, ceiling light of the bedroom, ceiling light of the study
turn on the radio	bedroom radio
turn off the radio	bedroom radio
open the blinds	kitchen blinds, bedroom blinds, study blinds
close the blinds	kitchen blinds, bedroom blinds, study blinds
open the curtains	bedroom curtains
close the curtains	bedroom curtains
give time	kitchen speakers, bedroom speakers, study speakers
give temperature	kitchen speakers, bedroom speakers, study speakers
call emergency	study phone
call a parent	study phone
do nothing	nowhere

### 4.1.3. Generated corpus

From a deep-learning point of view, the Sweet-Home corpus does not contain enough relevant samples of voice commands. Furthermore, it contains unpredictable samples or missing data, which is not advisable for a first proof of concept. As a workaround, a data generation system was developed based on a few generation rules. This system can be tuned to generate deterministic data (which always associate the same output to the same input), restricted data (focusing only on most significant events), etc.

The generated corpus follows the same format as the real raw data corpus. Thus, the subsequent translation of the annotations to SVG image (cf. [subsection 3.2](#)) can be applied seemingly to the real or generated data. The generation system producing samples on-demand, this corpus is theoretically of infinite size, a new sample always being available if necessary.

The corpus is generated to use the ability of the deep learning to build its own features from raw data. Thus, the generation consisted in generating annotations for the voice command and the expected action as well as simulating values for the raw sensor data. To achieve this, we generate an “annotated” state, similar to the ones used in our preliminary study [7], which is converted using logical rules. For example, if the annotated location is the kitchen, then the presence detector sensors of the kitchen are constrained to be active. In the meantime, the bedroom being the room next to the kitchen, its presence detector sensors are constrained to be active with a defined probability. This mechanism is applied for all relevant sensors, while random values are applied to irrelevant ones. The real Sweet-Home corpus is a set of CSV files, one for each sensor plus two for the uttered command annotation and the expected action (the target of the learning). These files contain the value for the data source for each time-stamp which can be parsed to generate a data structure similar to the one generated by our generation system.

### 4.2. Experiment: learning on raw data

First attempts to train a model from images generated with raw data lead to poor performances (oscillations, long convergence time, etc.). To address this problem, we choose to investigate the hyper-parameter choices, conducting an exhaustive search for three of them which are `target_q`, `minibatch_size` and `update_freq`. Result of this research is summed up in [Appendix A](#). These parameters are used by the agent, `minibatch_size` and `update_freq` being directly related to the learning procedure described in [Algorithm 4](#). The former is the number of samples used for the training, that is to say the size of  $\mathcal{B}$ . The latter is the number of steps done before calling the LEARN procedure. As explained, the learning does not happen after each interaction but after each `update_freq` interactions. The last parameter, `target_q`, is related to the target network trick introduced by Mnih et al. [37], and the whole details of its use is out of the scope of this paper. However, these three parameters are tightly linked. A high `target_q` and `minibatch_size` must improve the overall stability of the  $Q$ -network, but a high `minibatch_size` leads to a longer training time. Thus, increasing the `update_freq` must reduce this learning time.

While the data streams of synthetic generated corpus are perfectly synchronized, this is not the case for the real data. Despite our efforts to synchronize sensor streams and annotations, the real smart-home corpus does not provide a high quality of synchronization. To overcome this problem, we choose to provide to the  $Q$ -network a wider window of the last 3 seconds of the state of the environment. However, this mechanism generate many samples where no action is required by the agent (2 samples of no action for one sample of any of the 32 other actions), and thus heavily unbalance our classes. To overcome this, the reward function was updated to heavily penalize an error about this majority class. Instead of receiving a reward of  $(-1)$ , the agent receives a reward of  $(-192)$  if it chooses to do nothing while an action is expected.

The parameters of this experiment are detailed in [Table 3](#). The *pre-training phase* used synthetic data generating images similar to the one presented in [Figure 3](#). The model was trained using up to  $10 \cdot 10^6$  interactions, evaluating the system on 5000 independent interactions every  $100 \cdot 10^3$  interactions. Once a model is learned from scratch, the *validation phase* adapts it to new data. Each fold of the LOSOCV method runs the system for  $400 \cdot 10^3$  interactions on the real data. Test of ARCADES happens every 4000 interactions, running 5000 monitored interactions.

To compare against a classical approach a **baseline** system using a tabular Q-learning has been learned and tested by LOSOCV. However, since the classical approach uses an array to represent the state, this



Table 3: Parameters of the experiment.

(a) Pre-training		(b) Validation	
Parameter	Value	Parameter	Value
Git tag	<b>synthetic-raw-full-1</b>	Git tag	<b>real-raw-full-1</b>
Data	synthetic, raw	Data	real, raw
<i>totalLearningSteps</i>	$10 \cdot 10^6$	<i>totalLearningSteps</i>	$400 \cdot 10^3$
<i>learningSteps</i>	$100 \cdot 10^3$	<i>learningSteps</i>	4000
<i>evaluationSteps</i>	5000	<i>evaluationSteps</i>	5000

approach would not have been tractable if it was applied directly to the large amount of sensors values (cf. [subsection 3.3](#)). Hence, the baseline was trained on the ground truth humanly annotated location and activity labels. An excerpt of the ground truth data is showed below. It shows that a voice command to open the blinds in the kitchen should be followed by the opening of all the blinds of the kitchen. The second line is related to [Scenario 1](#), when the light is asked to be turned on while cooking in the kitchen, the only possible action is to light on the lamp above the sink.

```
open - blind kitchen none -> open blind kitchen
on - light kitchen cook -> on light kitchen - sink
```

Thus this kind of ground truth dataset favours the baseline performances.

