International Journal of Computer Applications (0975 - 8887)
Volume 152 - No.6, October 2016

# Deep Q-Learning for Home Automation

Vignesh Gokul
Student
SSN College of Engineering

Parinitha Kannan
Student
SSN College of Engineering

Sharath Kumar
Student
SSN College of Engineering

Shomona Gracia Jacob, PhD
Associate Professor
SSN College of Engineering

## ABSTRACT

In this paper, the first deep reinforcement learning model for home automation systems is presented. Home automation has been one of the most important applications in the field of Artificial Intelligence. The system should learn the pattern and behaviour of the user automatically from experience and take future actions accordingly. The system proposed here makes use only of images to learn the user's needs using Deep Q-Learning, thus minimizing the use of any sensors and other hardware. The model makes use of a Convolutional Neural Network that takes as input, the image and outputs the future reward for each action. The system was tested with images of a house and describes the methods and results in the paper.

## General Terms

Artificial Intelligence, Pattern Recognition

## Keywords

Home Automation, Smart Homes, Deep Q-Learning, Reinforcement Learning.

## 1. INTRODUCTION

Smart homes are artificially intelligent systems that need to adapt themselves based on user actions and surroundings. These systems need to carefully analyze the user needs and the conditions of the surroundings in order to predict future actions and also minimizes user interaction. Researchers have been working on predicting the user actions based on past actions[1]. Such systems demand a high accuracy in their predictions, because a wrong prediction would mean user inconvenience which foils the main purpose of such systems.

A single day in a person's life consists of a set of actions. These actions, over a period of time can be learned by an intelligent system and prove useful to predict future actions. An example of such a scenario would be, a person returning home from work at 8 o clock and switching on water heater to take a bath. An intelligent system could learn this pattern and switches on the water heater at a particular time predicting the arrival of the user. Another example of a scenario is: Consider a person who switches on the light after the light intensity decreases below a particular threshold. A system could learn to do this by analyzing the user's patterns.

Existing systems use a reinforcement learning approach[2] or a pattern mining technique[3] to determine what can be the next action. Those systems contain an agent in an environment, and a feedback from the environment is given. The inputs to these systems are previous actions that have taken place. But all these systems make use of a considerable amount of hardware, for example, light sensors to determine the light intensity, humidity and temperature sensors to determine the temperature of the room at a particular instant[4]. The system proposed in this paper tries to minimize the use of any hardware resources. The aim is to maximize the capture of information at an instant to the maximum potential given the state of a room at that instant.

In this paper, a deep reinforcement learning technique is proposed to predict the rewards for each action taken. The input to this system is a snapshot of the room at an instant. The system makes use of a Convolutional Neural Network to represent the image and Q-learning technique to predict the rewards for each action that can take place. The algorithm and methodology proposed, along with the results obtained are discussed in the paper.

## 2. RELATED WORK

Home Automation has been an important research area for a long time. Research groups such as MIT,IBM,Xerox and Microsoft have been working on this, trying out various AI and non AI techniques to automate home systems[5]. In 1998, Michael C. Mozer from University of Colorado, Boulder implemented a system called Adaptive Control of Home Environments (ACHE)[6] that used neural networks and reinforcement learning completely to learn the patterns of the user and perform actions without the help of the user. It monitors the environment, observes the actions taken by the user and tries to learn the patterns. ACHE is equipped with sensors to report the state of the environment. These sensors report information like: status of lights, status of fans and status of temperature and motion. These are only a few of the sensors used. It is evident that the system makes use of extensive hardware to infer the user actions.

