IJRCS International Journal of Robotics and Control Systems
Vol. 1, No. 1, March 2021, pp. 66-74
ISSN 2775-2658
http://pubs2.ascee.org/index.php/ijrcs

An Optimally Configured HP-GRU Model Using Hyperband for The Control of Wall Following Robot

Abdul Rehman Khan [a,1], Ameer Tamoor Khan [b,2,*], Masood Salik [c,3], Sunila Bakhsh [d,4]

[a] Department of Computer Science, Pakistan Institute of Engineering and Applied Science, Islamabad, Pakistan
[b] Department of Computing, Hong Kong Polytechnic University, Hong Kong
[c] Department of Electrical Engineering, Pakistan Institute of Engineering and Applied Sciences, Islamabad, Pakistan
[d] Department of Physics, Balochistan University of Information Technology, Engineering and Management Sciences, Quetta, Pakistan
[1] abdulrehmankhan27061998@gmail.com, [2] drop-in@atkhan.info, [3] masoodsalik547@gmail.com,
[4] sunila.bakhsh@buitms.edu.pk
* Corresponding Author

ARTICLE INFO

Article history
Received 07 March 2021
Revised 09 March 2021
Accepted 10 March 2021

Keywords
Wall Following Robot;
GRU;
Hyperband;
Hyper-parameters;
RNN;
Robotics;
Control

ABSTRACT

In this paper, we presented an autonomous control framework for the wall following robot using an optimally configured Gated Recurrent Unit (GRU) model with the hyperband algorithm. GRU is popularly known for the time-series or sequence data, and it overcomes the vanishing gradient problem of RNN. GRU also consumes less memory and is computationally more efficient than LSTMs. The selection of hyper-parameters of the GRU model is a complex optimization problem with local minima. Usually, hyper-parameters are selected through hit and trial, which does not guarantee an optimal solution. To come around this problem, we used a hyperband algorithm for the selection of optimal parameters. It is an iterative method, which searches for the optimal configuration by discarding the least performing configurations on each iteration. The proposed HP-GRU model is used on a dataset of SCITOS G5 robots with 24 sensors mounted. The results show that HP-GRU has a mean accuracy of 0.9857 and a mean loss of 0.0810, and it is comparable with other deep learning algorithms.

1. Introduction

Autonomous robots play a vital role in several commercial and domestic applications. These robots are widely used in military factories, secular power plants, chemical industries, and locomotive industries. One of their kind is a wall following robot, which plays a vital role in detecting faults in machinery, cracks in infrastructure, perform rescue activities, and are used in medical and rehabilitation centers [1-4]. The accurate control of these autonomous robots is necessary for their robust performance and the surroundings' safety.

There are many classical to modern state-of-the-art methods proposed for the controlled motion of wall-following robots [5-13]. Juang et al. [5] proposed a fuzzy control for the hexapod robot trained through differential evolution. Dash et al. [7] used a dataset of 24 ultrasonic sensors mounted on SCITOS G5 robot and applied a neural network (NN) to train a model to design control with an accuracy of 92.67%. Later, Dash et al. [9] proposed another hybrid model composed of gradational search approach with feedforward neural network and achieved an accuracy of 86.38% within 0.28 sec. Likewise, Dash et al. [10] proposed another approach known as Adaptive Resonance Theory-1 for the control of a wall-following robot with an average accuracy of 91.78%. Some heuristic techniques are also employed for the control of the robot [14, 15]. For instance, Chen et al. [16] used an optimization-based meta-heuristic approach known as PSO and achieved a maximum accuracy of 98.8%. Likewise, Isaac et al. [17] compared the performance of Bayesian and k-NN networks in a dynamic environment with the accuracy of 93.3% and 73.3%.

The Recurrent Neural Networks (RNNs) are popularly known for the time-series data since they can understand the contextual information stored in sequential data. RNNs are popularly used in several real-world applications [18-25]. Likewise, they have used in the control of the wall the following robot as well. Hammad et al. [26] employed LSTM and GRU models (two variants of RNN) for the control of the robot and obtained an accuracy of 96.15% and 96.52%.

