BEST: International Journal of Management Information Technology and Engineering (BEST: IJMITE)
ISSN (P): 2348-0513, ISSN (E): 2454-471X,
Vol. 5, Issue 12, Dec 2017, 17-28

AGENT-BASED CONVOLUTION AND REINFORCEMENT LEARNING

SAMER I. MOHAMED & AMR ABDELNABI

Associate Professor, Department of Communication Engineering, Faculty of Engineering,

October University for Modern Science and Arts - MSA University, Cairo, Egypt

ABSTRACT

The problem with the current models like Darwin-OP or Boston Dynamic's ATLAS is their up-time, especially with increased number of joints. These models try mimicking the human motion; they end up using a lot of actuators, which in turn leads to the use of a lot of battery power. This paper discusses the creation of a new model of humanoid robots, that does not try to mimic the bipedal walking gait used by humans, but who instead uses a full model constructed from scratch, that consists of a model free Deep Q-Learning (DQN) algorithm, which doesn't need any walking sequence or walking models, it just learns from trial and error by applying actions on the robot and observing the reward from that action to make an under-actuated robot able to balance and walk forward, backwards, sideways, and rotate in place using only 4 actuators (two in each leg). The proposed model uses a Regional Convolutional Neural Network (R-CNN) to detect and inform the robot about the place of its goal. A full sensory system of a camera and Inertial Measurement Unit (IMU) is utilized to extract and gather the required inputs for reaching the goal from the robot' environment. Thus instead of thinking that robot as a pre-programmed entity who performs specific task, we treat this as agent who can learn to take whatever actions towards specific goal controlled by evaluation function to maximize specific reward.

KEYWORDS: Deep Q-learning, Humanoid robots & Multilayer Perceptron Deep Neural Networks

INTRODUCTION

Robotics science is in continuous improvement and development over the last decades, because of the continuously-changing busy modern life. In our proposed system, an intelligent under-actuated humanoid robot with a lower cost is introduced to help human in day-to day tasks. By applying our system methods, the robot will be able to learn how to balance, detect its goal and walk towards it. The proposed system of that intelligent humanoid robot's composed of a sensory system with a suitable processing unit to form an integrated system. This integrated system is required to feed the robot' environments state to the lower body learning algorithm (DQN) that enables the robot to learn how to balance while walking consists of two integrated blocks (Q-Learning and Neural Network [1]). And finally the Convolution Neural Network (CNN) that takes inputs from the camera to detect the surrounding objects in order to move around them towards final goal. It is only in recent years that manufacturers are making robotics increasingly available and attainable to the public and there are four previous systems that are under the category of humanoid robots with artificial intelligence like ASIMO, NAO, DARWIN and ATLAS. Thanks to the Graphical Processing Unit (GPU) the challenge of managing the infinite action search space for the robot becomes feasible.

ASIMO [2], stands for Advanced Step in Innovative Mobility is one of the humanoid robots that is delivered with special capabilities who can work in a real-life environment. ASIMO moves between two points without stopping by comparing the change between the embedded location map information and the information gathered from the environment

using the embedded sensors.

NAO [3] is an autonomous humanoid robot developed to simulate the human and is considered a well model of integrated technologies for humanoid robot that can understand, interpret and act autonomous to perform specific actions.

DARWIN-OP [4] Intelligence humanoid robot with an open Platform supported with very high processing power to fit for different applications.

ATLAS [5] is a humanoid robot given with 16 Degree of Freedoms (DoF). This robot is one of the systems that use the Deep Q-Learning/Network (DQN), to simulate the human walking via learning. ATLAS uses the reinforcement learning algorithms, to perform the optimal action based on the current state of the surrounding environment, to achieve a specific goal/objective using Q-learning mechanism.

