Explainable Hybrid CNN and FNN Approach Applied on Robotic Wall-Following Behaviour Learning

1st Jakub Kwiatkowski
Faculty of Computing
Poznan University of Technology
Poznan, Poland
jakub.k.kwiatkowski@doctorate.put.poznan.pl

2nd Liang Ou
School of Computer Science
University of Technology Sydney
Sydney, Australia
liang.Ou-1@student.uts.edu.au

3rd Yu-Cheng Chang
School of Computer Science
University of Technology Sydney
Sydney, Australia
yu-cheng.chang@uts.edu.au

4th Chin-Teng Lin
School of Computer Science
University of Technology Sydney
Sydney, Australia
chin-teng.lin@uts.edu.au

***Abstract*—Fuzzy Neural Network (FNN) that is applied to robotic control tasks has proved to be effective by previous researchers. However, FNN has an inherent deficiency in dealing with inputs of large dimensions, such as images. Therefore, this research utilizes a Convolutional Neural Network (CNN) model to convert image into distance values and delivers these values to FNN based robot controller as inputs. The proposed hybrid CNN+FNN are tested with both a regression model and a multi-task model. Results show that the multi-task method performs better with less information loss from input images. This paper also proved that the proposed hybrid approach can be generalized into an unknown robotic simulation environment and performs better than its FNN counterpart. By utilizing state of the visual art explainable analysis method, our both the CNN part and the FNN part of the hybrid approach can be explained in a human-understandable way, thus the trustworthiness of the proposed approach is guaranteed by its high explainability.**

***Index Terms*—Explanable AI, Fuzzy System, Robotic Navigation**

I. Introduction

Deep learning reached impressive results in many areas of machine learning [1], especially in the image processing [2]. This excellent performance eventuates from the ability of DNN to extract complex non-linear features from the raw input data. Although this capability makes DNN outperforms other types of machine learning approaches, its complexity resulting in uninterpretable by humans. In contrast to deep learning, fuzzy learning system (FLS) is much more explainable, since FLS leanings fuzzy rules from input data and those fuzzy rules consist of fuzzy linguistic term sets that enable humans to understand the inherent knowledge extracted by a fuzzy system [3]. A Fuzzy Neural Network (FNN) can be considered as a combination of a set of fuzzy rules, which is proved to straightforward explanations for knowledge learned. To further foster hybridisation of fuzzy systems and DNNs, and overcome the curse of dimensionality of FNNs, we propose a simple framework, where fuzzy neural network makes decisions using interpretable rules on features extracted by DNN from raw data. The proposed model merges these two types of learning regime. We take advantage of their strong point and hide their drawbacks by employing them only in the context where they shine. This paper tested the hybrid approach on robot control application. The control problem is defined as achieving Wall-Following behaviour of a robot, by which a robot is expected to surround a wall (obstacle) without collision. The baseline model is fuzzy learning system trained on input from distance sensors, while the hybrid approach trains DNN to mimic the functionality of distance sensors. Then trained DNN is used as a substitution of sensor information during FLS training. An FNN is used to make the final decision. We believe that generalizable and continuous nature of deep neural network will improve the performance of the system. We analyse the trained models in the context of explainability to show that even with the use of deep neural network and processing raw images as inputs, we are still able to preserve explainability of the algorithm.

The article / publication was created thanks to participation in program PROM of the Polish National Agency for Academic Exchange. The program is co-financed from the European Social Fund within the Operational Program Knowledge Education Development, non-competitive project entitled "International scholarship exchange of PhD students and academic staff" executed under the Activity 3.3 specified in the application for funding of project No. POWR.03.03.00-00-PN13 / 18.

This research has been partially supported by the statutory funds of Poznan University of Technology.

This work was also supported in part by the Australian Research Council (ARC) under discovery grant DP150101645, and in part by Central for Artificial Intelligence, UTS, Australia. Research was also sponsored in part by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-10-2-0022.

