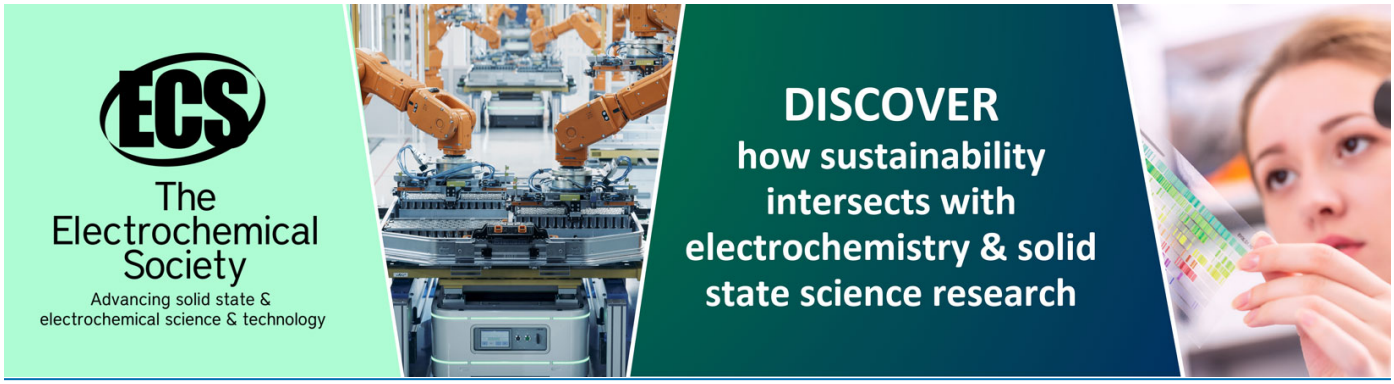
Hierarchical extreme learning machine based reinforcement learning for goal localization



IOP Conf. Series: Materials Science and Engineering **184** (2017) 012055 doi:10.1088/1757-899X/184/1/012055

Hierarchical extreme learning machine based reinforcement learning for goal localization

Nouar AlDahoul [1, a], Zaw Zaw Htike [1], Rini Akmeliawati [1]

[1] Department of Mechatronics Engineering

International Islamic University Malaysia, Kuala Lumpur, Malaysia

[a] nouar.aldahoul@live.iium.edu.my

Abstract. The objective of goal localization is to find the location of goals in noisy environments. Simple actions are performed to move the agent towards the goal. The goal detector should be capable of minimizing the error between the predicted locations and the true ones. Few regions need to be processed by the agent to reduce the computational effort and increase the speed of convergence. In this paper, reinforcement learning (RL) method was utilized to find optimal series of actions to localize the goal region. The visual data, a set of images, is high dimensional unstructured data and needs to be represented efficiently to get a robust detector. Different deep Reinforcement models have already been used to localize a goal but most of them take long time to learn the model. This long learning time results from the weights fine tuning stage that is applied iteratively to find an accurate model. Hierarchical Extreme Learning Machine (H-ELM) was used as a fast deep model that doesn't fine tune the weights. In other words, hidden weights are generated randomly and output weights are calculated analytically. H-ELM algorithm was used in this work to find good features for effective representation. This paper proposes a combination of Hierarchical Extreme learning machine and Reinforcement learning to find an optimal policy directly from visual input. This combination outperforms other methods in terms of accuracy and learning speed. The simulations and results were analysed by using MATLAB.

1. Introduction

Goal localization is one of the important tasks in indoor navigation systems. Visual data from a camera can be used, instead of traditional sensors such as GPS, to get information about the position of the agent and the goal. This observed data may be noisy and needs to be processed before being applied to a path planning algorithm.

The reinforcement learning suffers from the curse of dimensionality problem [1]. This problem appears when the agent needs to process and analyse high dimensional data (hundreds or thousands of dimensions). This problem can be solved by using one of the dimension reduction techniques. Traditionally dimensionality reduction techniques, such as Principle Component Analysis (PCA), Independent Component Analysis (ICA), and local linear embedding, are used to transform raw data

to a lower dimensional manifold. Recently, feature learning techniques have been exploited to learn good representation automatically.

The autonomy of a learning system is achieved by its ability to adapt to the changes in the environment by finding suitable representations automatically. It is so important to stop depending on manual engineering that uses hand-crafted pre-processing stages and replace it by recently proposed deep models.

Unsupervised learning of deep auto encoder network was integrated into batch-reinforcement learning in [2, 3]. The near-optimal policy was demonstrated automatically by learned feature spaces in grid-world like task. Deep fitted q-iteration was proposed for this object.