Our proposed system uniqueness compared to other systems is two folds, first the under actuation concept that results in lower cost and power consumption, second, using Deep Q-Learning or model-free DQN algorithm. Most of other systems are not using learning as a concept, but some of these systems like ATLAS use model-based DQN for learning. That is how our proposed system outperforms those systems to execute any task via model-free learning that can be generalized in any application. The embedded system of our proposed root model is implemented based on Linux microcomputers Raspberry Pi [6] while the main learning functions and algorithms is achieved based on reinforcement learning (Deep Q-Networks) and convolution neural network using MATLAB and Python. Our humanoid robot learns how to balance and walk via Deep Q-Networks algorithm using the rewards system for each action performed by the robot towards achieving specific value function or goal. The proposed model also uses the Convolution Neural Network (CNN) for robot object detection and categorization. Thus we used the CNN that takes robot vision data or images as input and start learning to detect the object from the image. From that different parameters like distance from goal can be calculated to feed into the DQN algorithm. One of the main challenges we face with using CNN is the limiting computation processing resources on mobile robots [15] for real time operations.

The reminder of the paper is organized as follows: the system overview and high level design of the proposed system will be covered in section 2, section 3 will cover the testing results and experimental analysis and debate for the proposed system and finally the conclusion for the overall proposed system will be concluded in section 4.

PROPOSED SYSTEM DESIGN

SYSTEM OVERVIEW

Mimic the dynamics of humans walking pattern is a complex process. The learning algorithm Deep Q Network (DQN) can be broken down into two parts, the Q-learning [7] algorithm and the neural network [8]. Firstly, the Q-learning algorithm is considered special version of the reinforcement learning (RL) algorithms where agent learns by trial and error through taking actions and observing the reward reserved based on that action. The system also includes a convolution neural network (CNN) [11] and regional-convolution neural network R-CNN for the upper body to aid in knowing and detecting where the goal from the robot's perspective is. Convolution Neural Networks [10] are a widely-used type than normal type of Neural Networks especially for pattern detection and classification type of problems. CNN outperforms any other object detection algorithms via series or stack of layers where each layer is dedicated for specific action on the input image. As the system desperately needs to be customized to our specifications we didn't use any available libraries or toolboxes and wrote the code for every algorithm and neural networks from scratch to have the full control and

accessibility on every aspect of the code. Our proposed system is based on the learning algorithm that is written in MATLAB and connected through WI-FI with high processing unit (Raspberry pi). Figure 1 shows the block diagram for our proposed robot systems where sensory system composed of Camera, Inertial measurement unit (IMU) and Proximity systems grasp the state of the robot from the environment in terms of position, location, and surrounding and feed this into the processing unit (Raspberry pi) via wireless network or WI-FI. The state is then processed via DQN to enable the agent/robot to take action towards the goal. This action mapped into set of motor angles to move the robot lower body.

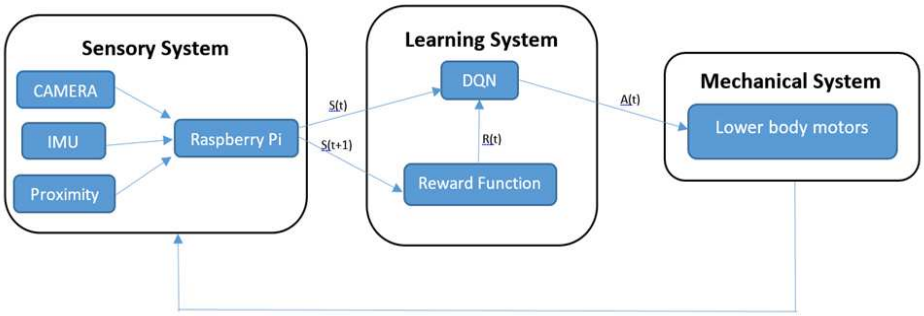

Figure 1: Robot Systems Block Diagram

The sensory system composed of 3 main elements which are the Camera, IMU (Inertia Measurement Unit, and Proximity) detailed as follows:

Camera

Mounted on the robot upper part to give the visual support for the robot to track objects, and perceive the surrounding environment around it. The camera will feed captured images (matrices in RGB mode) into the CNN for goal detection and classification.