II. Related Work

A. Deep learning for navigation

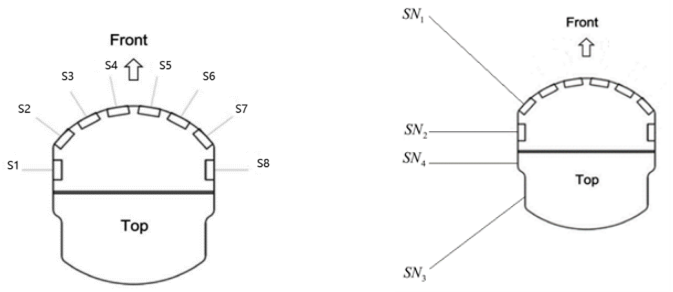
The deep neural network is increasingly popular in navigation tasks, including autonomous vehicles [4] or drones [5]. When it comes to perception in navigation setting, DNNs are often used as depth estimators, that determine the distance from the camera to object for every pixel in the image. [6] shows the possibility of learning the depth of the image from the natural images; just one image is sufficient for depth prediction. A depth estimation introduced in [7] as auxiliary task suggests that it can extract valuable information for the navigation task.

B. Fuzzy learning system

Fuzzy logic has proved to be effective in the robotic control field [8], [9], with the advantage of dealing with uncertainty with if-then-rule architecture [10]–[12]. Tasks with high uncertainties are suited for FNN. In consideration of robot models, wheeled mobile robot is one type of popular robots that attraction attention by researchers.

C. Hybrid approach

Authors of "A Survey on Fuzzy Deep Neural Networks" [13] divide hybrid approaches into 2 groups: integrated models and ensemble models. In the first group, fuzzy logic is directly integrated into a DNN [14]. In contrast, the second group represents approaches where a fuzzy system and a DNN are individual parts that jointly generate the output [15] (our approach belongs to this group). We encourage readers interested in a broader overview of hybrid models to check out the survey. The major difference between the previous studies and our paper is that these approaches are used for image processing task, mainly for classification, whereas we use multi-task deep learning model as image feature extractor for an FNN that controls a robot in the environment.



(a) Main Lidar sensor coverage illustration

(b) Four coverage reads (SN_1 to SN_4) of the auxiliary sensor for detecting distance during training

Fig. 1. 1a shows the 8 coverage (S_1 to S_8) of the distance sensor, while S_1 to S_4 are used in the robot navigation. 1b show the coverage of an auxiliary sensor used only in the training session.

III. Model

A. Fuzzy learning systems

In our model, the Fuzzy system (FS) is established based on 10 Takagi–Sugeno-type fuzzy rules. Each of the fuzzy rules is defined in the fowling formula:

$$R_i : if\ d_1(k)\ is\ \mu_{i1}\ \&\ ...\ \&\ \ d_n(k)\ is\ \mu_{in}\ Then\ u(k)\ is\ a_i, \tag{1}$$

where $d_1(k)$ to $d_n(k)$ represent inputs of the rule, $u(k)$ is the output of the rule, a_i is the weight of this rule, and μ_{i1} to μ_{in} are corresponding membership functions for inputs. These membership functions, defined in formula 2, are Gaussian functions that transform inputs into a number between 0 and 1.

$$\mu_{ij}(d_j) = exp\left\{-\left(\frac{d_j - m_{ij}}{\sigma_{ij}}\right)\right\}, \tag{2}$$

where d_j is the input of the function, m_{ij} is the centre of the fuzzy set and σ_{ij} is the width of the fuzzy set. The output of the whole system is calculated by the weighted sum of the outputs of the 10 rules, defined as follows:

$$y(k) = \frac{\Sigma_{i=1}^{r}\Phi_i(\vec{d}(k))a_i}{\Sigma_{i=1}^{r}\Phi_i(\vec{d}(k))}, \tag{3}$$

where $\Phi_i(\vec{d}(k))$ is the output of a rule, a_i is the weight of the rule, and $y(k)$ is the output of the fuzzy system. Specifically, $\Phi_i(\vec{d}(k))$ is the summation of membership values:

$$\Phi_i(\vec{d}(k)) = \Pi_j^n \mu_{ij}(d_j(k)), \tag{4}$$

where $\mu_{ij}(d_j(k))$ represents the transformed value from membership functions defined by formula 2, n is the number of the membership function.

