AI-FML Agent with Patch Learning Mechanism for Robotic Game of Go Application

Yi-Lin Tsai
Dept. of Computer Science and Information Engineering
National University of Tainan
Tainan, Tawain
m10759011@gm2.nutn.edu.tw

Chang-Shing Lee
Dept. of Computer Science and Information Engineering
National University of Tainan
Tainan, Tawain
leecs@mail.nutn.edu.tw

***Abstract*—In this paper, we propose an AI-Fuzzy Markup Language agent with patch learning mechanism for robotic game of Go application. The proposed agent contains three kinds of intelligence, including a perceptual intelligence, a cognitive intelligence, and a computational intelligence, for the robotic application. Additionally, we embed the patch learning mechanism into our agent. The method for running patch learning involves three steps. First, trains an initial global model, then trains a patch model for each identified patch, and finally updates the global model using the training data that do not fall into any patch. This paper adopts the Google DeepMind Master 60 games to be the training data and testing data. The experimental results show that the proposed agent with the patch learning mechanism can improve the performance of regression for robotic game of Go application.**

Keywords—AI-FML, Agent, Patch Learning, Fuzzy Machine Learning, Game of Go

I. INTRODUCTION

Nowadays, machine learning has been used in multiple real-world fields and industries, such as medical diagnosis, image processing, regression etc. However, training a high performance machine learning model is usually an iterative process which relies on experience and trial-and-error. Sometimes, we need to take some remedies to enhance its performance when it is dissatisfied. Some strategies are implemented to enhance the performance on machine learning mechanisms, such as using a single deeper model, using a single broader model, connecting multiple simple base learners in parallel, or connecting multiple simple weak learners in series. In this paper, we propose an AI-Fuzzy Markup Language (AI-FML) agent with the patch learning (PL) mechanism to improve the performance of the machine learning. According to [1], PL consists of three steps described as follows: (1) Train an initial global model using all training data first. (2) Identify patches from the initial global model and train a patch model for each patch. (3) Update the global model using training data. Wu and Jerry defined a patch as a connected polyhedron in the input domain. For example, a patch in a 1D input domain is an interval, and a patch in a 2D input domain can be a rectangle, an ellipse, and so on. However, generally identifying the patch locations is a very challenging task, and not every problem can be easily visualized. The PL connects multiple simple models both in parallel and in series to improve the learning performance. Mendel [2] introduced how PL can be performed using rule-based fuzzy systems, because it is easy to initialize patch candidates in a fuzzy system.

Regression, ensemble methods, and deep learning are important machine learning methods for data scientists [9]. An adaptive network-based fuzzy inference system (ANFIS) was proposed by Jang [11] in 1993. The ANFIS integrates both machine learning model and fuzzy logic principles and it has a potential to capture the benefits of both in real-world applications. Its inference system corresponds to a set of fuzzy rules that have learning capability to approximate nonlinear functions and is considered to be a universal estimator. In our previous paper [3], we used AlphaGo Master [14] 60 games as the experimental dataset to make the win rates predicted by the Darkforest AI bot [12] closer to those predicted by the ELF Open Go AI bot [13] based on FML-based genetic learning (GFML), XGBoost learning, and a seven-layered deep fuzzy neural network (DFNN) learning. In this paper, we further combine FML, PL, and adaptive network-based fuzzy inference system (ANFIS) with the deep learning to train a model to predict the win rates of the AlphaGo Master 60 games. The experimental results show that introducing PL mechanism has a better performance than the methods presented in our previous paper [3].

The remainder of this paper is as follows: Section II presents the structure of ANFIS-based patching learning mechanism. Section III describes the patch learning mechanism for game of Go data set. Section IV proposes the PL-based AI-FML agent for robotic application on predicting the win rate of Go game. Section V shows the experimental results. Finally, section VI draws the conclusions.