IMU

The Inertial measurement unit used to capture the agent or robot orientation parts in the three axes space and is composed of gyroscope and the accelerometer. IMU can also, used to obtain the agent inclination to prevent robot from falling.

Proximity

The sensor is used to detect the surrounding objects and can be used to detect distance to goal, this is done via beam of electromagnetic waves/radiation.

Raspberry Pi

The raspberry pi is our processing unit which is single-board computer that we uses to run all the learning algorithms for our humanoid robot integrated system like the DQN and R-CNN.

SYSTEM DESIGN

SENSORY SYSTEM DESIGN

The data required to be collected from the sensors for the learning algorithms are distance to goal, angle away from goal, and orientation in X, Y axes. The real robot sensory system inputs are mapped to the Q-Learning state

representation. As shown in figure. 2, using a single camera which is connected to raspberry pi, we can calculate the distance to goal which is a parameter required in the state representation. The angle away from goal can be calculated using either the servo mounted below the camera which is used for rotating the camera in Z-axis or the IMU using the value of the heading (how far the goal is away in z-axis).

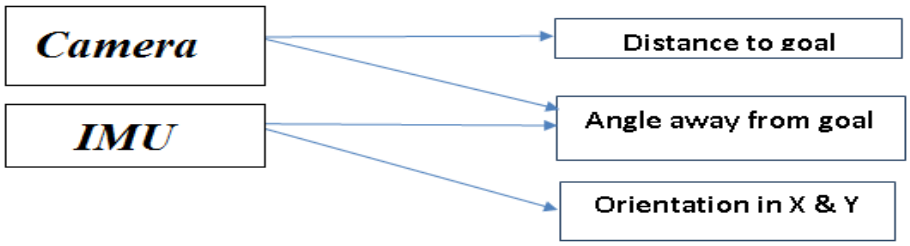

Figure 2: Sensory System Mapping to the Learning Algorithm

OBJECT DETECTION AND DISTANCE TO GOAL DESIGN

This algorithm calculates the distance to goal using a single vision camera by substituting in a function which requires the camera to take a snapshot and move closer to the goal with defined distance and take another snapshot and get the height of the object in pixels in the two snapshots, this algorithm called the duty cycle algorithm [15] as detailed in figure 3. After that we can calculate the distance to the goal using equation. (1):

$$D = M/ [1-(b/a)] \quad (1)$$

D: distance to the detected object [Required]

M: distance of the camera to be moved closer to the target

a: Object height in pixels before moving the camera

b: Object height in pixels after moving the camera

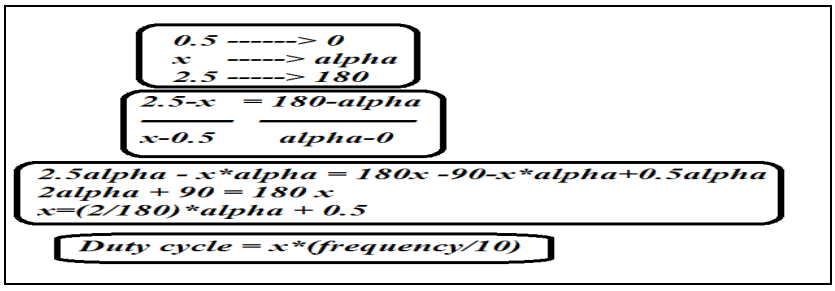

Figure 3: Duty Cycle and Angle Equation (Servo Motor)

SERVO MOTOR DESIGN

The servo will be at 0-degree position when the signal is high with 0.5ms delay, 90-degree position with 1.5 ms delay and 180-degree position with 2.5 ms delay. The leaning algorithm will send a value of angle with which the motor will rotate. So, a mathematical equation that relates the duty cycle with the angle rotation was proved as shown in figure 3. The code was tested in the command window and the servo is rotating to the desired angle perfectly.