The Multi-Objective Vibration-Based Particle Swarm Optimization (MO-VBPSO) [16] method is applied for training the FS. With MO-VBPSO, the learning process of the system aims to find an optimal set of parameters that defined the FS which can direct the robot to execute Wall-Following behaviour.

B. Deep neural network

The DNNs are trained to substitute the distance sensor. The models take images x from camera of the robot and outputs distances y to the obstacle. We experiment with two types of deep neural network: canonical regression model and multi-taks deep neural network.

1) Regression model: The regression model produces continuous values $\hat{y}$ that represent distances that robot would obtain from sensor. The head of regression model $f_h(\theta_h)$, parametrized by θ_h, produce a representation of the image r.

$$s = f_h(x; \theta_h) \tag{5}$$

The regression model estimates the distances y_r through linear mapping $f_r(\theta_r)$ of representation r.

$$\hat{y}_r = f_r(x; \theta_r) \tag{6}$$

In case of regression model estimation $\hat{y}_r$ is forwarded as the output of the DNN module $\hat{y}$.

2) Multi-task model: To tackle this problem, we model its also as multi-task learning. Our multi-task approach consists of two models: previously described regression neural network and multi-label classification deep neural network that predicts whether obstacle is out of the sensor range. Each output of mulit-label part is treated as binary classification problem with two class out-of-the-range O and in-the-range I. The multi-label clarification model exploits representation r, created by head $f_h(\theta_h)$, to produce classification result $\hat{y}_c$ through linear mapping $f_c(\theta_c)$.

$$\hat{y}_c = f_c(x; \theta_c) \tag{7}$$

The classification result is represented as vector of binary values $[l_1, l_2, ..., l_N]$, where each value tells if particular label was assigned to example x. The size of vector N equals the number of distance values collect by robot sensor.

In the training phase, we optimize cross-entropy loss for each label. Total loss for the multi-task model is presented below

$$\mathcal{L}(\theta_h, \theta_r, \theta_c|x) = ||\hat{y_r} - y_r|| - \alpha \sum_{l=1}^{N} y_{c,l} \log(\hat{y}_{c,l}) \tag{8}$$

where α describes the importance of multi-labels loss. For regression model we only train regression part (α is equal 0).

In the inference stage, the outputs of models are combined to generate final the result. Firstly the multi-label classification model is executed. In the case of out-of-the-range results, the output is forwarded as the final result. In other case, the result of regression model is used.

$$y = \begin{cases} 10, & \text{if } \hat{y}_c = O \\ \hat{y}_r, & \text{otherwise} \end{cases} \tag{9}$$

C. Deep Neural Network explainability

The explanation of deep learning model in visual task is often presented as a matrix of values that shows the importance of each pixel in output creation (saliency map). The occlusion visualisation [17] creates new images by taking the source images and applying black patches at different places to cover some part of the images (the size of the patch is parameter of the model). Then the trained model predicts the outputs based on patched images. The goal of the visualisation is to analyze the difference of the model output dependently on the location of patch. This technique is mostly used in classification setting, when the created saliency map determines the confidence on classifying the patched image as given class. For our multi-task model this works in exact same manner, but for regression model we needed to do some modification. In the regression model, the saliency map we created, shows the squared difference between the output of the patched image and source image. The gradient visualisation [18], for the purpose of saliency map creation, computes gradient of loss function wrt the input images instead of parameters of the DNN as stochastic gradient descend do.