II. STRUCTURE OF ANFIS-BASED PATCH LEARNING MECHANISM

A. Patch Learning Mechanism

During the design stage of the traditional ANFIS, the training data are used to optimize the input membership functions and consequent parameters over input domain, and the performance metrics are optimized using all training data [1, 11]. Based on the concept of PL in [1], this paper proposes an AI-FML agent with patch learning mechanism and it begins with a globally designed rule-based fuzzy system, but then locates the patches which have made the most contribution to the performance metrics. Fig.1 shows the structure of patch learning for predicting the win rate of Master 60 Go games which described as follows: (1) We use all of the training data to train the global model. (2) We identify that the input regions from move 1 (M1) to move P1 (M_{P1}) give rise to large learning

errors so that we use the training data which fall into this region to train the patch model 1 (PM_1) to reduce the overall learning error. (3) Finally, the global rule-based fuzzy system is updated, using the remaining training data that have not been used by patches 1, 2, .., and L. Fig. 1 takes Game 2 as an example to show that L = 3, M_{P1} = 20, M_{P2} = 40, and M_{P3} = 90.

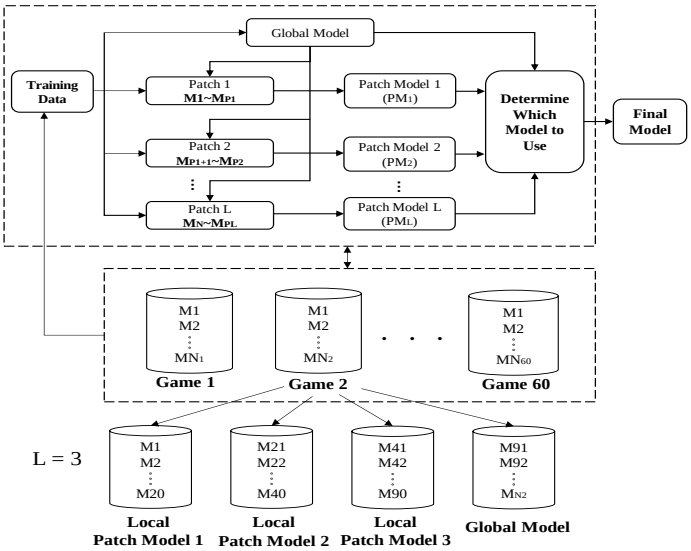


Fig. 1. Structure of patch learning for predicting the gane of Go Data set.

B. *Adaptive Network-based Fuzzy Inference System (ANFIS) Mechanism*

The ANFIS-based patch learning mechanism is embedded into the AI-FML agent and we apply it to predict the win rate of game of Go data set. Additionally, ANFIS algorithm is adopted to train the patch models to improve the performance metrics [1, 11]. The ANFIS is one of artificial neural network models that is based on Takagi–Sugeno fuzzy inference system, and developed in the early 1990s [10, 11]. In this paper, we utilize PyTorch to implement the framework of the ANFIS-based patch learning model. Fig.2 shows the structure of ANFIS contains two parts, including premise part and consequence part and its architecture is composed of five layers [12] described as follows: (1) First Layer/Fuzzification Layer: The membership degrees of each function are computed by using the premise parameter set. (2) Second Layer/Rule Layer: It is responsible of generating the firing strengths for the rules by multiplying the incoming signals. (3) Third Layer/Normalizes Layer: It normalizes the computed firing strengths by diving each value for the total firing strength. (4) Fourth Layer/Inference Layer: It multiples the normalized values from the third layer with the consequent parameters and sends the results to the fifth layer. (5) Fifth Layer/Defuzzified Layer: It computes the overall output as summation of all incoming signals and to generate the final output.

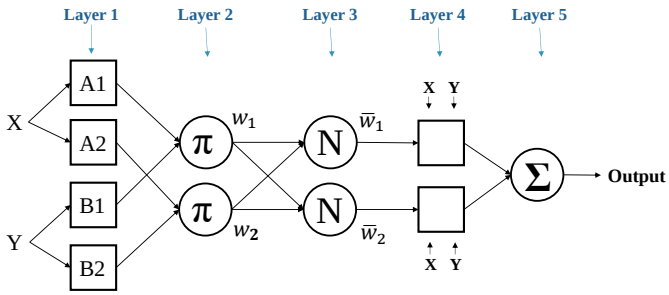


Fig. 2. Structure of ANFIS with five layers[11].