A class-based active detection model was proposed in [4]. It learns to localize object known by the system. The agent analyses the content of region to select the next best action. DeepQNetwork is used to learn the localization policy. Pre-trained convolutional neural network is utilized to extract features from the current region.

Another system that uses raw visual input data is pole balancing controller, it was proposed to learn a control policy by using reinforcement learning [5]. The deep encoder neural network is utilized for dimensionality reduction of the raw images.

A real slot car controller is able to learn the optimal actions autonomously [6]. It is based on cluster-reinforcement learning which was added to a Fitted-Q batch reinforcement learning to approximate the Q function.

An artificial deep Q-network agent was demonstrated to compete human in game playing [7]. The inputs are the pixels and the game scores. The objective is to produce actions that maximize accumulated scores.

This paper proposes a deep model that is based on Hierarchical Extreme learning machine [8] in a visual reinforcement learning task. The combination of deep learning and Reinforcement learning is very fruitful to have a complete system that gets visual data as input and gives optimal policy as output. Fig. 1. Shows the block diagram of the proposed system. The deep learning is used to learn suitable features from high dimensional data. The model free reinforcement learning is utilized to learn the optimal policy.

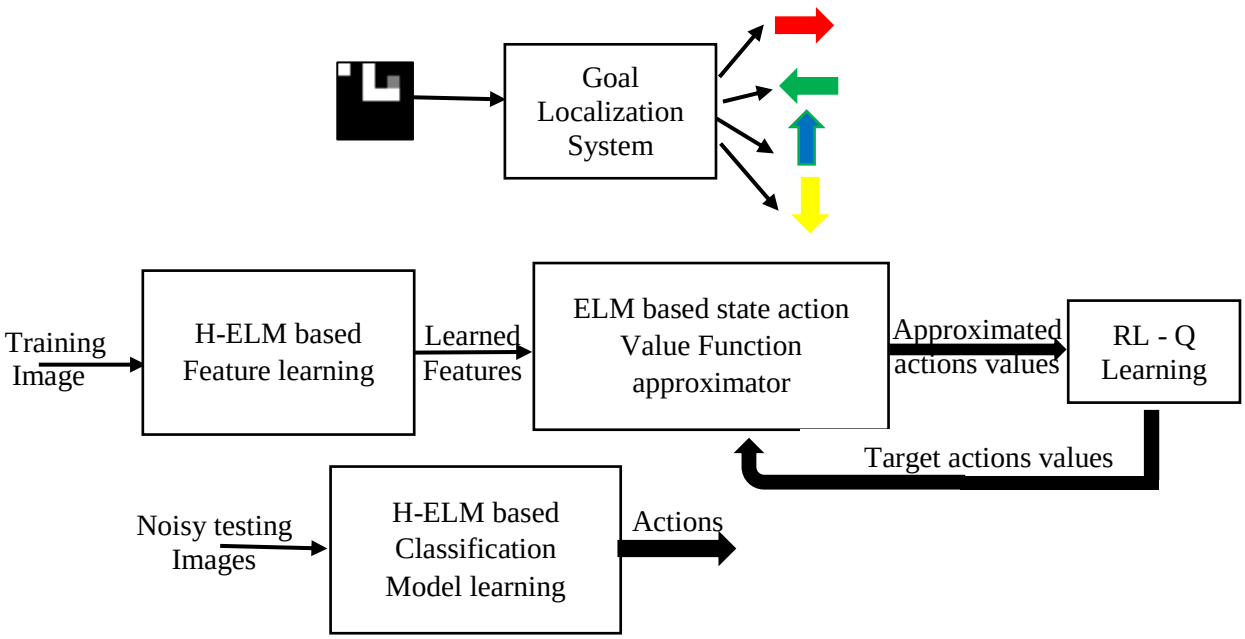


Figure 1. Block diagram of the proposed system

2. Methodology

2.1 Extreme Learning Machine for classification and regression

Extreme learning machine (ELM) is a feedforward neural network that has one hidden layer. The high generalization and high speed learning are behind the success of this learning method [9]. The biases and weights of the hidden layers are set randomly but the weights of output are calculated analytically.

$$f\ (x) = \sum_{i=1}^{L} F_i(x, W_i\ , b_i).\ \beta_i \quad ,\ W_i \in R^d \quad ,\quad b_i\ , \beta_i\ \in\ R \tag{1}$$

Where $F_i\ (\ \cdot\ )$ is the activation function of i th hidden node, W_i is the input weight, b_i is the bias, and β_i is the weight of output, L is the number of nodes in the hidden layer.

$$\beta\ =\ \mathrm{H}^{\dagger}\mathrm{T} \quad ,\qquad \beta\ =\ H^T\ (\frac{1}{\lambda} + H\ .H^T\ )^{-1}\ .T \tag{2}$$

Where H is the output matrix of the hidden layer, $H^{\dagger}$ is the Moore–Penrose generalized inverse of H, T is the matrix of target, and λ is the regulation coefficient.