DQN DESIGN

The neural network structure is a 2 layers' feedforward neural network each layer contains 250 neurons, we used this architecture because it gives us better results while training the robot, although a 1-layer neural network is sufficient enough to approximate most learning problems. We tried using a 1-layer neural network with 1000 neuron that gives us approximately the same result of our proposed model but it takes more time converging to this solution under same computation power used for testing the algorithm. Thus we preferred using the architecture of 2 layers, with 250 neurons per each layer.

CNN AND R-CNN DESIGN

CNNs consists like normal neural networks of input, output and hidden layers. Hidden layers mainly consist of convolution, pooling, normalization/RELU and fully connected layers detailed as follows in the following sections:

- Convolution layer: This layer is the core building block for the CNN; it breaks the input image into smaller features representing the whole image. After that, these features will undergo a filtering process in which each feature will be lined up with the image, multiply each image pixel by the corresponding feature pixel, adding them up and dividing by the total number of feature pixels, and then a map of where the feature occurred is formed for each feature. So, in convolution layer, one image becomes a stack of filtered images.
- Pooling layer: This layer shrinks the image stack by picking a window size and a stride then moving this window across the filtered images and from each window it takes the maximum value or the averaged value in case if average pooling. This is done for each filtered image to make them smaller enough to easily deal with them.
- Normalization or ReLU (Rectified Linear Units) layer: In this layer, each filtered image will be processed to transform all the negative values into zeros.
- Fully connected layer: In this layer, a list of feature values becomes a list of votes as there will be back propagation in this layer to compensate the error in the voting weights just like a regular multilayer perceptron neural network's hidden layer.

These layers can be stacked based on the application in hand starting with convolution layer towards the fully connected layer as shown in figure. 4. The used structure is the recommended structure from the MATLAB documentation [14].

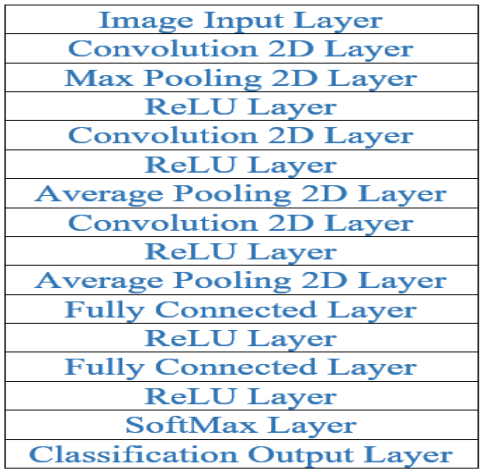

Figure 4: CNN Architecture and Design

CNN AND R-CNN WORKFLOW

Regional CNN or R-CNN [17] is considered one of the promising object detection algorithms under the CNN domain with faster and accurate results. Our proposed model utilizes the R-CNN algorithm to better detect the goal object from the captured image. Agent or robot calculates the Region of Interest (RoI) related to the goal/ object, from the image of interest then creates a warped image region, for the selected RoI, to feed later into the Convolutional network. Once each region has been forwarded to the network, bounding box regression techniques [9] are applied with classification. Where figure 4., shows the CNN architecture as used from MATLAB and the different layers used for the classification purposes, figure 5 shows the training workflow for the current CNN and Appendix equations for forward/backward propagation.