In 2008, A.Assim et al presented their system that uses pattern matching and reinforcement learning to predict the correct ac-

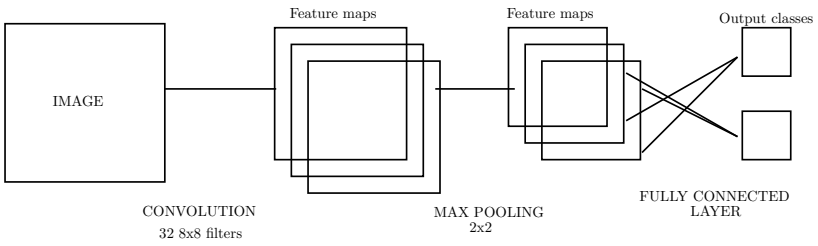


Fig. 1: A simple Convolutional Neural Network Architecture.

tions[7]. Q-learning along with pattern matching provided good improvement in results. However, the hardware issue remained the same. The states of each and every appliance and the user movement should be known in order to learn the patterns.

In 2013, DeepMind technologies published the first deep reinforcement model to make the system play Atari Games[8][9]. It used a Q-Learning model combined with a Convolutional Neural Network to predict the next action, given the image of a game at that instant.

In this paper, this approach is extended to home automation, using Deep Q-Learning to make the system understand and learn the user patterns and actions without the need for intervention of a user. This system also minimizes the amount of hardware used, as it needs only a camera to capture images.

## 3. REINFORCEMENT LEARNING

In the proposed system, the system decides what would be the best action to perform given the input as follows, the image of the room at that instant. Consider a single equipment such as light (for example), the actions that are possible are: LIGHTS_ON and LIGHTS_OFF. Normally, it would make sense to treat this problem as a typical classification problem with two output labels LIGHTS_ON and LIGHTS_OFF. However, a huge amount of data would be needed to train the system. Images would be required to tell the system what to do at each snapshot of the room, which can be enormous. Reinforcement learning solves this task. Reinforcement learning lies between supervised and unsupervised learning. It makes use of sparse and time-delayed labels - rewards. The system at each stage tries to maximize the reward. In the proposed system, deep Q- learning is used to capture as much information about the environment in a snapshot and then predict the next action. The problem can be formalized as follows:

X (the home automation system) is an agent situated in an environment (the room) which is in a certain state S (e.g. The lights may be turned on with the position of the user in the middle). X can perform certain actions A, with each action A yielding a particular reward R. Actions transform the environment and lead to a new state. The rule to decide which action to perform is called policy.

The set of all states and actions along with the rules for moving from one state to another contribute to a Markov Decision Process. One episode of this process,which is the events happening in one day in our case, gives rise to a finite sequence of states, actions and rewards:

$$st_0, ac_0, re_1, st_1, ac_1, re_2, ..., st_{n-1}, ac_{n-1}, re_n, st_n$$

Here $st_i$ represents the state, $ac_i$ is the action performed and $re_{i+1}$ stands for the reward obtained on carrying out the action. The terminal state $st_n$ represents the end of the episode (the last event for the day in our case). A Markov Decision is based on the Markov assumption that the probability of the next state $st_{i+1}$ relies only on the current state $st_i$ and action $ac_i$ but not on the previous states or actions

### 3.1 Discount Factor

For long term performance, it is essential to take into account the future rewards that will be obtained in addition to the immediate rewards. Given a single run of the Markov Decision Process, the total reward for one episode can be calculated as :

$$RE = re_1 + re_2 + re_3 + ... + re_n$$

Given that, the total future reward from time point t can be expressed as:

$$RE_t = re_t + re_{t+1} + re_{t+2} + ...re_n$$

Given that the environment is stochastic, it is not possible to assure that the same rewards will be obtained on performing the same actions. The rewards diverge as the system progresses towards the future. For that reason, it is intended to use discounted future reward instead :

$$RE_t = re_t + \gamma re_{t+1} + \gamma^2 re_{t+2} + ... + \gamma^{n-t} re_n$$

gamma is the discount factor between 0 and 1, the more into the future the reward is, the less it is taken into consideration. The discounted reward at time step t can be given in terms of the same thing at time step t+1:

$$RE_t = re_t + \gamma(re_{t+1} + \gamma(re_{t+2} + ...)) = re_t + \gamma RE_{t+1}$$

If the discount factor is set as zero, then this strategy will be shortsighted and depend only on the immediate rewards. In order to provide a balance between immediate and future rewards, it is necessary to set the discount factor to a non-zero value like 0.9. If the environment is deterministic such that the same actions result in producing the same rewards, then the discount factor can be set to 1. A good strategy is to always choose an action that gives the maximum (discounted) future reward.