2.2 Hierarchical ELM for feature learning

When dealing with visual data such as images, the deep architecture of extreme learning machine is required [8]. This architecture can achieve self-taught feature learning by unsupervised elm-based sparse encoder. H-ELM gives better generalization and less learning time. The elm-based sparse encoder is built by using fast iterative shrinkage-thresholding algorithm (FISTA). This encoder is used as a basic component of the H-ELM. Deep architecture can be achieved by stacking multiple encoders. It guarantees better data recovery and reduces the testing time by reducing the number of neural nodes. Return to H-ELM paper [8] for more details.

2.3 Reinforcement Learning

Reinforcement learning is an important learning method that focuses on how agents should take optimal actions in the environment to maximize the discounted cumulative reward [1]. See Equation 3.

$$R = \sum_{t=0}^{\infty} \gamma^t\ r_{t+1} \tag{3}$$

Where $0 < \gamma < 1$ is the discounted factor.

RL framework is formulated as a Markov decision process (MDP). The main difference between the traditional learning and the model free reinforcement learning is that there is no need to have a prior knowledge about the model of the environment.

The main components of the reinforcement learning model are:

1. Environment states S.
2. Environment observations O.
3. Agent actions A.
4. Relation between states (S_t, a_t, S_{t+1}).
5. Reward R.

One of the most famous and used RL algorithms is Q-learning. It is a model free RL algorithm. Its core is a value iteration update. The value function is found by Equation. 4. The optimal policy can be found by Equation. 5.

$$Q\ (s\ , a) =\ Q\ (s\ , a) +\ \alpha\ (R +\ \gamma\ \ Q\ (s'\ , a') -\ Q\ (s\ , a)) \tag{4}$$

$$\pi\ (\mathrm{s}) = arg_a \max\ \big(Q(s, a)\big) \tag{5}$$

Where Q is the value function, α is the learning rate.

2.4 The proposed architecture

The proposed architecture is a combination of H-ELM and Q-learning RL. The H-ELM is used to learn good features so as to find representations for input images. The Q-learning is utilized to use the learned features as states in MDP model to produce optimal actions.

The main steps of the proposed method are as follows:

1) **Initialization** Initialize the weights W_0 of the encoder's hidden layers randomly. Initialize episode counter c and sample counter s. Initialize value function Q (s, a) = 0.
2) **Exploration** explore the environment by using ϵ-greedy policy to collect training data (observations O_t). These observations are used as inputs to H-ELM based deep model. Increment s with each new observation.
3) **Encoding** transfer all training data from observation space to feature space by applying auto encoder on the observations to get feature vectors Z_t , $Z_t = \text{Encode}(O_t)$.
4) **Reinforcement Learning** Find the state action value for each feature vector by using Equation4 and by utilizing supervised ELM based value function approximator. If the convergence is achieved, return the approximated value function, greedy policy and the encoder. Else, repeat the steps from 2 to 4.
5) **H-ELM based classification** after mapping between the observations and the values of actions, the H-ELM based classifier is used to map each noisy observation with the best action results from the previous steps.

3. Experimental Results

3.1 The environment description and simulation

In this goal localization task with noisy image, the agent observes an image with 30*30 = 900 pixels. Gaussian noise is added to the images with zero mean and variance $\sigma = 0.1$. The agent should localize the goal region with a few number of moves (shortest-path problem).

The MDP (Markov Decision Process) model of the environment contains: a reward of 0 is given for any movement not leading to the goal area or obstacles. A reward of -1 is given to any movement colliding with obstacles. A reward of 1 is given to a goal ending movement. The rules of transition between different states are also given. Four actions are performed to move the agent to 4 directions: right, left, up and down.

The number of raw observations (images) is determined by the number of agent's positions in the image. In our case, there are 36 observations. Fig. 2. Shows the 36 observations without (a) and with (b) Gaussian noise. The training and testing data consists of 3600 noisy images for training and 3600 for testing. For each agent position, there are 100 images.