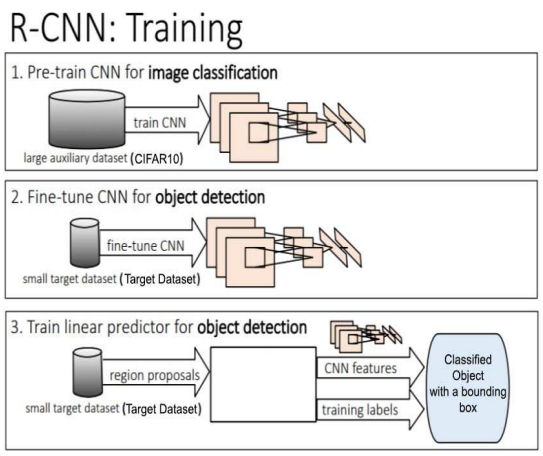

Figure 5: R-CNN Training Workflow

LEARNING OPTIMIZATION ALGORITHM

Our proposed model uses one of the optimization techniques to make the learning faster and make the algorithm coverage to a better solution like momentum algorithm. We are using stochastic or Incremental gradient descent (SGD) as per equation 2 to update the neural network thetas and back propagate the error [13]. The main strength or SGD is not based on the entire dataset to compute the gradient or each iteration. The main weakness for SGD is that in some scenarios

it oscillates around local minimum. In this problem, stochastic gradient descent oscillates around the optimum minimum and sometimes can't find the optimum and stuck in a local minimum like the picture on the left in figure. 6.

$$\theta_{w,b_i} = \theta_{w,b_{i-1}} - \delta * \frac{d\ cost}{d\theta_{w,b}} \tag{2}$$

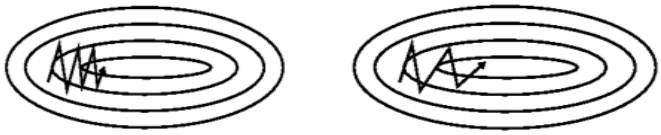

Figure 6: Momentum Optimization Logic

We used momentum [14] to fix the problem of oscillation and accelerates the SGD in the relevant direction by adding γ to the vector as shown in equation 3.

$$\mathbf{vt = \gamma vt - 1 + \eta\ \nabla\theta J(\theta),\ \theta = \theta - vt.} \tag{3}$$

Momentum adds velocity $v_{(t)}$ to the gradient ($\nabla\theta$) of the cost function $J(\theta)$, which facilitates accelerating and makes learning algorithm converges to the optimum minimum, which reduces the oscillation and make the algorithm converges faster. We also tried to better improve the learning algorithm performance, by normalizing the positive reward to 1 and all negative ones to either -1 or 0, in case no change in the reward value. This provides better performance results by limiting the error deviation and make the learning and neural network more stable with less fluctuations. The drawback of this approach, is that it makes the agent can't differentiate between the different magnitude of the reward. But the main value or strength of this approach is that it makes the algorithm more generic as it is just taking a portion of the error so it wouldn't over fit to one situation. [14]

SYSTEM TESTING

The testing strategy for our neural network parameters either for DQN reinforcement Deep Q-network or R-CNN Regional Convolution Neural Network based on having the thetas from the successful agent, and then using it to teach the other agents (neural networks) to improve the behavior of the first agent. Instead of using random exploration (epsilon greedy and simulated annealing [2]) we explored from the network that learnt already by doing feed forward with the current state. We keep still 1% totally random to give it a chance to explore and ensure that it won't over fit one solution. We tested the algorithm on 6 different goals around it. The robot managed to reach some goals and some goals were hard to reach and took too many moves as shown in figure. 7. Also, we found that this agent took a lot of time to rotate then it reached the goal.