[Table 4](#) summarizes the result of the learning. Even though the reinforcement learning objective is to maximise the reward, in this paper, standard classification measures such an F-Measure have been chosen. Indeed, the task can be seen as a classification one, classifying a sample (the environment state) between 33 classes (the actions). In our experiment, a correct classification is considered only when the chosen action is composed of the right action (e.g., turn on) on the right device (e.g., light) at the right place (e.g., study). Any other choice is a wrong classification.

Table 4: F-Measure of the pre-training and evaluation phases for the decision making task.

	Pre-training F-M (%) synthetic data	Validation F-M (%) real corpus
Baseline <sup>1</sup>	NC	46.0
ARCADES <sup>2</sup>	100	67.5

1. Learned and evaluated on humanly annotated data.      2. Learned and evaluated on raw data.

While the baseline tabular Q-learning only deals with humanly annotated data, it presents a low F-Measure of 46%. ARCADES on raw sensor data does not lead to very high performances, but it demonstrates that a context aware decision-making model without an explicit representation of context is possible and lead to a very promising result of 67.5% of F-Measure.

Detailed results of ARCADES are presented [Figure 7](#) and [Figure 8](#). These are presented using the following metrics:

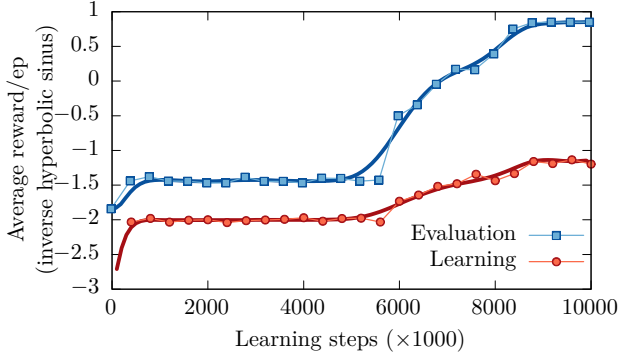
- *Reward per episode*: This value is the objective of the agent. It is the average sum of rewards obtain during each episode (an independent set of interactions). Given the reward function and the maximum length of an episode, the reward function cannot be greater than 1.
- *Precision, Recall, F1-Score*: standard performance measures for classification. ARCADES can be seen as a classifier, classifying a sample (the environment state) between 33 classes (the actions).

First, it is interesting to note that the evaluation performances are better than the learning ones. This is explained with the exploration factor (the  $\epsilon$  of the  $\epsilon$ -greedy policy), which is decreased from 0.99 to 0.5 during the first 200 000 learning interactions (after that, 1 action over 2 is selected randomly) while it is set to 0 during the evaluation phase (greedy policy, no random decision).

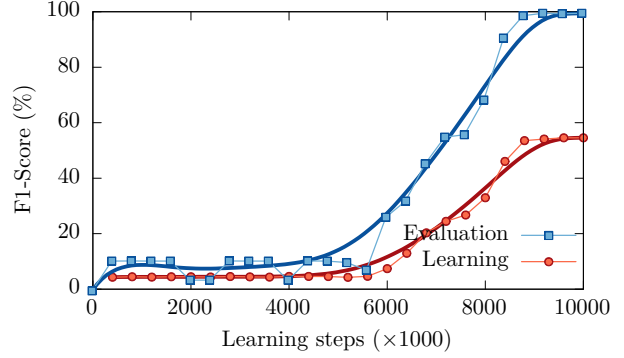
Second, while being very long, the *pre-training phase* finally reaches a very high plateau. With more than 4 days and 16 hours of training ARCADES acquires a coherent behavior as of the  $8 \cdot 10^6$ <sup>th</sup> interaction (about 100 days with one decision per second), as shown in [Figure 7](#). Then, during the *validation phase* the system slightly adapts to the real data, starting with low performances, to finally reach more than 65% of F1-Score ([Figure 8](#)). Compared to the pre-training phase, this adaptation happens in a short time, comparable to the one obtained with the baseline system.

(a) Final scores

Reward per ep.	Precision	Recall	F1-score
0.99	100 %	100 %	100 %



(b) Average reward per episode growth

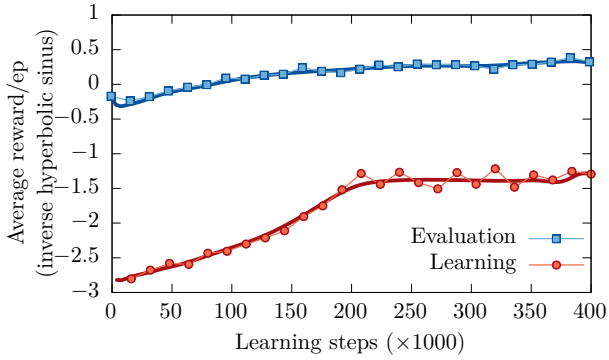


(c) F1-score growth

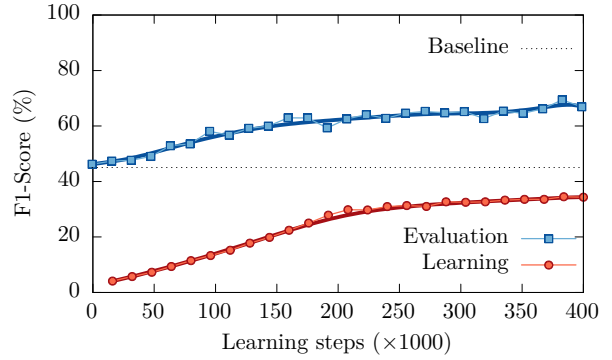
Figure 7: Pre-training results

(a) Final scores

Reward per ep.	Precision	Recall	F1-score
0.36	68.91 %	66.36 %	67.49 %



(b) Average reward per episode growth



(c) F1-score growth

Figure 8: Validation results

## 5. Complementary experiments

While in the case of a classical tabular  $Q$ -learning, the associative table can be interpreted, this is more complicated for the parameters of a neural network. This is a recurrent grief addressed to neuronal models, and thus methods have been developed to better understand how they work [24]. In this section, we present some complementary experiments aiming at understanding ARCADES mechanisms.