3.2 Accuracy analysis

The proposed method gives high performance with 100 % accuracy. Fig. 3. Shows the confusion matrix of the testing data. After finding efficient features, reinforcement learning is applied to find optimal actions. Fig. 4. Shows the average predicted state value. The convergence is achieved after only few steps.

The accuracy of different architectures of H-ELM (different number of nodes in the hidden layers) is shown in Table1. In Table 2, the comparison between H-ELM and PCA features is demonstrated. H-ELM seems to produce better accuracy with the same number of features. The Fig. 5. Displays the curve of PCA Eigen values.

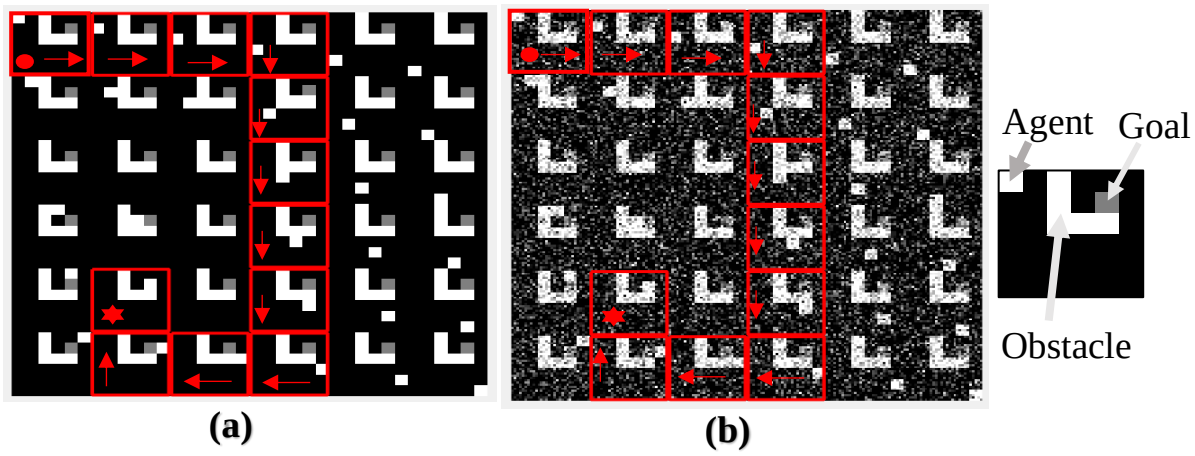


Figure 2. Training and testing images
a) Images without noise: red boxes refer to optimal actions from start (closed circle) to the goal (*), grey square represents the goal region.
b) Noisy images used to test the robustness of the system.