III. ANFIS-BASED PATCH LEARNING FOR GO GAME DATA SET

A. *Introduction to Data Set from IEEE WCCI 2020*

In this paper, we adopt 60 online games Master in Dec. 2016 and in Jan. to be the experimental data and they were designated to be the competition data held in IEEE CIS flagship conferences [3]. The participates can choose any 40 Games from 60 Games as the training data and the remaining 20 Games as the testing data. The participates construct the knowledge base and the rule base of the fuzzy inference system. Table 1 shows the information of move 1 to move 20 of Game 1. Each game includes the information predicted by Darkforest AI Bot and by EFL OpenGo AI Bot. MoveNo is the move number but MoveNo only lists "odd" numbers (i.e., 1, 3, 5, ...) because each row corresponds to a pair of one Black move and one White move. That is, the row with the MoveNo 1 corresponds to the Black first move (i.e., B1) and the White first move (i.e., W2). The row with the MoveNo 145 corresponds to the Black 145th move (i.e., B145) and the White 146th move (i.e., W146). If the final MoveNo is "odd," "White's information of the last row" will be vacant. There are 3,758 data adopted from Master Game 1 to Game 40 for the training dataset. The Game 41 to Game 60 with 1880 dataset will be adopted to be the testing data in this paper.

TABLE I. MOVE 1 TO MOVE 20 OF GAME 1.

Move No	DBSN	DWSN	DBWR	DWWR	DBTMR	DWTMR	EBWR	EWWR
1	3863	2274	0.52	0.48	0	1	0.49	0.50
3	9283	7866	0.51	0.48	0.5	1	0.45	0.54
5	11395	6798	0.51	0.47	0.66	1	0.45	0.57
7	4499	10703	0.51	0.46	0.75	1	0.49	0.52
9	7388	20017	0.52	0.46	0.8	1	0.47	0.53
11	20098	9693	0.53	0.46	0.83	1	0.47	0.52
13	20017	14595	0.53	0.47	0.85	1	0.48	0.49
15	6786	4892	0.52	0.45	0.87	0.875	0.48	0.53
17	20017	6432	0.53	0.46	0.88	0.7778	0.48	0.53
19	5267	13267	0.51	0.48	0.9	0.8	0.48	0.50

Note

(1) *DBSN* and *DWSN* denote the number of simulations for Black and White, respectively.
(2) *DBWR* and *DWWR* are the win rate of Black and White, respectively.
(3) *DBTMR* and *DWTMR* are the top-move rate of Black and White, respectively.
(4) EBWR and EWWR are the win rate of Black and White predicted by ELF OpenGo, respectively.

B. *Pre-processing Partition Function for Game Data Set*

There are different kinds of partition functions for partitioning the input domains, for example, a crisp partition function or a Type-1 partition function [2]. A crisp partition of real variable is comprised of non-overlapping adjacent regions that are intervals of real numbers, where the membership degree belonging in each region is unity, and is zero outside of that region. In addition, a first-order uncertainty partition of the real variable is comprised of overlapping intervals, where one is absolutely certain about where the overlap begins and ends, so that the degree of belonging in each of overlap is a real number that is an element of [0, 1]. A first-order uncertainty partition is characterized by non-rectangular type-1 partition function that overlaps with their nearest neighbors. Fig. 3 shows the adopted type-1 partition with 64 Gaussian function for the game of Go data pre-processing in this paper.

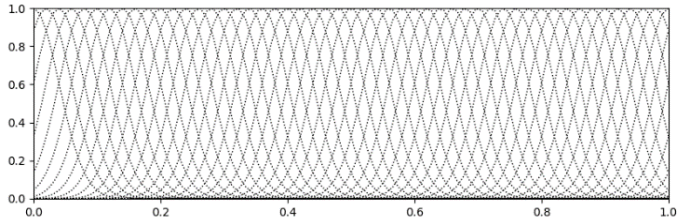


Fig. 3. Type-1 partition functions with 64 Gaussian functions for Go.

In addition, there are two criteria utilized for data pre-processing in this paper. The first criteria is "if the final prediction result of Darkforest is mistake, then the dataset of this game will not adopted for training data or testing data." For example, the dataset of Game 15, Game 25 and Game 29 will not adopted for the training data.