### 3.2 Q-Function

To use Q-Learning, a function is defined as Q($st_t$,$ac_t$) representing the maximum discounted future reward when an action act is per-

Table 1. : Architecture of Convolutional Neural Network used

| Layer | Input Size | Filter Size | Number of Filters | Activation | Output |
|---|---|---|---|---|---|
| conv1 | 80x80x3 | 8x8 | 32 | RELU | |
| Pool1 | 73x73x32 | 2x2 | | | 36x36x32 |
| conv2 | 36x36x32 | 4x4 | 64 | RELU | 33x33x64 |
| Pool2 | 33x33x64 | 2x2 | | | 16x16x64 |
| conv3 | 16x16x64 | 3x3 | 64 | RELU | 14x14x64 |
| Pool3 | 14x14x64 | 2x2 | | | 7x7x64 |
| conv4 | 7x7x64 | 3x3 | 64 | RELU | 5x5x64 |
| Pool4 | 5x5x64 | 2x2 | | | 2x2x64 |
| FC1 | 256 | | | RELU | 256 |
| FC2 | 256 | | | RELU | 256 |
| FC3 | 256 | | | Linear | 2 |

formed in state stt :

$$Q(st_t, ac_t) = maxRE_{t+1}$$

Q($st_t$,$ac_t$) represents the maximum reward that is obtained at the end of the day if the action emphac$_t$ is performed when in state $st_t$. Given a state $st_t$, that the agent is currently in,the best action is chosen as

$$\pi(st) = argmax_{ac}Q(st, ac)$$

When a transition *($st_1$,$ac_1$,$re_1$,$st_2$)* is considered, the Q-function is given by:

$$Q(st_1, ac_1) = r + \gamma max_{ac}Q(st_2, ac)$$

### 3.3 Convolutional Neural Networks

The Q-Function is represented as a neural network. The inputs to the neural network are images of the environment at that instant and the output values are the rewards or the Q-values. Since to compute the features of the image effectively, a Convolutional Neural Network[10] is used. A CNN is similar to a normal neural network: they are made up of neurons that have weights and biases which can be learnt. But CNNs exploit the fact that its inputs are images and make changes to the architecture of normal neural networks. A CNN consists of 3 layers:

—Convolution Layer

—Pooling Layer

—Fully Connected Layer

#### Convolution Layer

The Convolution Layer can be visualized as a block or a cuboid. In our system, for efficiency purposes, the image was resized to 80x80x3. The 3 corresponds to the 3 colours Red,Green and Blue. This forms the base block of our convolution layer. Now a filter is run over our base block. A filter is a similar block having a smaller size but same depth. It starts at the top left corner and is swept across till the bottom left corner. The weighted sum is calculated at each position and a new value is obtained. The output size would be given by:

$$O_s = (N - F)/S + 1$$

where $O_s$ denotes the output size, $N$ denotes the image dimension (NxNxh),$F$ denotes the filter dimension (FxFxd) and $S$ denotes the stride, which is the number of cells to move in each step.

#### Pooling Layer

Pooling layers are used to reduce the size of the image and also provides translation invariance. Pooling is achieved again by using filters and a pooling strategy such as max pooling, mean pooling and min pooling.

#### Fully Connected Layer

This layer is the normal neural network layer. This comes after the convolution and pooling layers. It takes as input the convolved and pooled features and gives the desired output at the last layer.

## 4. DEEP Q-LEARNING

Deep Q-Learning refers to the fact that deep neural networks such as CNNs are used to represent the neural network. In the proposed system, the Q-network takes the image of the room as input and outputs the reward for each action possible. While taking the lights as example, LIGHTS_ON and LIGHTS_OFF are the two actions. From the output of the neural network, the action that gives the maximum reward is chosen. The architecture used in the system is as shown in the table.