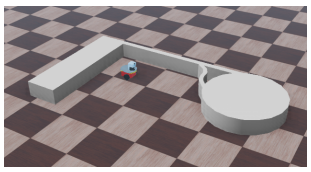


Fig. 2. Robotic training environment

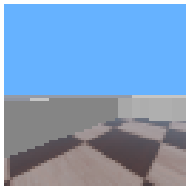


Fig. 3. Robotic training environment

IV. Experiments

A. Robotic environment

Training and testing for the hybrid approach are conducted in a robotic simulation environment (Fig. 2) (Webots). This paper adopts Pioneer 2 wheeled robot shown in the centre of the Figure. The robot is installed with a lidar sensor for distance detection and a camera for image collection. The white area represents the Wall or Obstacle. The Wall-Following Behaviour task requires the robot to move around the wall without collision.

Performance of the robot is evaluated by 3 fitness functions:

$$f_1 = T_c, \tag{10}$$

$$f_2 = \sum_{t=1}^{T_c} \frac{T_c}{\left| \frac{1}{2} \left(\frac{SN_1(t)}{SN_3(t)} + \frac{SN_2(t)}{SN_4(t)} \right) - 1 \right|}, \tag{11}$$

$$f_3 = \sum_{t=1}^{T_c} \frac{T_c}{|SN_2(t) - 0.2|}, \tag{12}$$

Note that SN_1 to SN_4 are distance from an auxiliary sensor used only in the training, shown in Fig. 1. The first fitness function evaluates the robot's running time until failure. Fail of the robot is defined as robot moving too far (more than 5m) or too close (less than 0.2m) to the Wall. The second fitness function evaluates the smoothness of the robot's movement. By implementing this fitness function, the robot is encouraged to not change its direction frequently. The last fitness function evaluates the closeness of the robot toward the wall. This fitness provide restriction to prevent the robot from moving far from the wall to gain running time, formula (10).

B. Deep learning dataset

For training deep neural network we created an images dataset that was extracted from a robotic environment. Along with the movement of the robot, the camera with record images for its surrounding environment, i.e. the view of the wall, while the lidar sensor will record the distance between the robot and the wall. Fig. 3 is an image example taken from the camera, while Fig. 1a demonstrate the coverage of the lidar sensor. S_1 to S_8 are 8 reads of the fan-shaped lidar sensor coverage, although in this experiment only S_1 to S_4 are in use because the experiment only concerns the left-hand side detection of the wall. For the purpose of the multi-target model we also created the second set of binary labels for each sensor that determine if the obstacle is in the range of a particular coverage of the sensor.

TABLE I
ARCHITECTURE OF REGRESSION AND MULTI-TARGET MODEL.

Layer Type	Hyper-parameters				
	Filters/Neurons	*Kernel size*	*Strides size*	*Padding*	*Activation*
Conv2D	64	4x4	2x2	Same	Relu
Conv2D	64	4x4	2x2	Same	Relu
Conv2D	64	4x4	2x2	Same	Relu
Conv2D	64	4x4	2x2	Same	Relu
Dense	256				Relu
Dense	4				Linear

C. Deep learning experiments

We trained the regression model and the multi-target model using same hyper-parameters. We have used Adam optimizer with 0.0001 learning rate and batch size that equals 64. These models have also exact same architecture (Table I), except that the multi-target model has two output layers: one for regression and another one for classification. We were able to reach 0.2482 loss for regression model and loss for multi-target model 0.3017. Accuracy for multi-target model equals 0.9899.

D. Deep Neural Network explainability

We perform occlusion visualisation on both regression and multi-target models. The width and height of used path equal 4 pixels. For multi-target models we also conduct a gradient visualisation. Since we use the SmoothGrad [18] version of gradient visualisation, we create the batch of the image that are copies of the source image with gaussian noise applied to them. We add the Gaussian noise with mean of 0 and variance of 0.125 to images where each pixel is represented by 3-dimensional tensor with 3 channels that values range from 0 to 1.

The results of these methods are presented in similar manner as a grid of images. The first column show the samples of original images from dataset and the rest of columns presents the original images with saliency maps applied. In case of the occlusion visualisation of regression model the second columns show the averaged saliency map for all output. In other cases the averaged saliency map for all outputs does not exist. The rest of columns are related to each of outputs of neural network in order from left to right. As regression output in multi-target model have similar results to regression model, we present only visualisation multi-label part.