The second criteria is "*if the predicted moves of White or Black are different between ELF OpenGo and Darkforest that are more than 40%, then the dataset for the game will not adopted in the training data or testing data*." For example, if one game with 100 moves, but there are 40 moves predicted by ELF OpenGo and Darkforest are different, then this game will be removed from the dataset. Based on this criteria, the Game 8 and Game 36 are removed from training dataset. In addition, Games 44, 47, 56 and 57 are removed from the testing dataset in this paper. Moreover, we check the missing data from the dataset. If one game with the final move and the first move are the same as Black or White, then we move the final move from this game. After checking, we remove 19 data from training dataset and 6 data from testing dataset. Finally, we adopt 3143 data for the training dataset and 1239 data for the testing dataset in this paper.

IV. PL-BASED AI-FML AGENT FOR ROBOTIC GAME OF GO APPLICATION

A. *ANFIS for AI-FML Agent on Game*

The Adaptive Network-Based Fuzzy Inference System (ANFIS) architecture is a long-established and popular approach to implementing fuzzy systems. In this paper, we implement ANFIS using PyTorch framework. PyTorch is an open-source deep learning platform for Python, featuring: (1) tensor computing. (2) automatic differentiation in recorded tensor operations. (3) libraries for neural nets, optimisers, and loss functions. For the experiment, we using Python 3.7.3, the Anaconda 4.7.12 distribution and PyTorch 1.5.0+cpu. The training was carried out on a computer with an Intel Xeon E5-2667 v4 processor running at 3.20 GHz using 16 GB of DDR4 RAM, running Ubuntu version 16.04.5 LTS (Xenial Xerus). The core fuzzy rule-based system of PL-based AI-FML agent is ANFIS composed of the following functional blocks: (1) a knowledge base with database, (2) a rule base, (3) the inference operators on the rules, (4) a fuzzification process, and (5) a defefuzzification process [12]. Fig. 4 shows the structure of ANFIS for AI-FML agent on robotic application which is constructed by following the steps as follows: (1) retrieve the input values for fuzzy variables with membership functions on the premise part to compute the membership degree of each linguistic label, (2) combine the membership degrees on the premise part to get the weight of each rule, (3) generate the consequents of each depending on the weights, and (4) aggregate the consequents to produce the final output [12]. Human input the data and AI-FML agent output the defuzzied value to the robots, such as Kebbi Air, Palro or Zenbo, to achieve the goal of human and robot colearing in the classroom [3].

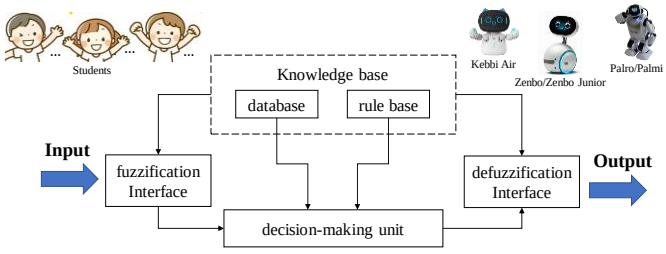


Fig. 4. Structure of ANFIS for AI-FML agent.

In this paper, we use 64 type-1 Gaussian partition functions to partition the six input fuzzy variables, including DBSN, DWSN, DBWR, DWWR, DBTMR, and DWTMR. For example, if x is the input value of DBSN for the PL-based AI-FML agent, then there are 64 rules as follows:

Rule 1: If x is Partition 1, Then y=y1(x)
Rule 2: If x is Partition 2, Then y=y2(x)
⋮
Rule 64: If x is Partition 64, Then y=y64(x).

where $y_1(x)$, $y_2(x)$, …, and $y_{64}(x)$ are different functions of x. In Partition P(1|x), if only Rule 1 is fired, and hence the fuzzy system output is y=$y_1(x)$. In addition, in Partition P(2|x), both Rule 1 and Rule 2 are fired, and hence the fuzzy system output is the weighted average of y1(x) and y2(x) with the membership degree of Partition 1 and Partition 2, respectively.