In subsection 5.1 we test the ability of the system not to adapt to user preferences, but to changes in the sensor topology. Then, we investigate, in subsection 5.2, which features the neural network builds, and if they match the ones usually found in the literature.

### 5.1. Adaptation to the hardware

One of the main goals of ARCADES is to adapt itself to the user’s preferences, but also to the sensors. To verify this hypothesis, we made experiments in which some sensors disappear (because of faulty sensors or if the user uninstall them). Four experiments were performed, based on a pre-trained (with synthetic raw data) model, in which different kinds of sensor were removed.

In the first experiment, the kitchen motion sensors were removed. Figure 9 shows that the performances are impacted but that the system quickly adapts its behavior to reach back its initial performances.

In the next experiments, the acoustic noise from the kitchen microphone was removed, then door switches for kitchen cupboard and fridge. The latter are particularly active when the user tidy the home, and then we expected a dip of the performances. However, for these two experiments, no impact can be measured (Figure 10 and Figure 11) suggesting that the information obtained from these sensors are of little use to ARCADES.

The last experiment simulates the break of the automatic speech recognition system. While we could expect a drastic fall of the performances, near the random agent which would get about 3% of F1-Score, the system still acts not so badly, with an F1-Score oscillating around  $(20 \pm 10)\%$ . Noticing that the average reward is not so far below (0) is another hint that its behavior is far from random. One way to explain this, is that the system still gets images 2 seconds before an action is expected. This information seems to be grabbed by the system during pre-training, and used to know when an action is expected (even if it is not the right one). However, the oscillations of the score shows that ARCADES is not able to infer the user intention only from the sensor values (i.e. from contextual information).

These results pushed us to ask which features the system does extract to make its decision.

### 5.2. State features

During its forward pass through the network, the input image is transformed to extract relevant features, resulting in high-dimensional vectors. One way to study these vectors is to project them in a two-dimensional space while preserving some inter-vectors relations such as the euclidean distance. One of the most used projection method is the t-SNE (t-distributed Stochastic Neighbor Embedding) projection [54] which perform very well to project vectors issued by a convolutional neural network.

In the following experiment, we provide some states (images) to a pre-trained model, while recording vectors after the last convolutional layer, and the one before the last fully connected layer. These vectors are also linked to the information characterizing the input state (location, activity, uttered command, expected action, sensors activity). Projecting the vectors provide a bi-dimensional coordinate  $(0,0) \leq (x,y) \leq (1,1)$  where to represent any of the linked information.

Results presented in Figure 13(c, b), show that some information are identified and extracted by the convolutional part of the network. For example, coherent blobs can be spotted for the uttered command and the user location. By contrast, no group can be made for the activity (Figure 13a), which does not seem to be a relevant feature to extract from the image for the DQN. These results confirm the usual use of the user location as a contextual information. However, the user’s activity is not exploited by the network. It might be possible that a lower level annotations, such as agitation level, could be extracted.

The last projection, presented in Figure 13d, provides information about the way the neural network transforms the space to make each class linearly separable. While there is a majority of “Do Nothing” action, we can identify some groups of common action type (icon type) and action place (icon color), proving that given states contain enough information to classify correctly most of them.

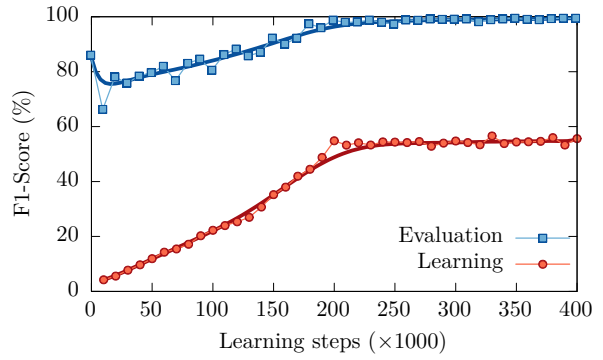
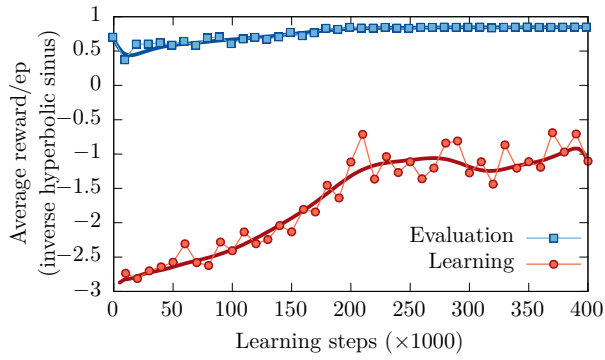


Figure 9: Performances without motion sensors

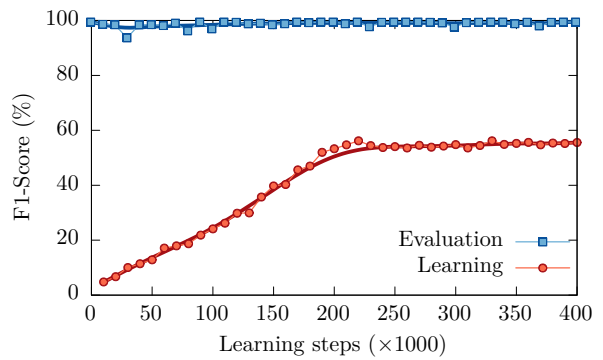
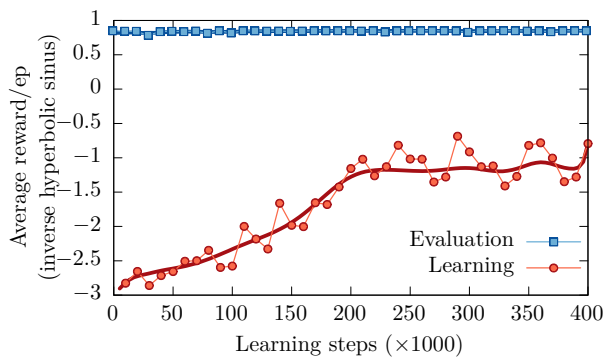