In this paper, we presented an optimally configured HP-GRU (Hyperband Gated Recurrent Unit) model for the robust control of the wall following the robot. In all the models discussed above, the hyper-parameters are manually chosen through hit and trial because the selection of hyper-parameters is an optimization problem. We employed a Hyperband algorithm to select optimal parameters from the search space for GRU and achieved a robust control framework for the wall-following robot. In the simulation, the dataset is taken from a SCITOS G5 robot with 24 ultrasonic sensors mounted on it [27]. The dataset contains four moves of the robot based on the sensory information. We used our proposed HP-GRU model on the dataset and achieved a mean accuracy of 98.58%. We compared the results with other methods to show the superiority of our optimally configured GRU model.

The rest of the paper is as follows. In section 2, we will discuss the architecture of RNN and, more particularly, GRU. In section 3, we will discuss in detail Successive-Halving and Hyperband Algorithm. Besides, we will also discuss our proposed optimally configured GRU model, and finally, we will discuss the nature of the dataset collected from SCITOS G5. In section 4, we will discuss the numerical results and the comparison with other algorithms. In section 5, we will conclude the paper with the final remarks.

2. Gated Recurrent Unit (GRU-RNN)

RNNs are known for sequential or time-series data. They are an extension of neural networks (NN), where several NNs are stacked together and share the common weights. RNNs overcome the limited input length limitation of NNs but still faces the vanishing gradient problem. There are two solutions to come around this problem, i.e., LSTM (Long Short Term Memory) and GRU (Gated Recurrent Units). Both have their pros and cons, but GRU consumes less memory and is faster than LSTM. The comparison between LSTM and GRU is given below,

- LSTM processes longer input-sequence than GRU.
- GRU has two memory gates, whereas LSTM has three memory gates.
- GRU model includes lesser trainable variables than LSTM.
- GRU are computationally and time-wise faster than LSTM.

At input, it includes an $\boldsymbol{X_t}$ and a cell state $\boldsymbol{C_{t-1}}$ and at the output, it has cell-state $\boldsymbol{C_t}$ for the next GRU unit. The formulation of GRU is given as,

$$\boldsymbol{z_t} = \sigma(\boldsymbol{W_z}[\boldsymbol{C_{t-1}}, \boldsymbol{X_t}]) \quad (1)$$

$$\boldsymbol{r_t} = \sigma(\boldsymbol{W_r}[\boldsymbol{C_{t-1}}, \boldsymbol{X_t}]) \quad (2)$$

$$\widehat{\boldsymbol{h}}_{\boldsymbol{t}} = \tanh(\boldsymbol{W}[\boldsymbol{r_t} * \boldsymbol{C_{t-1}}, \boldsymbol{X_t}]) \quad (3)$$

$$\boldsymbol{h_t} = (1 - \boldsymbol{z_t}) \times \boldsymbol{C_{t-1}} + (\boldsymbol{z_t} \times \widehat{\boldsymbol{h}}_{\boldsymbol{t}}) \quad (4)$$

where $\boldsymbol{r_t}$ and $\boldsymbol{z_t}$ are the intermediary matrices for the reset and update gate, respectively. The architecture of GRU is shown in Fig. 1. It shows the reset and update gate. These gates control the flow of information through the GRU cell so that relevant information passes on to the next cell and discard the useless information. This technique helps the GRU-RNN model to overcome the vanishing gradient problem.

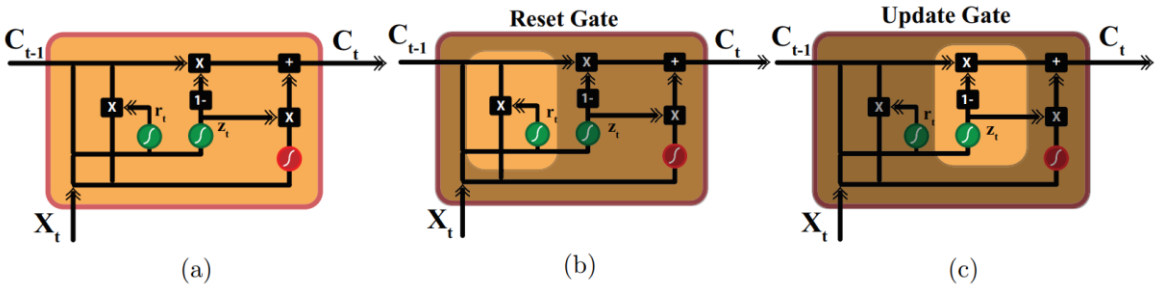

Fig. 1. (a) shows the schematic of the GRU model, which is similar to LSTM, but with fewer gates and trainable parameters. (b) shows the compact form of input and forget gates of LSTM as a reset gate, (c) shows the update gate of GRU, which will pass the information to the next GRU unit.