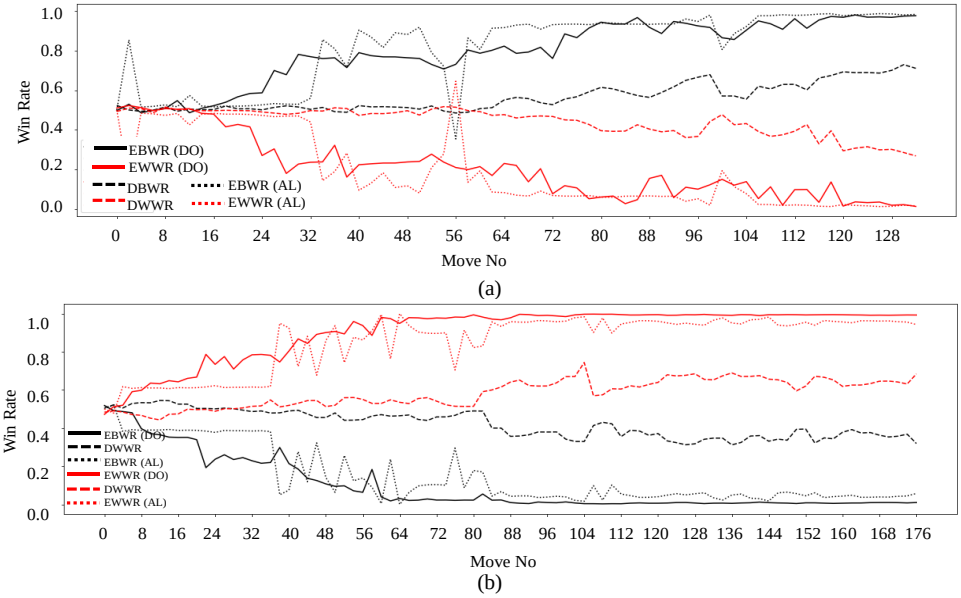


Fig. 5. Regression analysis for EXP. 1 in Games (a) 39 and (b) 58

B. *PL-based AI-FML Agent for Game of Go Dataset*

The PL-based AI-FML agent can use 1-Patch, 2-Patch and 3-Patch for game of Go Dataset. In Fig. 6, we first construct a global PL-based AI-FML agent by training all the dataset. Then construct the 1-Patch Learning model by predicting the move 1 to move 40, and the other moves are trained by the global model, shown as Fig. 6(a). Fig. 6(b) shows the 2-Patches learning model by the first Patch based on move 1 to move 20, the second Patch based on move 21 to move 40, and Global model based on the other moves. Fig. 6(c) shows the 3-Patches learning based on (1) Patch 1: move 1 to move 20, (2) Patch 2: move 21 to move 40, Patch 3: move 41 to move 90, and (4) Global: move 91 to end move.

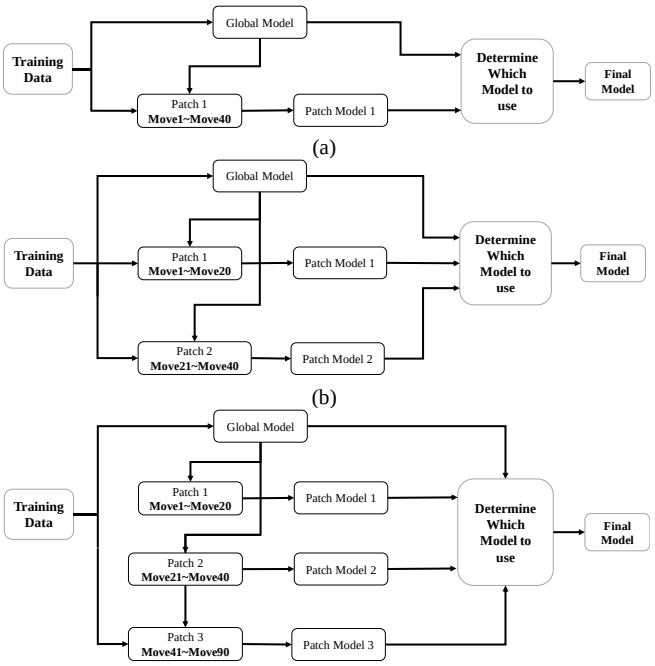


Fig. 6. Structures of Patch Learning. (a) 1-Patch; (b) 2-Patch; (c) 3-patch.