Figure 10: Performances without acoustic noise

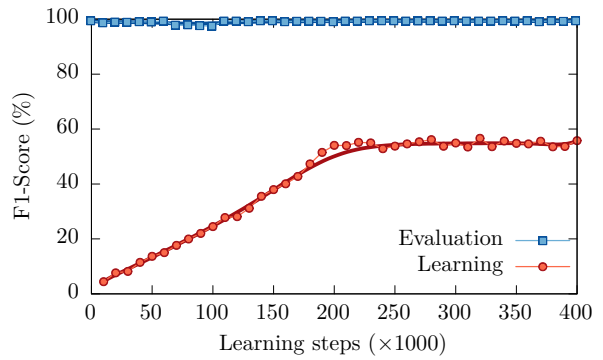
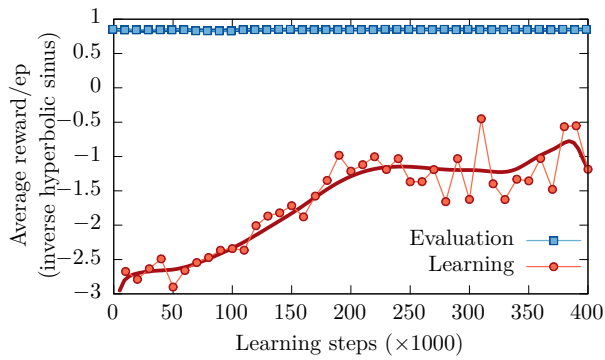


Figure 11: Performances without door switches

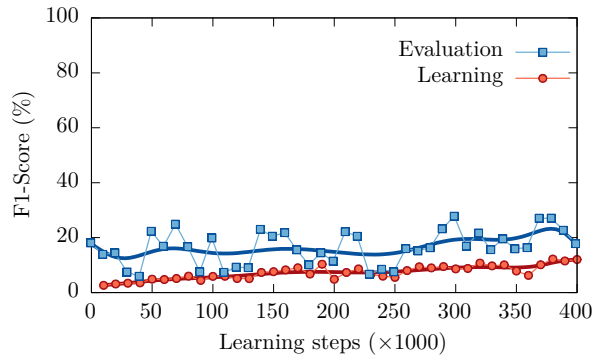
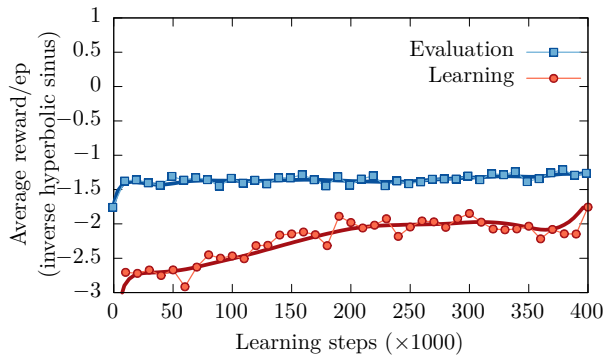
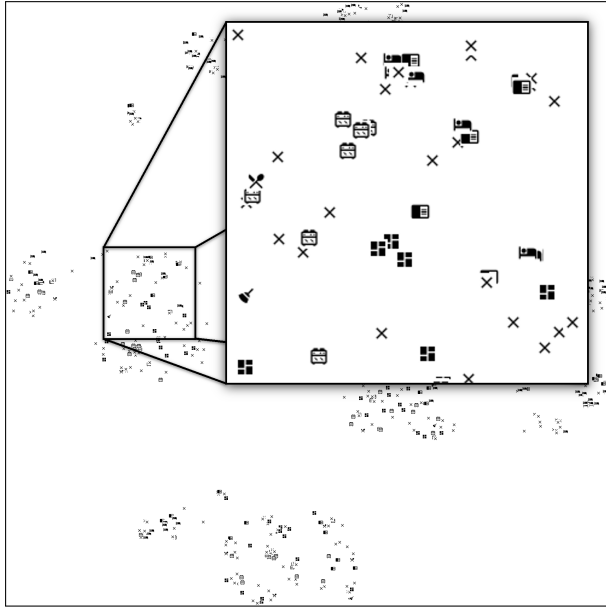
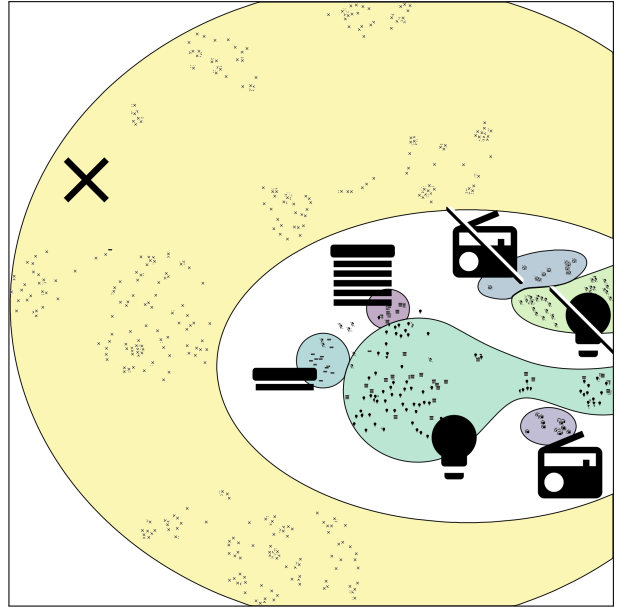