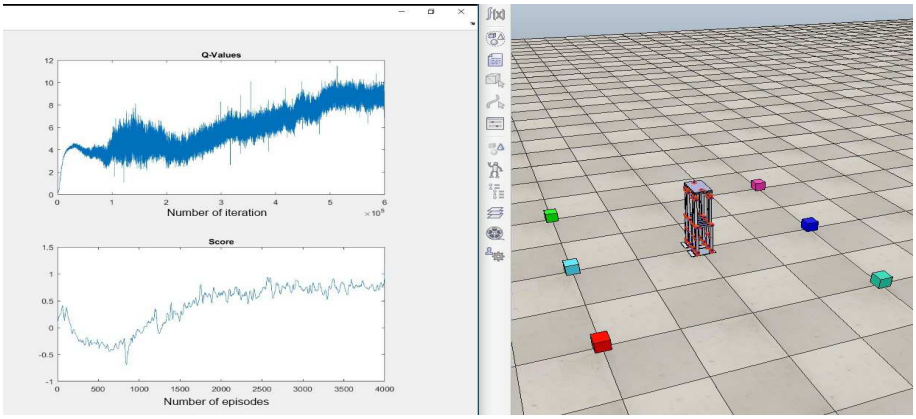

Figure 7: 6 goals testing Result

Then we built an agent with 8 goals and the task to merely rotate to face the goal, without having to walk towards the goal. We wanted to better improve the performance of the previous agent, as we wanted the robot to first rotate and when it faces the goal, it then starts to walk forward. We have done that, by removing the reward from walking forward and just give the algorithm a positive reward, when it decreases the angle away from goal. We achieved that in 2 generations, generation 1 learned to rotate and face one goal only then we took what this agent learnt and ran generation 2 to learn how to rotate and face any of the 8 goals. The robot learnt to rotate and maintain its position facing the goal successfully faster and better than the other agents with an average score of 2 over 14 thousand episode as shown in figure 8. This will let us merge the agent that can walk forward and backward with that agent that rotate and obtain better and more stable walking sequences.

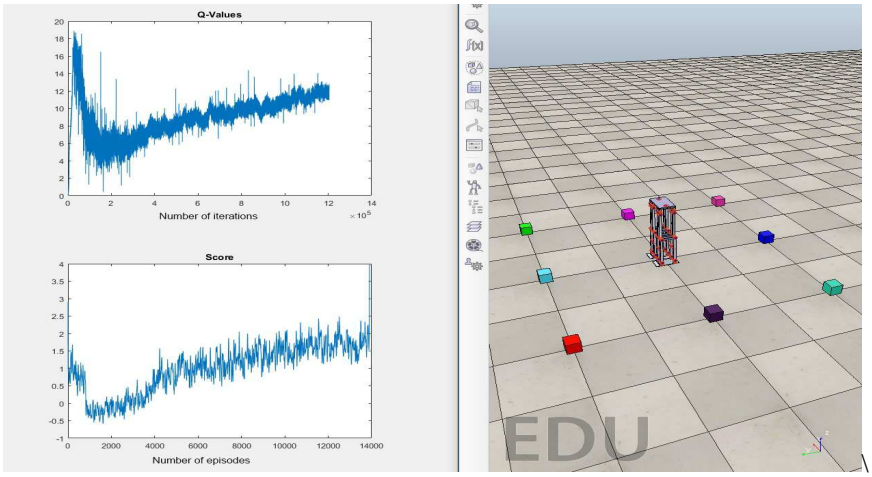

Figure 8: 8 Goals Testing Result

In the object detection part, we build using MATLAB algorithm to predict the existence of a stop sign in the inputted image from the camera, we used data augmentation to increase the number of training data, by digitally tinting the whole images set in a red and blue as shown in figure. 9, which led to better results and a more robust detector models.

Figure 9: Data Augmentation

Figure 10, shows the test was successful in various complex situations such as the object being occluded, scaled, transferred, tilted, horizontally flipped, shaky, and in grayscale. The detector model didn't output a false positive when red shiny objects were given to it as an input. It also overcame the bad quality of the camera as it doesn't have an autofocus. We obtain an output that specifies if the object exists in the inputted image and returns a bounding box of the object with respect to the image (the surrounding yellow box).

Figure 10: Object Detection Results

POWER CONSUMPTION

One of the main uniqueness for our proposed system is the under-actuation concept compared to others, the lower body of the robot managed to learn humanoid walking forward and backward, using only two servo motors. There are also 2 other servo motors that are used, for rotating the camera pan and tilt. This under-actuation concept is the key for making our power consumption less and certainly, those results in lower cost. Table 1 shows the power consumption of our robot until now.