The upper part of the image, that shows the sky of the environment, does not change in the whole dataset. Considering this fact, we exclude this part from all of the visualisations since it would show some random saliency that would unnecessarily clutter visualisation and made it harder to understand.

V. RESULT DISCUSSION AND ANALYSIS

A. Fuzzy training results

1) Trajectory comparison with FNN baseline: Robotic trajectory tests of the trained models are conducted in the simulation environment. Fig. 4 compares the moving trajectories of the Multi-Task model with lidar sensor-based FNN model.

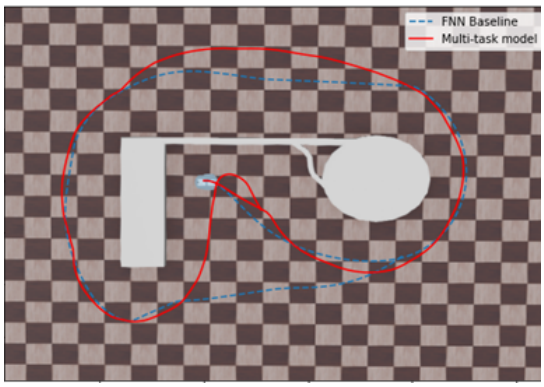


Fig. 4. Trajectory illustration of image driven hybrid approach with Multi-Task Model and the sensor driven FNN.

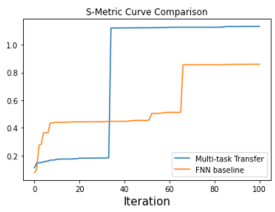


Fig. 5. Trajectory illustration of the image-driven hybrid approach with the multi-task Model and the sensor-driven FNN.

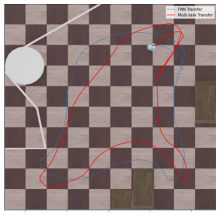


Fig. 6. Trajectory illustration of image driven hybrid approach with Multi-Task Model and the sensor driven FNN.

Although the Multi-Task model following the way with a similar approach, it sticks with the wall much better than other the baseline at the end of the cycle.

B. Generalization evaluation

Since the CNN part of the Hybrid model is trained to predict distance between the wall and the robot, we tested the generalization ability of the the pre-trained Multi-Task model in a new environment, and compare it with lidar sensor driving FNN baseline. The comparison is based on s-metric value introduced by [19]. Fig. 5 shows that, by taking inputs from the multi-task model, the FNN can learn even better performance at the end of the experiment, although it starts at comparatively lower level. Generalization ability is also evaluated by their trajectories. Fig. 6 compares the trajectories of pretrained FNN baseline and pretrained hybrid approach in a new environment. Although both of them have generalization ability, the baseline follows the wall better in alone straight path, whereas the hybrid approach performs better at corners.

We believe that the generalization ability of our method comes also from the hybrid nature of our model. The individual DNN, used for control of the robot, could just remember the steps needed to achieve a goal (over-fitting). However, that kind of strategy would not be able to achieve a goal in a different environment.

C. Explainability evaluation

1) Deep Neural Network explainability: The saliency maps indicates the parts of the image that neural network focuses.

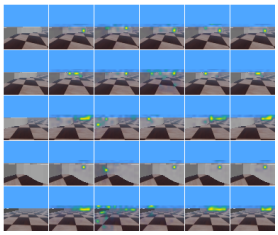


Fig. 7. Occlusion visualisation of regression model that shows the difference in prediction of regression model in case when marked part is covered with 4x4 black patch. Consecutively columns show: original image and then 4 saliency maps for each output of regression model.

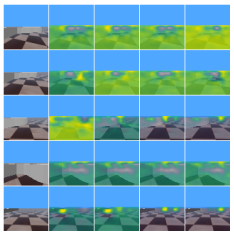


Fig. 8. Occlusion visualisation of multi-task model that shows the confidence that obstacle is in the range of sensor when that part is covered with 4x4 black patch. Consecutively columns show: original image and then 4 saliency maps for each output of regression model.