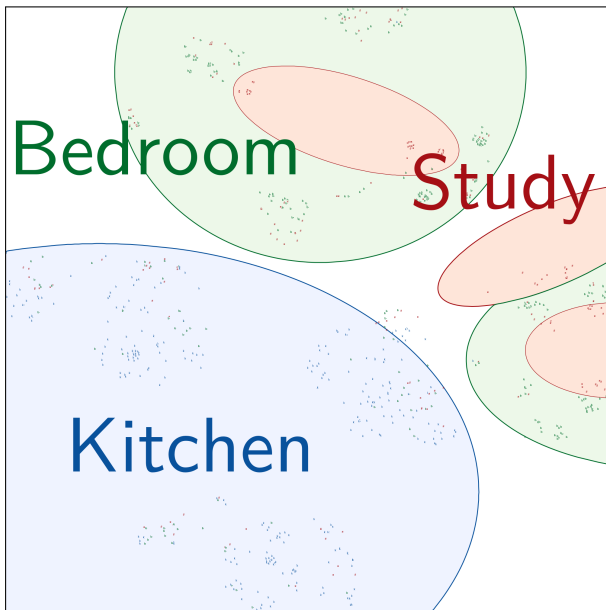
Figure 12: Performances without speech recognition



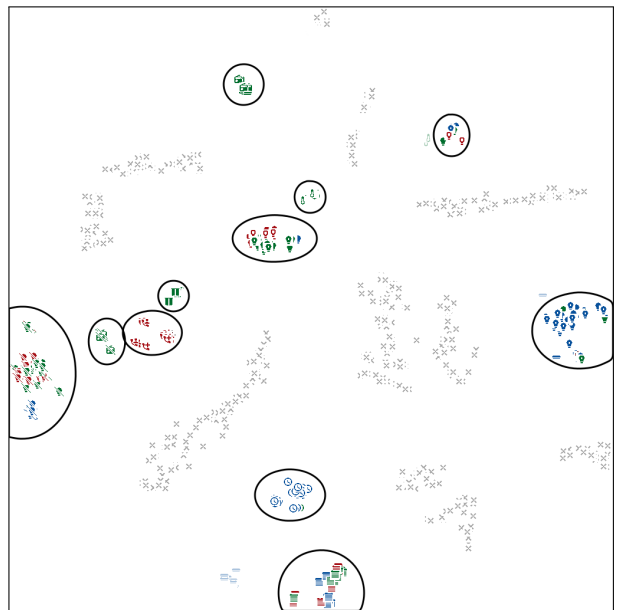
(a) Post-convolution: user activity



(b) Post-convolution: uttered command



(c) Post-convolution: user location



(d) Pre-classification: expected action

Figure 13: t-SNE projections

## 6. Discussion

The results of the experiments of a reinforcement learning method with a deep model for automatic adaptive decision-making in smart-homes show promises. In particular, the experiments show that:

- the system is able to use trivial user’s interactions (rewards) to adapt a decision-making process;
- graphical representation of multi-modal heterogeneous data can be interpreted to make decisions;
- the system can adapt its behavior to a user and to faulty or uninstalled sensors;
- the deep model can learn relevant contextual information;
- learning can be done on-line, using a limited history of the past and no information about the future;
- the method can scale to big input space (raw sensor data).

All the work has been conceived to be reproducible and all the written source code as well as the graphic components are available on-line<sup>2</sup>, as a set of Lua source code and SVG files. All these sources have been put under version control system (Git). This allows the reproduction of the experiments, freezing and tagging the source for each experiment. The source code follows a modular approach for improving re-usability of each component. Regarding the data used for the experiments, it can be retrieved separately<sup>3</sup>.

If ARCADES exhibits interesting results, it opens new research questions and possible improvements.

At the RL level, the reward function plays a very important role in driving the learning in an acceptable solution. In our previous study [7] different reward functions were tried, taking account of the semantic distance between two actions (e.g. lighting a lamp instead of another) or the real distance between two action locations (lighting in the kitchen rather than the study). Nevertheless, these more sophisticated functions did not bring a significant improvement in the agent behavior that is why we decided to stick to a more simple one. Though this function led to better learning, it is worth investigating other reward function that would be closer to human evaluation (e.g. a confusion between a lamp in a room should be less problematic than a confusion of room). Similarly, in a real interactive setup, microphones can be used to acquire a more nuanced evaluation from the user, for example using a set of keywords like “No!”, “Very good.”, “Keep going.”, etc. Another research direction would be to take into account implicit rewards from the user as done by Karami et al. [23].

At a deep learning level, it is well-known that these models require an unreasonable quantity of data as well as a long time before reaching the convergence point. These are important factors for acceptability of the system by naive user and should be reduced as much as possible. If the pre-training phase is a first solution, others should be studied, such as the use of an expert system during the first interactions, waiting for ARCADES to learn an appropriate model. Recent works propose different methods to leverage these parameters. For instance, Pritzel et al. [44] make an intensive use of few relevant samples to reduce the number of the overall required interactions. In parallel, Usunier et al. [51] propose a different strategy than the  $\epsilon$ -greedy one, which speeds up the exploration of the different actions. Knowledge transfer could be another alternative which need more experiments to be validated.

Finally, paradigm shifts can also be interesting. The use of MDP as the underlying model of RL tasks is related to the computation complexity induced by more generic models like the POMDP [8]. However, these models are more suitable to handle the unknown and variable behavior of the user. The DQN approach being able to scale to big state space, it can be interesting to try to use more generic models like POMDP or MOMDP [1] as the underlying one of our RL task. More complex neural network architectures can also be implemented such as Recurrent Neural Networks (RNN, [34]), allowing the system to better handle time dependency as well as implementing some attention mechanisms [57].

Despite this need for improvements, this study show that deep reinforcement learning is feasible and shows promises for a life long adaptive and robust context aware decision making.