3. Hyperband Algorithm (HP)

3.1. Successive-Halving

There are numerous classical approaches for the selection of hyper-parameters of learning models. Bayesian optimization methods are at the top of the list, with their probabilistic approach to configure the optimal configuration. For highly complex non-linear problems, they fail to optimize the selection of hyper-parameters, so they are integrated with heuristic approaches to come around this problem.

Successive-halving is a modern approach, and as suggested from the name, it allocates the resources R to all possible configurations and then evaluates their performances along with time-consumption. The half-best configurations move to the next iteration, while the remaining are drop. This iterative process continues until the best configuration of hyper-parameter is achieved. in the beginning, successive-halving allocates uniform resources to all the configurations, let us say, $\frac{R}{n}$, where $n \in \boldsymbol{R}^+$. With time, it allocates exponential resources to the best configurations. The selection of n itself is an optimization problem. For instance, if n is small, more resources (time) will be allocated to the configurations, which may not achieve the optimal design. Whereas, if n is large, fewer resources will be assigned to the configurations resulting in premature convergence.

3.2. Hyperband

Hyperband is an extension of Successive-Halving, which addresses the "n Vs. $\frac{R}{n}$." It computes the performance of the model for different values of n. The algorithm includes two nested loops, the inner loop performance successive halving for the given value of n, whereas the outer-loop tries different values of n to find the optimal value. The outer loop is known as the "bracket," where each bracket utilizes R resources.

The algorithm is shown in Fig. 2. It takes two inputs, R_{max}, the maximum resource allocates to one configuration and, α, which is the proportion of configurations to discard on each iteration. Likewise, it includes $i_{max} + 1$ different values of n.

Algorithm 1 Hyperband Algorithm

Input: R_{max}, α %Default Value: $\alpha = 3$
Output: OC_{best} %The Best Optimial Configuration
Initialize: $i_{max} = R_{max}(log_{\alpha})$,
$R = R_{max}(i_{max} + 1)$.
for i $\in \{i_{max}, i_{max} - 1, \ldots 0\}$
$n = \frac{R\alpha^i}{R_{max}(i+1)}$, $t = R_{max}\alpha^{-s}$.
In this iteration (n, t) are the parameters for Successive-Halving.
$OC = \text{get\_hyperparameter\_config}(n)$
for $j \in \{0, 1, 2, \ldots i\}$
$n_j = n\alpha^{-i}$
$t_j = t\alpha^i$
$L = \text{return\_val\_loss}(oc, t_j)$, $oc \in OC$
$OC = \text{get\_K\_config}(C, L, \frac{n_j}{\alpha})$
end for
end for
return Configuration with leastvalidation loss OC_{best}

Fig. 2. Hyperband Algorithm

For the implementation of Hyperband, we employed a `keras-tuner`. It takes different configurations of training models, e.g., number of layers, number of hidden units in each layer, activation function, dropout %, and learning-rate, etc. The user also inputs the number of trials N and the number of executions in each trial is M. Table 1 and Table 2 shows hyper-parameters configuration space and also show the optimal configuration of hyper-parameters for the GRU model. We tuned the HP-GRU model based on the validation loss. The model will the smallest loss will be the best to use. Fig. 3 shows the results that the worst model has the loss of 0.5275, best model has a validation loss of 0.1104, so the optimally configured model is almost five times better than the worst configured model.