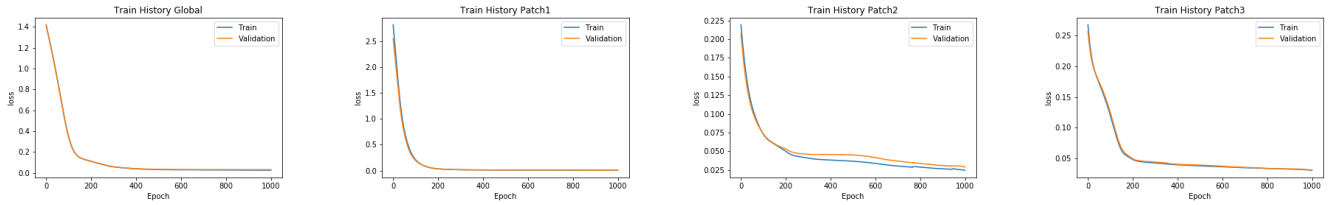
V. EXPERIMENTAL RESULTS

A. *Experiment 1: ANFIS Global Learning for AI-FML Agent*

In Experiment 1, we adopt the ANFIS with Adam optimizer and gradient decent learning algorithm with learning rate 0.001 to update all the parameters in the neural network. Table II shows the loss values using Mean Absolutae Error (MAE), Mean Square Error (MSE), and Root Mean Square Error (RMSE) criteria after 2,000 epochs. This method has a better performance than the one of using DNN with Adam machine learning implemented in our previous paper [3]. Additionally, Exp. 1 reduces both total number of learning parameters and learning time. Fig. 5(a) shows that the dotted curves have a tendency to approach to the solid ones predicted by ELF OpenGo, but the first five moves and the moves near move 56 still have a bigger variance. Fig. 5(b) indicates that the predicted behavior of the dotted curves is similar to the one of the solid curves in addition to the first 40 moves. Because of this situation, we implemented the concept of patch learning in Exp. 2.

TABLE II. LOSS EVALUATION BASED ON MAE, MSE, AND RMSE

	Training set	Validate set	Testing set
MAE	0.0683	0.0876	0.1197
MSE	0.0158	0.0286	0.0438
RMSE	0.1258	0.1692	0.2093



(a) (b) (c) (d)

Fig. 7. Learning curves for (a) Global Model, (b) Patch 1, (c) Patch 2, and (d) Patch 3.

B. *Experiment 2: PL-based AI-FML Agent with 3-Patches*

In Experiment 2, we adopt the PL-based AI-FML agent with three patches mechanim. We partition our three patches as follows: Patch 1: Move 1 to Move 20, Patch 2: Move 21 to Move 40, and Patch 3: Move 41 to Move 90. First, we learn the Global Model for 1000 epochs and then learn Patch 1, Patch 2, and Patch 3 for 1000 epochs. Figs. 7 (a), (b), (c), and (d) show the history learning curves of the training data set and validation data set during training Global Model, Patch 1, Patch 2, and Patch 3, respectively. Figs. 7(a), (b) and (d) show a better fit than Fig. 7(c) does. Mabye, it is because after move 20, Go is at the the middle stage which is under an unpredictable uncertainty situation which makes it more difficult to fit the actual situation during learning the model. In Exp. 2, we design two sub-experiments and describe as follows: (1) Exp. 2.1: We only use the trained Global Model with 1000 epochs to preidct the training data set. (2) Exp. 2.2: We use the trained Global Model, Patch 1, Patch 2, or Patch 3 to predict the training data set according to which patch the moves of the games fall into. Fig. 8 shows the curves of aveage loss for each move. The orange solid and purple dotted curves are the results of the Exp. 2.1 and Exp. 2.2, respectively. The loss values of Exp. 2.2 focused on the patches 1, 2, and 3 have a decreasing tendency compared to the ones of Exp. 2.1.

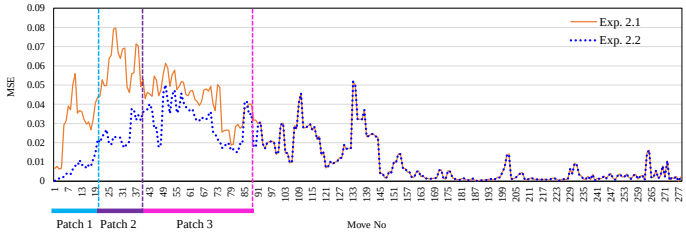


Fig. 8. MSE curves for each move in Exp. 2.