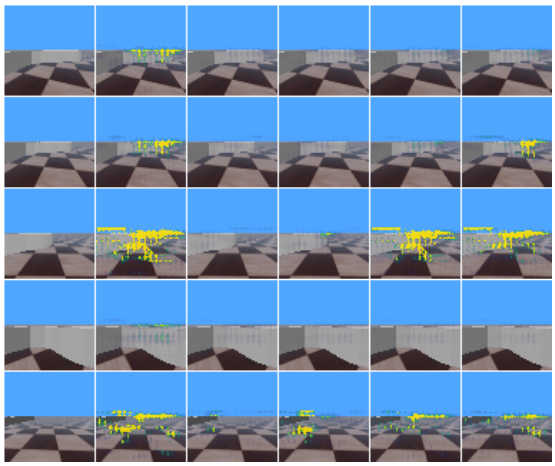


Fig. 9. Saliency map of SmoothGrad algorithm with number of images in batch that equals 10. The algorithm shows the averaged gradient of multi-label output wrt to image with applied Gaussian noise. Consecutively columns show: original image and then 4 saliency maps for each output of regression model.

These parts contain characteristic points that are good indicators of locations in the environment. We can clearly see that the most important parts of the image, according to the trained model, are edges of obstacle. Giving the most attention to these point is reasonable way of achieving goal in this environment, since it gives the most valuable information about the locations of robot in the environment. The point that has especially good amount of saliency is the right end of obstacle since its the most important point that robot should follow to achieve the goal. The far horizon of these environment that have a lot of saliency seems also like a important point for navigation. This is also part of the image that varies in high level sense. The robot learn to move around the obstacle that in most cases is on the left side of the robot, so the right part part varies in the presence of obstacle or floor, compared to left part which is mostly populated by presence of obstacle. Since we use the squared difference in saliency map creation, the visualisation focuses on the parts, that when covered, change the output in the most meaningful way. Although weaker we can also see the saliency on the floor in the environment, especially in contour of the pattern of squares on the floor, so the robot also uses a floor pattern to navigate thought environment.

The saliency map in Fig. 8 shows the probability of obstacle being in the range of sensor when that part of the image is covered. In the first row we have situation where obstacle is in the range of every read of sensor. The saliency is high in most places, because if we cover these places, model still predicts that obstacle is in the range of sensor. The only exceptions are locations on obstacle, especially the right end of obstacle, when covered the model would predict that sensor is out of range. In the second row the area of low probability of obstacle being in the sensor range increase for the right-most image. In first and second row we can also see few interesting cases of covering obstacle: When the patch covers the left part of obstacle the probability is high, because the model see the part of obstacle on the right side. Then cover starts to be close to the right edge of obstacle or slightly cover it, so the model don't know the exact location of right edge of obstacle. When most of cover is on the left side of edge of obstacle, there is possibility that the end is more to the left than is in reality, so the possibility of being out of the range increases. However, when the major part of the cover is right to the edge, there is possibility that the edge is more to right, so the probability of being in the range of sensor dramatically roses. In the third row the situation drastically changes as edge of obstacle is in the middle of image. The right reads of senor have obstacle out of range, so the only way of tricking these sensor to predict in-the-range output is to cover some part of horizon or the end of obstacle. In the fourth row situation is similar for all of reads of sensor, as all of them are in the range of obstacle and there is no easy way of changing it. The covering of the middle of obstacle is way of tricking a model to decrees the probability of obstacle being in the range of sensor. We hypothesise that when you cover the middle of obstacle is hard for the model to distinguish between covered obstacle and pattern on the floor. In the last row all of reads of sensor are out of the range, especially the right reads, so saliency is low in almost every part of the image. Only when you cover the some parts of the horizon the probability of being in the range of obstacle increases.