Table 1. Hyperband (HP): Optimal Model Selection

Model Parameters	Configurations Spaces		
	Choices	***Minimum***	***Maximum***
Number of Units (GRU)	-	32	512
Number of Units (Dense)	-	0	128
Dropout (%)	-	0	0.3
Activation Function	sigmoid, relu, tanh	-	-

Table 2. Hyperband (HP): Optimal Model Selection

Hyperband Options				
Total Trials	10	Executions/Trial	5	
Optimal Configuration				
Validations Loss	0.110	Total Trainable Parameters	1,608,292	
Model Summary				
Layer Name	***Type***	***Shape/ Units***	***Dropout (%)***	***Activation Function***
Input Layer	Input	(5455,4)	-	-
Layer_1	GRU	448	-	relu
Layer_1	Dropout	-	10	-
Layer_2	GRU	384	-	relu
Layer_2	Dropout	-	30	-
Layer_3	GRU	96	-	sigmoid
Output Layer	Output	(4)	-	softmax

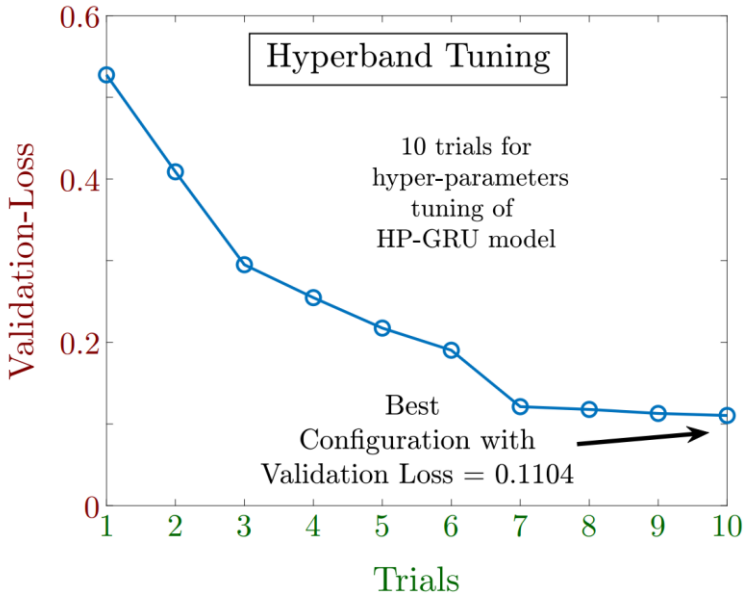

Fig. 3. HP-GRU is tuned based on validation loss. It shows that the worst model has a validation loss of 0.5275 best model has a validation loss of 0.1104.

3.3. Proposed Method

As mentioned above, the optimal configuration of the GRU model is selected through the hyperband algorithm. The details of hyper-parameters of each layer are given as follows.

3.3.1. Input Layer

The input layer consists of four input features extracted from the ultrasound sensors mounted on the wall following the robot, i.e., S_f (Front Sensor), S_l (Left Sensor), S_r (Right Sensor), and S_b (Back Sensor). The format of the input is (b_s, f)=(5455, 4), where b_f is batch size and f is features.

3.3.2. Layer 1

The first layer includes 448 hidden units with a "relu" activation function followed by a 10% dropout layer, which means that the network will randomly discard 10% of hidden units before passing the cell state $\boldsymbol{C_t}$ to the second layer.

3.3.3. Layer 2

The second layer includes 384 hidden units with a "relu" activation function followed by a 30% dropout layer, which means that the network will randomly discard 30% of hidden units before passing the cell state $\boldsymbol{C_t}$ to the flatten layer. The flattened layer flats the hidden units and connect them with the fully connected layer, i.e., the fourth layer.

3.3.4. Layer 3

The third layer includes 96 hidden units with a "sigmoid" activation function, and then it passes the output to the final and output layer.

3.3.5. Output Layer

The last and the output layer consists of 4 possible outputs for the wall following robot, i.e., Slight-Right-Turn, Move-Forward, Sharp-Right-Turn, and Slight-Left-Turn. To keep the output between 0 and 1, we used the "softmax" activation function.

3.4. The Datasheet

The dataset is recorded with the help of the SCITOS G5 robot navigated in a room in the clockwise direction. The 24 sensors are attached to the robot's waist to measure its distance from the walls. The sampling rate is nine samples per second during the four rounds of the robot in the room.