Table 1: Power Consumption

Component	Quantity	Volt (V)	Current (A/h)	Power (Watt/Hour)
Lower forward Motors	2	6	1	12
Controller board and camera	1	5	1	5

Table 1: Contd.,				
Camera motors	2	6	0.5	6
IMU	1	3.3	350 µA	0.0012
Total				23.0012 W/h
Total without the controller and camera				18 W/h
Battery				57.5 W/h
Approximate up running time				3 Hours

CONCLUSIONS

Humanoid robot domain is currently attracted both researches and industry, in a way to make human modern life much easier and autonomous. To enable these goals and contribute in the literature and R&D of the humanoid robot, we introduced our humanoid robot with under-actuation feature, that outperforms other current systems that use lots of motors, to mimic the human gait which results in a very short online time and enormous power consumption with a costly model. Our proposed system comes as well with another feature of using a model-free Deep Q-Learning algorithm which has not been used in humanoid robots except for ATLAS that is still based on a model-based learning. Having our system as model-free provides a main benefit, where agent or robot itself has no pre solution or sequence of actions (supervised learning), to enable the robot to walk and balance, but on counter the agent learns from the interaction with the environment itself by applying actions and getting reward based on the action towards a specific goal/value function. Finally, the Convolution Neural Network (CNN) that takes inputs from the camera to detect the goals, in order to move towards them with optimized route, using the value function. Achieved learning results show a promising outcomes so far, where still further improvement as part of our future work is undergoing, to improve the learning time and accuracy of detecting the objects/ obstacles in the route towards goal.

REFERENCES

1. L. Bottou, " Large-Scale Machine Learning with Stochastic Gradient Descent ", NEC Labs America, Princeton NJ 08542, USA
2. N. Srivastava, G. Hinton, A. Krizhevsky and I. Sutskever, " Dropout: A Simple Way to Prevent Neural Networks from Overfitting", Journal of Machine Learning Research 15, Canada, June 2014.
3. MathWorks, (2017). Vision Toolbox: User's Guide (R2017a). Retrieved June 15, 2017 from https://www.mathworks.com/help/vision/examples/object-detection-using-deep-learning.html
4. 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, 47:253–279, 2013.
5. Marc G. Bellemare, Joel Veness, and Michael Bowling. Bayesian learning of recursively factored environments. In Proceedings of the Thirtieth International Conference on Machine Learning (ICML 2013), pages 1211–1219, 2013.
6. George E. Dahl, Dong Yu, Li Deng, and Alex Acero. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. Audio, Speech, and Language Processing, IEEE Transactions on, 20(1):30 – 42, January 2012.
7. Matthew Hausknecht, Risto Miikkulainen, and Peter Stone. A neuro-evolution approach to general atari game playing. 2013.