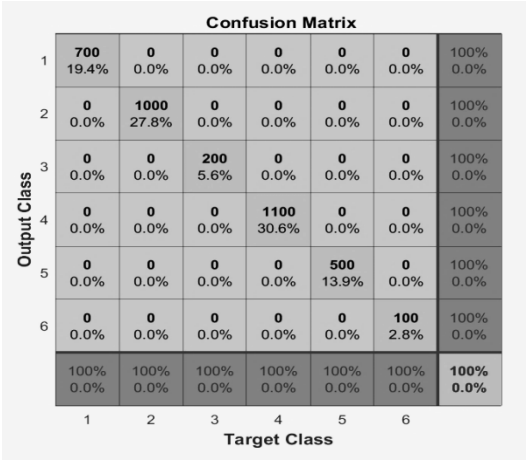


Figure 3. Confusion matrix of testing data

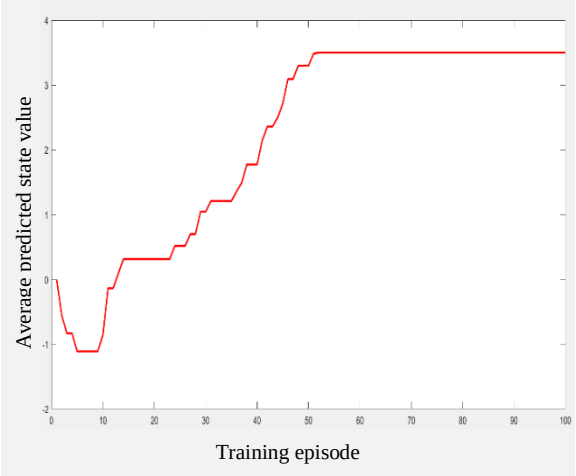


Figure 4. Average predicted state value curve

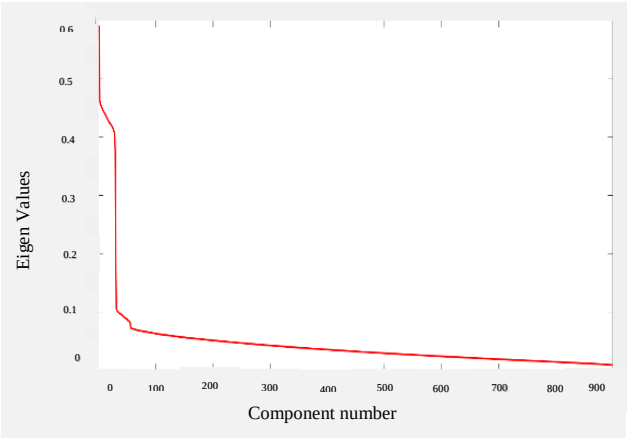


Figure 5. PCA Eigen values curve.

Table 1. Accuracy of testing noisy images with different model architectures

Number of nodes in first and second hidden layers	Accuracy %
500-200	99.97
500-100	99.94
500-50	99.97
500-20	97.63
500-10	97.47
500-7	96.52
500-5	87.91
500-3	66.11
200-3	69.88
400-3	76.52
400-5	90.83
400-7	95.72
400-10	97.02
400-100	100
400-200	100

Table 2. Accuracy comparison between PCA and H-ELM with different number of features

Number of features	PCA Accuracy %	H-ELM Accuracy %
50	100	**100**
20	97.17	**98.83**
10	97.08	**97.94**
5	93.61	**93.63**
3	71.33	**76.52**

3.3 Speed analysis

The combination of Hierarchical Extreme learning machine and Reinforcement learning outperforms other methods in terms of learning speed. Table 3 shows the training time of H-ELM and traditional stacked auto encoder [10]. The RL needs many episodes before achieving the convergence. From Fig.4, it is clear that the number of required episodes is almost 50. In each episode, the data samples are trained. Therefore the total time is the number of episodes multiplied by the training time in each episode.

Table 3. Training time comparison between H-ELM and traditional stacked auto encoder

Method	Training time in each episode (s)	Total Training time (s)
Stacked Auto encoder [10]	71	3550
H-ELM	5	250

4. Conclusion and future work

This paper proposed a combination of Hierarchical Extreme learning machine and Reinforcement learning. This combination is fruitful to learn optimal policy directly from visual input. This combination outperforms other methods in terms of accuracy and learning speed.

The advantages of the proposed architecture are:

1) Suitable action is produced when observation is forwarded through the network. This is important to reduce the testing time.
2) The use of H-ELM doesn't require to fine tune the encoder's weights iteratively and as the consequence, it reduces the time of learning and training.

This paper only focuses on the static images and does not take the relationship between individual frames into consideration. Future work should focus on this point to use dynamic information of the system.

5. References

[1] Sutton R S and Andrew B G 1998 Reinforcement Learning: An Introduction (*MIT Press, Cambridge, MA).*
[2] Lange S and Riedmiller M 2010 Deep Auto-Encoder Neural Networks in Reinforcement Learning *The International Joint Conference on Neural Networks (IJCNN), Barcelona.*
[3] Lange S and Riedmiller M 2010 Deep Learning of Visual Control Policies *ESANN, 18th European Symposium on Artificial Neural Networks.*
[4] Caicedo J C and Lazebnik S 2015 Active Object Localization with Deep Reinforcement Learning *The IEEE International Conference on Computer Vision (ICCV).*
[5] Mattner J, Lange S and Riedmiller M 2012 Learn to Swing Up and Balance a Real Pole Based on Raw Visual Input Data *In Neural Information Processing* **7667** 126-133.
[6] Lange S, Riedmiller M A and Voigtländer A 2012 Autonomous reinforcement learning on raw visual input data in a real world application *the International Joint Conference on Neural Networks (IJCNN), Brisbane, Australia.*
[7] Mnih V and et al 2015 Human level control through deep reinforcement *Nature* **518** 529–533.
[8] Tang J, Deng C and Huang G B 2015 Extreme Learning Machine for Multilayer Perceptron *IEEE Transactions on Neural Networks and Learning Systems* **27** 809 - 821.
[9] Huang G B, Zhu Q Y and Siew C K 2006 Extreme learning machine: Theory and applications *Neurocomputing* **70** 489–501.
[10] Hinton G E and Salakhutdinov R R 2006 Reducing the Dimensionality of Data with Neural Networks *Science* **313** 504-507