The dataset of 24 sensors is then compressed into four parts, i.e., Front Sensor (S_f), Right Sensor (S_r), Left Sensor (S_l), and Back Sensor (S_b) known as features, and based on these features, the robot can take four decisions, i.e., Slight-Right-Turn, Move-Forward, Sharp-Right-Turn, and Slight-Left-Turn, known as classes. There are mainly two data preprocessing techniques are employed, i.e., min-max normalization and conversion of classes into numerical numbers. All the data is normalized between 0 and 1 using min-max normalization, and since we used "Sparse Categorical Cross-entropy," so we assigned a number to the classes, i.e., Slight-Right-Turn = 0, Move-Forward = 1, Forward, Sharp-Right-Turn = 2, and Slight-Left-Turn = 3. The sample of the dataset is shown in Table 3. The dataset is divided into three portions, i.e., training, validation, and test. As mentioned earlier, the dataset contains 5455 samples, and it is divided as, the training data is 65% of the total data, validation data is 10%, and training data is 25% of the total dataset.

Table 3. Few Sample from Dataset

	Features				
Sample No	$\boldsymbol{S_f}$	$\boldsymbol{S_r}$	$\boldsymbol{S_l}$	$\boldsymbol{S_b}$	***Class***
1	0.264	0.022	0.359	0.013	0
10	0.053	0.027	0.355	0.019	1
20	0.070	0.030	0.207	0.022	1
30	0.253	0.028	0.209	0.021	0
40	0.241	0.028	0.198	0.030	0
238	0.2034	0.0354	0.160	0.104	2
5543	0.145	0.270	0.103	0.415	3

4. Results and Discussion

In the simulation section, we will explore the accuracy of our HP-GRU model. There are some additional parameters for the training phase mentioned in Table 4. First, we will discuss the accuracy of all the models tested during the best model selection through hyperband. The results are shown in Table 5. It shows the performance of the best HP-GRU model to the worst HP-GRU model. There are four performing metrics, i.e., validation loss, validation accuracy, test loss, and test accuracy. It can be seen that the best model, the first model, has out-performed the rest in all metrics because hyperband manages to converge the hyper-parameters of the

GRU model to the optimal configuration. It can also observe that the validation accuracy of HP-GRU models during ten trials increased from 0.7707 to 0.9659, and likewise, the test accuracy increased from 0.7780 to 0.9857, which is almost 1.25 times. Likewise, the test loss decreased from 0.5169 to 0.0810, which is 6.3 times.

Table 4. Additional Hyper Parameters

Additional Hyper-parameters	Value/Method
Optimizer	Adam
Loss Function	Sparse Categorical Cross-Entropy
Batch Size	32
Total Epoch	10
Weight Initialization	Xavier Initialization

Table 5. Additional Hyper Parameters

Models	Validation Loss	Validation Accuracy	Test Loss	Test Accuracy
1	0.1104	0.9659	0.0810	0.9857
2	0.1129	0.9415	0.0883	0.9766
3	0.1178	0.9390	0.0967	0.9661
4	0.1212	0.9610	0.1032	0.9700
5	0.1903	0.9195	0.1627	0.9413
6	0.2174	0.9122	0.1674	0.9560
7	0.2547	0.8927	0.2178	0.9403
8	0.2950	0.8707	0.2391	0.9071
9	0.4088	0.7829	0.3702	0.8302
10	0.5205	0.7707	0.5169	0.7780

We used the optimal HP-GRU model, the first model, for comparison with other state-of-the-art methods. For comparison, we used DFNN with Weight Sharing [26], DFNN (3 Hidden Layers) [26], FNN (1 Hidden Layer) [26], Gated Recurrent Unit (GRU) [26], and Long Short Term Memory (LSTM) [26]. The comparison is shown in Table 6. It shows that HP-GRU has higher accuracy of 0.9857 and a lower loss of 0.0810 as compared to other methods.

Table 6. Additional Hyper Parameters

Metrics	HP-GRU	DFNN [26] (Shared Weights)	DFNN [26] (3 Hidden Layer)	FNN [26] (1 Hidden Layer)	GRU [26]	LSTM [26]
Accuracy	**0.9857**	0.9680	0.9250	0.9010	0.9652	0.9615
Loss	**0.0810**	0.0990	0.1018	0.1089	0.0989	0.0995

5. Conclusion

In this paper, we presented an optimally configured Gated Recurrent Unit (GRU) model with a hyperband algorithm to design the control framework of the wall-following robot. GRU is a variant of RNN networks used for time-series or sequence data. With the help of hyperband, we selected the optimal configuration of hyper-parameters of our GRU model. The proposed HP-GRU model is used on a dataset of SCITOS G5 robots with 24 sensors mounted. There are four decision classes, i.e., Slight-Right-Turn = 0, Move-Forward = 1, Forward, Sharp-Right-Turn = 2, and Slight-Left-Turn = 3, and the dataset is normalized before the training. The results show that HP-GRU has a mean accuracy of 0.9857 and a mean loss of 0.0810, and it is comparable with other deep learning algorithms.