Gradient visualisation of multi-task model shows gradient of maximizing the probability of obstacle being out of the sensor range with respect to image. We see that most of saliency emerges on the obstacle and far horizon of the image. There is not much of the gradient in the first row of visualisation. However as the right edge of obstacle moves to left, the obstacle is getting out of range for right reads of sensor and the saliency starts to emerge on these images, what we can see in the next two rows. In the 4 row there is little saliency, because the obstacle is in range of sensor. In the last row we see a little bit of saliency on each of the image as every one of them have obstacle out of range. In last row is also hard to localize obstacle, so model focus some attention on the pattern on the floor to navigate in the environment.

TABLE II
FNN RULES LEARNT WITH THE MULTI-TASK MODEL

Rule Number	Fuzzy Inference				
	Input1	***Input2***	***Input3***	***Input4***	***Output***
1	VS(-1.0,2.9)	H(7.4,-0.3)	H(5.9,2.0)	VS(-0.7,0.0)	0.0
2	M(2.0,-0.7)	S(0.0,0.0)	M(3.9,-0.8)	S(0.3,0.6)	-0.3
3	VS(-0.8,0.3)	S(0.6,-1.9)	VS(-0.4,7.5)	VS(-0.8,-0.3)	-0.1
4	H(7.5,-0.8)	VS(-0.3,1.7)	S(1.5,5.5)	S(1.0,4.2)	-0.9
5	H(5.5,1.0)	H(4.2,1.0)	M(2.6,1.4)	VS(-0.1,0.9)	1.9
6	S(1.4,-0.1)	S(0.9,-0.8)	M(2.0,2.9)	H(5.0,2.7)	-0.9
7	M(2.9,5.0)	M(2.7,-2.4)	S(1.7,-0.9)	M(3.3,0.6)	0.0
8	VS(-0.9,3.3)	S(0.6,3.8)	M(2.9,7.5)	M(2.0,1.2)	-4.6
9	H(7.5,2.0)	S(1.2,2.0)	S(1.9,2.3)	S(0.1,0.1)	-0.8
10	M(2.3,0.1)	S(0.1,1.2)	H(5.0,1.0)	S(1.2,1.3)	3.1

2) FNN explainability: Taking advantage of the inherent interpretability of fuzzy system, the parameters in the pre-trained FNN can be analysed with rule-based logic. Table II displays the logic of 10 rules in the FNN learnt together with the Multi-Task CNN model. Each input corresponding to the predicted distance values from the CNN, which are the mimic of lidar sensor range S1 to S4, and are transformed with Gaussian membership function (formula 2). The centre and width of the membership functions are all displays in the brackets of the table. Semantically, we define negative distances as Very Short ("VS"), distances from 0 to 2 as Short ("S"), distances from 2 to 4 as Medium ("M"), distances larger than 4 as Hight ("H"). The last column of the table shows the consequent part of each rule, in this case, the steering angle to direct the robot. Positive angles direct the robot moving toward the left, whereas negative angles direct the robot moving towards the right. Therefore, we can explain the FNN by looking at each of the learnt parameters of rules. For, example, if S1 is very short, S2 and S3 are short, S4 is high, rule 7 will be fired by directing the robot moving straight forward. Semantically, this rule means if the robot finds obstacles in its side but not in its front, it will go straight. This logic of achieving wall-following align with human intuition.

VI. CONCLUSION

The proposed model combines the robustness and generalisability of deep neural network and explainability of fuzzy learning system. By replacing lidar sensor inputs into image from camera, the hybrid approach we proposed is more robust than traditional lidar sensor-based learning methods. This robustness of the hybrid approach is identified in the testing simulation environment. Since our DNN is trained to mimic distance from sensors, and applied comprehensive visual analysis, our approach secures the interpretability of the DNN. In our future work we investigate the possibility of training two models jointy. Another interesting case is possibility of combining fuzzy learning system with unsupervised deep learning models, like Variational AutoEncoders or GenerativeAdversarial Networks. The question rises if there is possible cooperation between models and whether deep neural network can extract features that are sufficiently meaningful to control robot through fuzzy learning system.