In the architecture used by[9], pooling layer is not used since the translation invariance it provides is not needed. However, in the proposed system, translation invariance is very important. This is explained with the following example: Suppose the user is inside the room and is at position (x,y) and the system learns that it has to perform action LIGHTS_ON. However,this would also be applicable if the user is at some other position (x',y'). Hence to achieve translation invariance, pooling layers are used.

### 4.1 Cost Function

Input to the network is a 80x80x3 image and the outputs are the Q-values or rewards for each action(2 in our case). Since the Q-values are real, this can be treated as regression problem. Given transition *(st,ac,re,st')* the cost function J is given by:

$$J = 1/2 * [re + max'_{ac}Q(st', ac') - Q(st, ac)]^2$$

Using this cost function, the system is trained to predict the rewards accurately.

### 4.2 Replay Memory

To avoid local minimums, instead of using the recent transition to train the Q-network, random mini-batches are obtained from

Table 2. : Rewards obtained by the system at the end of each session

| States | Rewards obtained for LIGHTS_ON | | | | | | | | Rewards obtained for LIGHTS_OFF | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 100-200 | 200-300 | 300-400 | 400-500 | 500-600 | 600-700 | 700-800 | 800-900 | 100-200 | 200-300 | 300-400 | 400-500 | 500-600 | 600-700 | 700-800 | 800-900 |
| Day | 88.34 | 200.45 | 300.47 | 250.09 | 210.45 | 120.56 | 98.43 | 42.095 | 20.38 | 100.89 | 295.68 | 350.43 | 599.765 | 700.86 | 950.25 | 991.319 |
| Noon | 150.63 | 30.45 | 100.25 | 120.45 | 98.99 | 72.67 | 50.85 | 42.4042 | 78.87 | 40.25 | 200.78 | 285.47 | 397.55 | 934.123 | 821.987 | 1000.0101 |
| Night | 20.37 | 85.281 | 211.7543 | 295.894 | 402.772 | 635.89 | 745.238 | 999.754 | 200.85 | 357.75 | 219.54 | 135.2518 | 95.867 | 52.576 | 48.797 | 43.589 |

the replay memory.Replay memory consists of all the transitions (*st*,*ac*,*re*,*st'*) that has been encountered so far.

### 4.3 Exploration-Exploitation

Since a CNN is used to predict the Q-values for each (state,action) pair, initially the Q-values will be random numbers as the weights of the CNN are initialized randomly.In order to avoid taking wrong actions in the beginning,as this can affect the system in the long run, the system chooses random actions at the beginning. This step is known as *Exploration*.

After the Q-function converges, the rewards are more accurate. From now on, the system chooses the action that would give the maximum reward for that state.

### 4.4 Algorithm

**Algorithm 1:** Deep Q-Learning

Initialize the replay memory $M$
Initialize the Q-Function randomly
Initialize the threshold $\lambda$
Initialize loop counter $i = 0$
**while** *End of episode* **do**
  Observe the current state $st_i$
  **if** $i \leq \lambda$ **then**
    Select action $a_i$ with probability $\epsilon$
    otherwise select action $a_i$ as $a_i$ = $\text{argmax}_{ac}$Q($st_i$,ac)
  **else**
    Select action $a_i$ as $a_i$ = $\text{argmax}_{ac}$Q($st_i$,$ac$)
  **end**
  Execute action $a_i$ and observe the reward $re_i$
  Capture the new image and store it as state $st_{i+1}$
  Store transition ($st_i$,$a_i$,$re_i$,$st_{i+1}$) in replay memory $M$
  Sample random mini-batches ($st_j$,$ac_j$,$re_j$,$st_{j+1}$) from $M$
  **if** $st_{j+1}$ *is terminal state* **then**
    Set $y_j = r_j$
  **else**
    Set $y_j = r_j + \gamma * \max_{ac}(st_{j+1},ac)$
  **end**
  Train the Q-Network using target as $[(y_j - Q(st_j,ac_j)]^2$ as loss
**end**