References

[1] N. P. Varma, V. Aivek, and V. R. Pandi, "Intelligent wall following control of differential drive mobile robot along with target tracking and obstacle avoidance," in 2017 International Conference on

Intelligent Computing, Instrumentation and Control Technologies (ICICICT), pp. 85–91, IEEE, 2017. https://doi.org/10.1109/ICICICT1.2017.8342539

[2] A. Ma'arif, A. A. Nuryono, et al., "Vision-based line following robot in webots," in 2020 FORTEI-International Conference on Electrical Engineering (FORTEI-ICEE), pp. 24–28, IEEE, 2020. https://doi.org/10.1109/FORTEI-ICEE50915.2020.9249943

[3] A. Ma'arif, A. I. Cahyadi, S. Herdjunanto, and O. Wahyunggoro, "Tracking control of high order input reference using integrals state feedback and coefficient diagram method tuning," IEEE Access, vol. 8, pp. 182731–182741, 2020. https://doi.org/10.1109/ACCESS.2020.3029115

[4] A. Maarif, S. Iskandar, and I. Iswanto, "New design of line maze solving robot with speed controller and short path finder algorithm," International Review of Automatic Control (IREACO), vol. 12, no. 3, p. 154, 2019. https://doi.org/10.15866/ireaco.v12i3.16501

[5] C.-F. Juang, Y.-H. Chen, and Y.-H. Jhan, "Wall-following control of a hexapod robot using a data-driven fuzzy controller learned through differential evolution," IEEE Transactions on Industrial electronics, vol. 62, no. 1, pp. 611– 619, 2014. https://doi.org/10.1109/TIE.2014.2319213

[6] A. T. Khan, S. Li, and Z. Li, "Obstacle avoidance and model-free tracking control for home automation using bioinspired approach," Advanced Control for Applications: Engineering and Industrial Systems, p. e63, 2021. https://onlinelibrary.wiley.com/doi/abs/10.1002/adc2.63

[7] T. Dash, S. R. Sahu, T. Nayak, and G. Mishra, "Neural network approach to control wall-following robot navigation," in 2014 IEEE International Conference on Advanced Communications, Control and Computing Technologies, pp. 1072– 1076, IEEE, 2014. https://doi.org/10.1109/ICACCCT.2014.7019262

[8] A. T. Khan, S. Li, S. Kadry, and Y. Nam, "Control framework for trajectory planning of soft manipulator using optimized RRT algorithm," IEEE Access, vol. 8, pp. 171730–171743, 2020. https://doi.org/10.1109/ACCESS.2020.3024630

[9] T. Dash, T. Nayak, and R. R. Swain, "Controlling wall following robot navigation based on gravitational search and feed forward neural network," in Proceedings of the 2nd international conference on perception and machine intelligence, pp. 196–200, 2015. https://doi.org/10.1145/2708463.2709070

[10] T. Dash, "Automatic navigation of wall following mobile robot using adaptive resonance theory of type-1," Biologically Inspired Cognitive Architectures, vol. 12, pp. 1–8, 2015. https://doi.org/10.1016/j.bica.2015.04.008

[11] A. T. Khan, S. Li, and X. Zhou, "Trajectory optimization of 5-link biped robot using beetle antennae search," IEEE Transactions on Circuits and Systems II: Express Briefs, pp. 1–1, 2021. https://doi.org/10.1109/TCSII.2021.3062639

[12] A. T. Khan, X. Cao, and S. Li, "A survey on blockchain technology and its potential applications in distributed control and cooperative robots," arXiv preprint arXiv:1812.05452, 2018. https://arxiv.org/abs/1812.05452v3