REFERENCES

[1] D. Bahdanau, K. Cho, and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate." [Online]. Available: http://arxiv.org/abs/1409.0473

[2] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition." [Online]. Available: http://arxiv.org/abs/1409.1556

[3] L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, and L. Kagal, "Explaining explanations: An overview of interpretability of machine learning," in *2018 IEEE 5th International Conference on data science and advanced analytics (DSAA)*. IEEE, 2018, pp. 80–89.

[4] X. Liang, Y. Liu, T. Chen, M. Liu, and Q. Yang, "Federated Transfer Reinforcement Learning for Autonomous Driving." [Online]. Available: http://arxiv.org/abs/1910.06001

[5] H. X. Pham, H. M. La, D. Feil-Seifer, and L. V. Nguyen, "Autonomous UAV Navigation Using Reinforcement Learning." [Online]. Available: http://arxiv.org/abs/1801.05086

[6] D. Eigen, C. Puhrsch, and R. Fergus, "Depth Map Prediction from a Single Image using a Multi-Scale Deep Network." [Online]. Available: http://arxiv.org/abs/1406.2283

[7] P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. J. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu, D. Kumaran, and R. Hadsell, "Learning to Navigate in Complex Environments." [Online]. Available: http://arxiv.org/abs/1611.03673

[8] X. Xiang, C. Yu, L. Lapierre, J. Zhang, and Q. Zhang, "Survey on fuzzy-logic-based guidance and control of marine surface vehicles and underwater vehicles," *International Journal of Fuzzy Systems*, vol. 20, no. 2, pp. 572–586, 2018.

[9] R. H. Abiyev, N. Akkaya, and I. Gunsel, "Control of omnidirectional robot using z-number-based fuzzy system," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 1, pp. 238–252, 2018.

[10] I. Couso, C. Borgelt, E. Hullermeier, and R. Kruse, "Fuzzy sets in data analysis: From statistical foundations to machine learning," *IEEE Computational Intelligence Magazine*, vol. 14, no. 1, pp. 31–44, 2019.

[11] I. Caylak, E. Penner, and R. Mahnken, "A fuzzy uncertainty model for analytical and numerical homogenization of transversely fiber reinforced plastics," *PAMM*, vol. 19, no. 1, p. e201900356, 2019.

[12] D. Chen, X. Zhang, L. L. Wang, and Z. Han, "Prediction of cloud resources demand based on hierarchical pythagorean fuzzy deep neural network," *IEEE Transactions on Services Computing*, 2019.

[13] R. Das, S. Sen, and U. Maulik, "A Survey on Fuzzy Deep Neural Networks," vol. 53, no. 3, pp. 54:1–54:25. [Online]. Available: https://doi.org/10.1145/3369798

[14] S. R. Price, S. R. Price, and D. T. Anderson, "Introducing Fuzzy Layers for Deep Learning," pp. 1–6. [Online]. Available: http://arxiv.org/abs/2003.00880

[15] S. Zhou, Q. Chen, X. Wang, and X. Li, "Hybrid Deep Belief Networks for Semi-supervised Sentiment Classification," in *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*. Dublin City University and Association for Computational Linguistics, pp. 1341–1349. [Online]. Available: https://www.aclweb.org/anthology/C14-1127

[16] L. Ou, G. Zeng, Y.-C. Chang, and C.-T. Lin, "Multi-objective vibration-based particle-swarm-optimized fuzzy controller with application to boundary-following of mobile-robot simulation environment," in *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2020, pp. 1893–1898.

[17] M. D. Zeiler and R. Fergus, "Visualizing and Understanding Convolutional Networks." [Online]. Available: http://arxiv.org/abs/1311.2901

[18] D. Smilkov, N. Thorat, B. Kim, F. Viégas, and M. Wattenberg, "SmoothGrad: Removing noise by adding noise." [Online]. Available: http://arxiv.org/abs/1706.03825

[19] E. Zitzler, "Evolutionary algorithms for multiobjective optimization: Methods and applications."