## 5. RESULT

The architecture given in Table 1 were used to create a convolutional neural network. While training the network, a Mobile phone was used to capture the images of the environment and given as input to the algorithm mentioned above. The corresponding action predicted by the algorithm was executed and the new image was captured and stored as the next state. When the trained network was run on a given set of images, it was seen that the actions predicted by the system were 98% accurate. The obtained results were plotted as a graph of Reward vs Time. The rewards take random values initially. As the usage of an action increases with time, the reward obtained for that action also increases. It can be seen from the graph that starting at a random value, the reward for LIGHTS_ON action slowly decreases during the day and then increases during the night. Similar trend can be seen with reward for LIGHTS_OFF action, it increases during the day and decreases during the night.

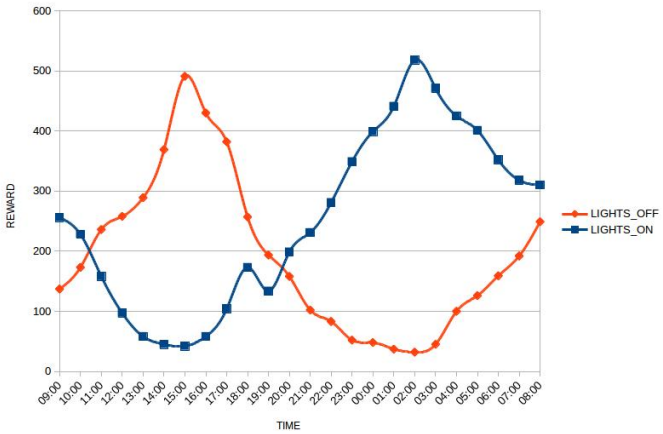


Fig. 2: Variation of Reward with Time

## 6. CONCLUSION

In this paper, the first Deep Reinforcement Learning model for Home Automation Systems is introduced. With the help of Deep Q-learning and Convolutional Neural Network, the ability of the system to predict the subsequent actions given images of the environment without the need for any sensors or hardware is successfully demonstrated. This approach yielded accurate results in the test set of images of a given environment, with no adjustment of the architecture or hyperparameters. In this paper, the system was tested based on its ability to predict actions for lights. More research work has to be performed to generalize this system to all appliances. Apart from using this model for home appliances, it can be extended to serve Industrial applications. Machines at a work environment can be automated to follow a particular work pattern, thus further reducing the required human labor. Automated door locking systems with security cameras can facilitate more security.

## 7. REFERENCES

[1] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 2012.

[2] Natalie Kcomt Ché, Niels Pardons, Yves Vanrompay, Davy Preuveneers, and Yolande Berbers. An intelligent domotics

system to automate user actions. In *Ambient Intelligence and Future Trends-International Symposium on Ambient Intelligence (ISAmI 2010)*, pages 201–204. Springer, 2010.

[3] Sajal K Das and Diane J Cook. Designing smart environments: A paradigm based on learning and prediction. In *International Conference on Pattern Recognition and Machine Intelligence*, pages 80–90. Springer, 2005.

[4] Peter Gorniak and David Poole. Predicting future user actions by observing unmodified applications. In *AAAI/IAAI*, pages 217–222, 2000.

[5] Edwin O Heierman and Diane J Cook. Improving home automation by discovering regularly occurring device usage patterns. In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, pages 537–540. IEEE, 2003.

[6] Li Jiang, Da-You Liu, and Bo Yang. Smart home research. In *Machine Learning and Cybernetics, 2004. Proceedings of 2004 International Conference on*, volume 2, pages 659–663. IEEE, 2004.

[7] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[8] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[9] Michael C Mozer. The neural network house: An environment hat adapts to its inhabitants. In *Proc. AAAI Spring Symp. Intelligent Environments*, volume 58, 1998.

[10] Mamun Bin Ibne Reaz, Awss Assim, Muhammad I Ibrahimy, Florence Choong, and Faisal Mohd-Yasin. Hardware simulation of home automation using pattern matching and reinforcement learning for disabled people. In *IC-AI*, pages 213–218, 2008.