8. Nicolas Heess, David Silver, and Yee Whye Teh. Actor-critic reinforcement learning with energy-based policies. In European Workshop on Reinforcement Learning, page 43, 2012.
9. Takubo, T., Inoue, K., Arai, T.: Pushing an Object Considering the Hand Reflect Forcesby Humanoid Robot in Dynamic Walking, Proceedings of the 2005 IEEE International Conference on Robotics and Automation (2005).
10. Soft Bank Robotics. (2016, Nov. 20). WHO IS NAO? Available: https://www.ald.softbankrobotics.com/en/cool-robots/nao
11. S. Yi, B. Zhang, D. Hong and D. Lee, "Online Learning of a Full Body Push Recovery Controller for Omni-directional Walking", 2011 11th IEEE-RAS International Conference on Humanoid Robots Bled, Slovenia, October 26-28, 2011
12. "DARPA Robotics Challenge", Defense Advanced Research Projects Agency, 2015. [Online]. Available: http://www.theroboticschallenge.org/. [Accessed 16 November 2016].
13. " RASPBERRY PI 3 MODEL B ", [Online], [Accessed 10 April 2017] Available: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/
14. R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction, London, England: The MIT Press, 2012.
15. V. Mnih1, K. Kavukcuoglu1, D. Silver and A. Rusu1, “Human-level control through deep reinforcement learning ", Nature Vol. 518, February 26, 2015.
16. Volodymyr Mnih. Machine Learning for Aerial Image Labeling. PhD thesis, University of Toronto, 2013.
17. Andrew Moore and Chris Atkeson. Prioritized sweeping: Reinforcement learning with less data and less real time. Machine Learning, 13:103–130, 1993.
18. Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In Proceedings of the 27th International Conference on Machine Learning (ICML 2010), pages 807–814, 2010.
19. Jordan B. Pollack and Alan D. Blair. Why did td-gammon work? In Advances in Neural Information Processing Systems 9, pages 10–16, 1996.
20. Martin Riedmiller. Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method. In Machine Learning: ECML 2005, pages 317–328. Springer, 2005.
21. Brian Sallans and Geoffrey E. Hinton. Reinforcement learning with factored states and actions. Journal of Machine Learning Research, 5:1063–1088, 2004.
22. Pierre Sermanet, Koray Kavukcuoglu, Soumith Chintala, and Yann LeCun. Pedestrian detection with unsupervised multi-stage feature learning. In Proc. International Conference on Computer Vision and Pattern Recognition (CVPR 2013). IEEE, 2013.
23. John N Tsitsiklis and Benjamin Van Roy. An analysis of temporal-difference learning with function approximation. Automatic Control, IEEE Transactions on, 42(5):674–690, 1997.
24. Brian Sallans and Geoffrey E. Hinton. Reinforcement learning with factored states and actions. Journal of Machine Learning Research, 5:1063–1088, 2004.

25. Pierre Sermanet, Koray Kavukcuoglu, Soumith Chintala, and Yann LeCun. Pedestrian detection with unsupervised multi-stage feature learning. In Proc. International Conference on Computer Vision and Pattern Recognition (CVPR 2013). IEEE, 2013.

26. John N Tsitsiklis and Benjamin Van Roy. An analysis of temporal-difference learning with function approximation. Automatic Control, IEEE Transactions on, 42(5):674–690, 1997.

APENDECIES

Equation 1 : Max. Action

$$a_t = max_a Q^*(\phi(s_t), a; \theta)$$

Equation 2 : Forward propagation algorithm

$$\begin{aligned} a^{(1)} &= x \\ z^{(2)} &= \Theta^{(1)} a^{(1)} \\ a^{(2)} &= g(z^{(2)}) \quad (\text{add } a_0^{(2)}) \\ z^{(3)} &= \Theta^{(2)} a^{(2)} \\ a^{(3)} &= g(z^{(3)}) \quad (\text{add } a_0^{(3)}) \\ z^{(4)} &= \Theta^{(3)} a^{(3)} \\ a^{(4)} &= h_\Theta(x) = g(z^{(4)}) \end{aligned}$$

Equation 3 : Loss equation

$$loss = (r + \gamma * maxQ(s'_i, a'_i, \theta) - Q(s'_{i-1}, a'_{i-1}, \theta))^2$$

Equation 4 : Update thetas by gradient descent

$$\theta_{w,b_i} = \theta_{w,b_{i-1}} - \delta * \frac{d\, cost}{d\theta_{w,b}}$$

Equation 5 : Back propagation algorithm

$$\begin{aligned} \delta^{(3)} &= (\Theta^{(3)})^T \delta^{(4)} .* g'(z^{(3)}) \\ \delta^{(2)} &= (\Theta^{(2)})^T \delta^{(3)} .* g'(z^{(2)}) \end{aligned}$$

Equation 6 : Cost equation

$$cost = loss + L_1 \sum \theta_w + L_2 \sum \theta_w{}^{2|}$$

Equation 7 : Back propagation algorithm