---

<sup>2</sup>[brenona.gricad-pages.univ-grenoble-alpes.fr/arcades/](http://brenona.gricad-pages.univ-grenoble-alpes.fr/arcades/)

<sup>3</sup><http://sweet-home-data.imag.fr/>

- [1] Araya-López, M., Thomas, V., Buffet, O., Charpillet, F., 2010. A closer look at MOMDPs. In: 22<sup>nd</sup> International Conference on Tools with Artificial Intelligence (ICTAI 2010). Proceedings of the 22<sup>nd</sup> International Conference on Tools with Artificial Intelligence. pp. 197–204.
- [2] Attard, J., Scerri, S., Rivera, I., Handschuh, S., 2013. Ontology-based situation recognition for context-aware systems. In: Proceedings of the 9<sup>th</sup> International Conference on Semantic Systems. I-SEMANTICS 2013. ACM, New York, NY, USA, pp. 113–120.
- [3] Augusto, J. C., Callaghan, V., Cook, D., Kameas, A., Satoh, I., Jun 2013. Intelligent environments: a manifesto. *Human-centric Computing and Information Sciences* 3 (1), 12.
- [4] Bellman, R., 1957. *Dynamic Programming*, 1st Edition. Princeton University Press.
- [5] Berlink, H., Costa, A. H. R., 2015. Batch reinforcement learning for smart home energy management. In: Proceedings of the 24th International Conference on Artificial Intelligence. IJCAI'15. pp. 2561–2567.
- [6] Bolchini, C., Curino, C. A., Quintarelli, E., Schreiber, F. A., Tanca, L., 2007. A data-oriented survey of context models. *SIGMOD Rec.* 36 (4), 19–26.
- [7] Brenon, A., Portet, F., Vacher, M., 2016. Preliminary study of adaptive decision-making system for vocal command in smart home. In: 12<sup>th</sup> International Conference on Intelligent Environments. pp. 218–221.
- [8] Cassandra, A. R., 2016. Pomdp for dummies. Website, consulted on 2016-05-17.  
URL <http://pomdp.org/>
- [9] Chahuara, P., Fleury, A., Portet, F., Vacher, M., 7 2016. On-line human activity recognition from audio and home automation sensors: comparison of sequential and non-sequential models in realistic smart homes. *Journal of Ambient Intelligence and Smart Environments* 8 (4), 399–422.
- [10] Chahuara, P., Portet, F., Vacher, M., 2017. Context-aware decision making under uncertainty for voice-based control of smart home. *Expert Systems with Applications* 75, 63–79.
- [11] Chen, L., Nugent, C. D., Mulvenna, M., Finlay, D., Hong, X., Poland, M., Dec. 2008. A logical framework for behaviour reasoning and assistance in a smart home. *International Journal of Assistive Robotics and Mechatronics* 9 (4), 20–34.
- [12] Christensen, H., Casanueva, I., Cunningham, S., Green, P., Hain, T., 2013. homeservice: Voice-enabled assistive technology in the home using cloud-based automatic speech recognition. In: 4<sup>th</sup> Workshop on Speech and Language Processing for Assistive Technologies. pp. 29–34.
- [13] De Carolis, B., Cozzolongo, G., 2004. C@sa: Intelligent home control and simulation. In: Okatan, A. (Ed.), International Conference on Computational Intelligence. International Computational Intelligence Society, Istanbul, Turkey, pp. 462–465, international Conference on Computational Intelligence, ICCI 2004, December 17-19, 2004, Istanbul, Turkey, Proceedings.
- [14] Dey, A. K., 2001. Understanding and using context. *Personal Ubiquitous Computing* 5 (1), 4–7.
- [15] Filho, G., Moir, T. J., 2010. From science fiction to science fact: a smart-house interface using speech technology and a photorealistic avatar. *International Journal of Computer Applications in Technology* 39 (8), 32–39.
- [16] Gallissot, M., Caelen, J., Jambon, F., Meillon, B., 2013. Une plateforme usage pour l'intégration de l'informatique ambiante dans l'habitat. l'appartement domus. *Technique et Science Informatiques (TSI)* 32 (5), 547–574.
- [17] Gómez-Romero, J., Serrano, M. A., Patricio, M. A., García, J., Molina, J. M., Oct. 2012. Context-based scene recognition from visual data in smart homes: an information fusion approach. *Personal and Ubiquitous Computing* 16 (7), 835–857.
- [18] Helaoui, R., Riboni, D., Stuckenschmidt, H., 2013. A probabilistic ontological framework for the recognition of multi-level human activities. In: Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing. UbiComp 2013. ACM, New York, NY, USA, pp. 345–354.
- [19] Hinton, G. E., 2015. Lecture 6e – rmsprop: Divide the gradient by a running average of its recent magnitude. Online courses.  
URL [http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture\\_slides\\_lec6.pdf](http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf)
- [20] Howard, R. A., Matheson, J. E., 1981. Influence diagrams. In: *Readings on The Principles and Applications of Decision Analysis*. Vol. 2. Institute for Operations Research and the Management Sciences (INFORMS), pp. 127–143.
- [21] Ibarra, U. A., Augusto, J. C., Clark, T., 2016. Engineering context-aware systems and applications: A survey. *Journal of Systems and Software* 117, 55–83.
- [22] Istrate, D., Vacher, M., Serignat, J.-F., 2008. Embedded implementation of distress situation identification through sound analysis. *The Journal on Information Technology in Healthcare* 6, 204–211.
- [23] Karami, A.-B., Fleury, A., Boonaert, J., Lecoeuche, S., 2016. User in the loop: Adaptive smart homes exploiting user feedback — state of the art and future directions. *Information* 7, 35.
- [24] Karpathy, A., 2015. Cs231n: Convolutional neural networks for visual recognition. Web site.  
URL <https://cs231n.github.io/>
- [25] Katzouris, N., Artikis, A., Paliouras, G., 2014. Event recognition for unobtrusive assisted living. In: 8<sup>th</sup> Hellenic Conference on AI. Springer-Verlag, Ioannina, Greece, pp. 475–488.
- [26] Kofler, M. J., Reinisch, C., Kastner, W., Apr. 2012. A semantic representation of energy-related information in future smart homes. *Energy and Buildings* 47, 169–179.
- [27] Lecouteux, B., Vacher, M., Portet, F., Aug. 2011. Distant speech recognition in a smart home: Comparison of several multisource asrs in realistic conditions. In: Interspeech 2011. Florence, Italy, pp. 2273–2276.
- [28] LeCun, Y., Bengio, Y., Hinton, G., 2015. Deep learning. *Nature* 521, 436–444.
- [29] Lee, S.-H., Cho, S.-B., 2012. Fusion of modular bayesian networks for context-aware decision making. In: Corchado, E., Šnášel, V., Abraham, A., Wozniak, M., Graña, M., Cho, S.-B. (Eds.), *Hybrid Artificial Intelligent Systems*. Vol. 7208 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg, pp. 375–384.
- [30] Leong, C. Y., Ramli, A. R., Perumal, T., 2009. A rule-based framework for heterogeneous subsystems management in smart home environment. *IEEE Transactions on Consumer Electronics* 55 (3), 1208–1213.