VI. Conclusions

The AI-FML agent with patch learning mechanism for robotic game of Go application is presented in this paper. Various PL models including, PL with global model, 1-patch, 2-paches, or 3-patches model for each identified patch, are performed in the experiments. In addition, PL can be implemented using AI-FML agent for robotic application. We adopt the Google DeepMind Master 60 games to be the training data and testing data set. The experimental results show the AI-FML agent with patch learning can improve the performance of regression for robotic game of Go application.

References

[1] D. Wu and J. M. Mendel, "Patch learning," *IEEE Transactions on Fuzzy Systems*, Jul. 2019. (DOI: 10.1109/TFUZZ.2019.2930022)

[2] J. M. Mendel, "Adaptive variable-structure basis function expansions: Candidates for machine learning," *Information Science*, vol. 496, pp. 124-149, 2019.

[3] C. S. Lee, Y. L. Tsai, M. H. Wang, W. K. Kuan, Z. H. Ciou, and N. Kubota, "AI-FML agent for robotic game of Go and AIoT real-world co-learning applications," *2020 World Congress on Computational Intelligence* (*IEEE WCCI 2020*), Glasgow, Scotland, UK, Jul. 19-24, 2020.

[4] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, pp. 484–489, Jan. 2016.

[5] C. S. Lee, M. H. Wang, L. W. Ko, Y. Hsiu Lee, H. Ohashi, N. Kubota, Y. Nojima, and S. F. Su, "Human intelligence meets smart machine: a special event at the IEEE International Conference on Systems, Man, and Cybernetics 2018," *IEEE Systems, Man, and Cybernetics Magazine*, vol. 6, no. 1, pp. 23-31, Jan. 2020.

[6] C. S. Lee, M. H. Wang, Y. L. Tsai, L. W. Ko, B. Y. Tsai, P. H. Hung, L. A. Lin, and N. Kubota, "Intelligent agent for real-world applications on robotic edutainment and humanized co-learning," *Journal of Ambient Intelligence and Humanized Computing*, 2019. (DOI: 10.1007/s12652-019-01454-4).

[7] C. S. Lee, M. H. Wang, L. W. Ko, N. Kubota, L. A. Lin, S. Kitaoka, Y. T Wang, and S. F. Su, "Human and smart machine co-learning: brain-computer interaction at the 2017 IEEE International Conference on Systems, Man, and Cybernetics," *IEEE Systems, Man, and Cybernetics Magazine*, vol. 4, no. 2, pp. 6-13, Apr. 2018.

[8] C. S. Lee, M. H. Wang, S. C. Yang, P. H. Hung, S. W. Lin, N. Shuo, N. Kubota, C. H. Chou, P. C. Chou, and C. H. Kao, "FML-based dynamic assessment agent for human-machine cooperative system on game of Go," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 25, no. 5, pp. 677-705, 2017.

[9] J. Castanon, "10 machine learning methods that every data scientist should know," May 2019, [Online] Available: https://towardsdatascience.com/10-machine-learning-methods-that-every-data-scientist-should-know-3cc96e0eeee9.

[10] Wikipedia, "Adaptive neuro fuzzy inference system," 2020 [Online] Available:https://en.wikipedia.org/wiki/Adaptive_neuro_fuzzy_inference_system

[11] J. S. R. Jang, "ANFIS: Adaptive-Network-based Fuzzy Inference Systems," *IEEE Transaction on Systems, Man, and Cybernetics*, vol. 23, pp. 665-685, May 1993.

[12] Y. Tian and Y. Zhu, "Better computer Go player with neural network and long-term prediction," Feb. 2016, [Online] Available: https://arxiv.org/abs/1511.06410.

[13] Y. Tian, J. Ma, Q. Gong, S. Sengupta, Z. Chen, J. Pinkerton, and C. L. Zitnick, "ELF Open Go: An analysis and open reimplementation of AlphaZero," May 2019, [Online] Available: https://arxiv.org/abs/1902.04522.

[14] DeepMind, AlphaGo Master series: 60 online games, Jan. 2020, [Online] Available: https://deepmind.com/alphago-master-series