[13] Z. Xu, S. Li, X. Zhou, S. Zhou, T. Cheng, and Y. Guan, "Dynamic neural networks for motion-force control of redundant manipulators: An optimization perspective," IEEE Transactions on Industrial Electronics, vol. 68, no. 2, pp. 1525–1536, 2021. https://doi.org/10.1109/TIE.2020.2970635

[14] A. T. Khan, S. Li, and X. Cao, "Control framework for cooperative robots in smart home using bio-inspired neural network," Measurement, vol. 167, p. 108253, 2021. https://doi.org/10.1016/j.measurement.2020.108253

[15] A. T. Khan and S. Li, "Human guided cooperative robotic agents in smart home using beetle antennae search," Science China Information Sciences, 2021. https://doi.org/10.1007/s11432-020-3073-5

[16] Y.-L. Chen, J. Cheng, C. Lin, X. Wu, Y. Ou, and Y. Xu, "Classification-based learning by particle swarm optimization for wall-following robot navigation," Neurocomputing, vol. 113, pp. 27–35, 2013. https://doi.org/10.1016/j.neucom.2012.12.037

[17] I. Osunmakinde, C. Yinka-Banjo, and A. Bagula, "Investigating the use of bayesian network and k-nn models to develop behaviours for autonomous robots," in Mobile Intelligent Autonomous Systems, CRC Press, Taylor & Francis Group, USA, 2012. https://doi.org/10.1201/b12690-38

[18] F. Wang and D. M. Tax, "Survey on the attention based rnn model and its applications in computer vision," arXiv preprint arXiv:1601.06823, 2016. https://arxiv.org/abs/1601.06823

[19] Z. Zhang, L.-D. Kong, and L. Zheng, "Power-type varying-parameter rnn for solving tvqp problems: Design, analysis, and applications," IEEE transactions on neural networks and learning systems, vol. 30, no. 8, pp. 2419–2433, 2018. https://doi.org/10.1109/TNNLS.2018.2885042

[20] A. T. Khan, X. Cao, S. Li, B. Hu, and V. N. Katsikis, "Quantum beetle antennae search: a novel technique for the constrained portfolio optimization problem," Science China Information Sciences, 2020. https://doi.org/10.1007/s11432-019-2735-6

[21] A. H. Khan, S. Li, D. Chen, and L. Liao, "Tracking control of redundant mobile manipulator: An rnn based metaheuristic approach," Neurocomputing, vol. 400, pp. 272–284, 2020. https://doi.org/10.1016/j.neucom.2020.02.109

[22] A. H. Khan, S. Li, and X. Luo, "Obstacle avoidance and tracking control of redundant robotic manipulator: An rnn-based metaheuristic approach," IEEE transactions on industrial informatics, vol. 16, no. 7, pp. 4670–4680, 2019. https://doi.org/10.1109/TII.2019.2941916

[23] A. H. Khan, S. Li, and X. Cao, "Tracking control of redundant manipulator under active remote center of motion constraints: An rnn-based metaheuristic approach," Science China Information Sciences, 2019. https://doi.org/10.1007/s11432-019-2735-6

[24] A. H. Khan, X. Cao, and S. Li, "Obstacle avoidance based decision making and management of articulated agents," in Management and Intelligent Decision-Making in Complex Systems: An Optimization-Driven Approach, pp. 1–29, Springer, 2021. https://doi.org/10.1007/978-981-15-9392-5_1

[25] A. H. Khan, X. Cao, S. Li, V. N. Katsikis, and L. Liao, "Bas-adam: an adam based approach to improve the performance of beetle antennae search optimizer," IEEE/CAA Journal of Automatica Sinica, vol. 7, no. 2, pp. 461–471, 2020. https://doi.org/10.1109/JAS.2020.1003048

[26] I. Hammad, K. El-Sankary, and J. Gu, "A comparative study on machine learning algorithms for the control of a wall following robot," in 2019 IEEE International Conference on Robotics and Biomimetics (ROBIO), pp. 2995–3000, IEEE, 2019. https://doi.org/10.1109/ROBIO49542.2019.8961836

[27] "Uci machine learning repository: Data set." https://archive.ics.uci.edu/ml/datasets/Wall-Following+Robot+Navigation+Data. (Accessed on 03/07/2021). https://archive.ics.uci.edu/ml/datasets/Wall-Following+Robot+Navigation+Data