- [31] Li, D., Jayaweera, S. K., 2014. Reinforcement learning aided smart-home decision-making in an interactive smart grid. In: 2014 IEEE Green Energy and Systems Conference (IGESC). pp. 1–6.
- [32] Liao, H.-C., Tu, C.-C., 2007. A rdf and owl-based temporal context reasoning model for smart home. *Information Technology Journal* 6 (8), 1130–1138.
- [33] Lin, L.-J., 1993. Reinforcement learning for robots using neural networks. Tech. rep., DTIC Document.
- [34] Lipton, Z. C., Berkowitz, J., Elkan, C., 2015. A critical review of recurrent neural networks for sequence learning. ArXiv.
- [35] Loke, S., 2006. *Context-Aware Pervasive Systems*. Auerbach Publications, Boston, USA.
- [36] Mileo, A., Merico, D., Bisiani, R., 2011. Reasoning support for risk prediction and prevention in independent living. *Theory Pract. Log. Program.* 11 (2-3), 361–395.
- [37] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al., 2015. Human-level control through deep reinforcement learning. *Nature* 518 (7540), 529–533.
- [38] Moore, P., Hu, B., Jackson, M., 2011. Rule strategies for intelligent context-aware systems: The application of conditional relationships in decision-support. In: *International Conference on Complex, Intelligent and Software Intensive Systems, CISIS 2011*. Seoul, Korea, pp. 9–16.
- [39] Mozer, M. C., 1998. The neural network house: An environment that adapts to its inhabitants. *Proceedings of the American Association for Artificial Intelligence*.
- [40] Mozer, M. C., 2005. *Smart Environments: Technology, Protocols and Applications*. John Wiley & Sons, Inc., Ch. 12 - Lessons from an Adaptive Home, pp. 271–294.
- [41] Nishiyama, T., Hibiya, S., Sawaragi, T., 2011. Development of agent system based on decision model for creating an ambient space. *AI & Society* 26 (3), 247–259.  
URL <http://www.springerlink.com/content/j64226q53803r301/abstract/>
- [42] Peetoom, K. K. B., Lexis, M. A. S., Joore, M., Dirksen, C. D., De Witte, L. P., 2014. Literature review on monitoring technologies and their outcomes in independently living elderly people. *Disability and Rehabilitation: Assistive Technology*, 1–24.
- [43] Portet, F., Vacher, M., Golanski, C., Roux, C., Meillon, B., 2013. Design and evaluation of a smart home voice interface for the elderly: acceptability and objection aspects. *Personal and Ubiquitous Computing* 17, 127–144.
- [44] Pritzel, A., Uria, B., Srinivasan, S., Puigdomènech, A., Vinyals, O., Hassabis, D., Wierstra, D., Blundell, C., 2017. Neural episodic control. *CoRR*.
- [45] Rodríguez, N. D., Cuéllar, M. P., Lilius, J., Calvo-Flores, M. D., 2014. A fuzzy ontology for semantic modelling and recognition of human behaviour. *Knowledge-Based Systems* 66, 46–60.
- [46] Rodríguez, N. D., Cuéllar, M. P., Lilius, J., Calvo-Flores, M. D., Mar. 2014. A survey on ontologies for human behavior recognition. *ACM Computing Surveys* 46 (4), 1–33.
- [47] Schmidhuber, J., 2015. Deep learning in neural networks: An overview. *Neural Networks* 61, 85–117.
- [48] Singh, M. S., Pondenkandath, V., Zhou, B., Lukowicz, P., Liwicki, M., 2017. Transforming sensor data to the image domain for deep learning - an application to footstep detection. In: *International Joint Conference on Neural Networks, IJCNN 2017*. Anchorage, USA, pp. 2665–2672.
- [49] Sutton, R. S., Barto, A. G., 2017. *Reinforcement Learning: An Introduction*, 2nd Edition. The M. I. T. Press.  
URL <http://incompleteideas.net/sutton/book/the-book-2nd.html>
- [50] Tsitsiklis, J., Roy, B. V., 1997. An analysis if temporal-difference learning with function approximation. *IEEE Trans. Automat. Contr.*, 674–690.
- [51] Usunier, N., Synnaeve, G., Lin, Z., Chintala, S., 2016. Episodic exploration for deep deterministic policies: An application to starcraft micromanagement tasks. ArXiv.
- [52] Vacher, M., Caffiau, S., Portet, F., Meillon, B., Roux, C., Elias, E., Lecouteux, B., Chahuaara, P., 2015. Evaluation of a context-aware voice interface for ambient assisted living. *ACM Transactions on Accessible Computing* 7 (2), 1–36.
- [53] Vacher, M., Lecouteux, B., Chahuaara, P., Portet, F., Meillon, B., Bonnefond, N., 2014. The sweet-home speech and multimodal corpus for home automation interaction. In: *The 9<sup>th</sup> edition of the Language Resources and Evaluation Conference (LREC)*. Reykjavik, Iceland, pp. 4499–4506.
- [54] van der Maaten, L. J., Hinton, G., 2008. Visualizing high-dimensional data using t-sne. *Journal of Machine Learning Research*, 2579–2605.  
URL <https://lvdmaaten.github.io/tsne/>
- [55] Watkins, C. J. C. H., 1989. Learning from delayed rewards. Ph.D. thesis, King’s College.
- [56] Weiser, M., 1991. The computer for the 21<sup>st</sup> century. *SIGMOBILE Mob. Comput. Commun. Rev.* 3 (3), 3–11.
- [57] Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., Zemel, R., Bengio, Y., 2015. Show, attend and tell: Neural image caption generation with visual attention. *Computer Research Repository (CoRR)*.
- [58] Yau, S. S., Liu, J., 2006. Hierarchical situation modeling and reasoning for pervasive computing. In: *Proceedings of the The Fourth IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems, and the Second International Workshop on Collaborative Computing, Integration, and Assurance (SEUS-WCCIA 2006)*. SEUS-WCCIA 2006. IEEE Computer Society, Washington, DC, USA, pp. 5–10.
- [59] Yilmaz, O., Erdur, R. C., Feb. 2012. iconawa – an intelligent context-aware system. *Expert Systems with Applications* 39 (3), 2907–2918.  
URL <http://www.sciencedirect.com/science/article/pii/S0957417411012838>
- [60] Zaidenberg, S., Reignier, P., 2011. Reinforcement learning of user preferences for a ubiquitous personal assistant. In: Mellouk, A. (Ed.), *Advances in Reinforcement Learning*. Intech, pp. 59–80.
- [61] Zaidenberg, S., Reignier, P., Mandran, N., 2010. Learning user preferences in ubiquitous systems: A user study and a reinforcement learning approach. In: Papadopoulos, H., Andreou, A., Bramer, M. (Eds.), *Artificial Intelligence Applications*

and Innovations. Vol. 339 of IFIP Advances in Information and Communication Technology. Springer Berlin Heidelberg, pp. 336–343.

## Appendix A. Hyper-parameters optimization

Through the whole available hyper-parameters of the system, we choose to make an exhaustive search for two of them. The `target_q` parameter defines the number of interactions during which a frozen network is used to compute the target  $Q$ -values before being updated with the current  $Q$ -network. The `minibatch_size` parameter defines the number of samples drawn to build a mini-batch used for training the network.

Mnih et al. [37] state that these two parameters are very important to provide a good stability of the performances. Nevertheless, as shown on the bottom left plot of the Figure A.14, the `minibatch_size` parameter heavily impact the execution time. In spite of improving the average reward per episode and F1-score, using a high `minibatch_size` make the system slower to process the same number of interactions. The bottom right plot provides an intuitive view of a compromise, relating the F1-score to the execution time. This lead us to choose a `target_q` of 4096 with a `minibatch_size` of 16.

However, another hyper-parameter, called `update_freq`, allows to reduce the execution time. The learning step of the agent is indeed made each `update_freq` interactions. Thus, increasing this number will execute the time-consuming step less often, and yet need bigger mini-batches to reach the same level of performance. The Figure A.15 presents the same plots but with different hyper-parameters. As expected, a low `update_freq` increases the execution time. The plot of the score-time ratio lead us to choose a bigger `minibatch_size` parameter of 32, increasing the `update_freq` parameter to 12.

The old and new values of hyper-parameters are summed up in the Table A.5.

Table A.5: Evolution of the hyper-parameters

	Initial value	Intermediate value	Final value
<code>target_q</code>	1024	4096	4096
<code>minibatch_size</code>	64	16	32
<code>update_freq</code>	3	3	12

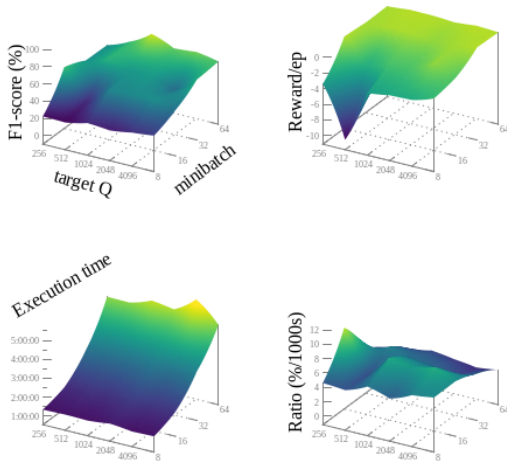


Figure A.14: Metrics related to different choices of hyper-parameters

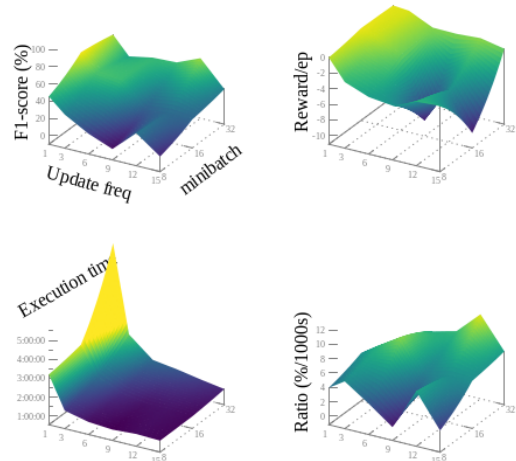


Figure A.15: Metrics related to different choices of hyper-parameters