RoboCup Asia-Pacific 2021

Team Description Paper

(Cover Page)

League Name:	Soccer Small Size League
Age Group:	Major
Team Name:	KIKS
Team Website:	www.ee.toyota-ct.ac.jp/~sugi/RoboCup.html
Participants Name:	Toko Sugiura Yusei Naito Yasutaka Tsuruta Ryoma Mitsuoka Kosei Naito Hayato Mitsuda Daichi Miyajima Hironobu Suzuki Chihiro Takanashi Yota Dori Ryuto Tanaka
Mentor Name:	
Institution:	National Institute of Technology, Toyota College
Country:	Japan
Contact Person:	Toko Sugiura
Contact Email:	sugiura.toko@toyota.kosen-ac.jp
Date:	10 August 2021

RoboCup Asia-Pacific 2021

Team Description Paper

Soccer Small Size League

Yusei Naito, Ryoma Mitsuoka, Yasutaka Tsuruta, Kosei Naito and Toko Sugiura

KIKS, National Institute of Technology, Toyota College, Japan

1. Abstract

This paper is used to qualify as participation to the RoboCup Asia-Pacific 2020 soccer small size league. Our team's robots and systems are designed under the RoboCup 2020 rules. In this paper for hardware, it is described about the various idea of dribble mechanism, whose importance was again emphasized in the final of RoboCup 2019 Sydney. In software, the implementation of UCT (Upper Confidence bound applied to Trees) in MCTS (Monte Carlo Tree Search) method was performed to achieve the action-decision of the motion for the robots. Moreover, we tried to use Deep Learning Library (Sony products) and Neural Network using GUI (Graphical User Interface) for layered design, learning and evaluation in AI system.

2. Introduction

KIKS is the team consisting of students and faculty of mainly Department of Electrical and Electronic Engineering in National Institute of Technology, Toyota College. About 10 members work after school every normal day. We have continuously participated RoboCup small size league since 2004.

Websites:
http://www.ee.toyota-ct.ac.jp/~sugi/RoboCup.html
https://twitter.com/hashtag/kiks_ssl

3. Strategy

Hardware Research Overview In RoboCup SSL, to realize a strategy based on AI, it is necessary to have a mechanism to accurately kick the ball and a mechanism to accurately receive the ball. Our present robots, however, did not have enough performance of a mechanism for holding the ball, and often bounce off the ball during the game. Also, when the ball was kicked, the ball sometimes rolled in a different direction from the AI command because the ball was not held at the center of dribbling bar. If we can improve the ball holding ability, the dribbler will be an effective tactics like ZJUNlict in final game of RoboCup 2019 Sydney. In addition, when the dribbler's rotation can be sufficiently transmitted to the ball, a curve shot may be realized, which will be effective in expansion of AI strategies. Therefore, in this section, it focuses on dribbling performance which improves speed and stability for a ball and describes an idea of new dribblers to improve the ball holding ability.

Present robot We have been using a robot with a 70W brushless DC motor as the drive motor since 2017. By using this motor, the torque of the robot is higher than previous one. In addition, direct drive is possible because no gear for deceleration is required. In addition, the omni wheel has 20 small tires per wheel. As a result, the shape of the omni-wheel is more like a regular circle. This is a change that improves the acceleration performance of the robot because the maximum speed has been improved by changing the motor. Typical our robots are shown in Fig. 1. Specification of present robot is tabulated in Table 1.

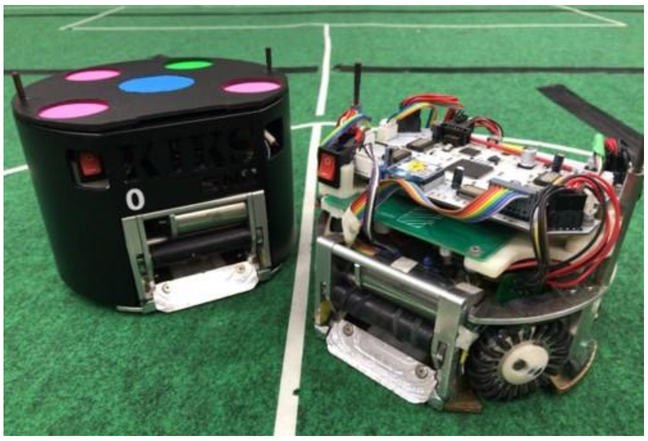


Fig. 1. Present our robots

Table 1. Specification of present robot

Dimension	ϕ180×135mm
Total weight	2040g
Driving motors	Maxon EC-45 flat 70W
Gear	No use
Wheel diameter	50mm
Dribbling motor	Maxon EC-max 22 25W
Dribbling gear ratio	28:20
Dribbling bar diameter	15mm
MCU	ATMEL AT32UC3B0512
FPGA	Xilinx XC6SLX9
Communication link	Digi XBee-pro 2.4GHz

Idea of dribbling device In the RoboCup 2019 Sydney final, the importance for dribbling performance was once again demonstrated. So we tried an approach to receive the ball in center of the dribble bar. In general, the speed of a dribble roller increases with increasing the radius. Therefore, by making the central part thicker and the ends narrower, a force toward the central part is applied to the ball. We propose a mechanism that can change the angle of the dribble bar using a servomotor. The 3D CAD image is shown in Fig. **2**. With this mechanism, when the ball is held, an infrared sensor recognizes the position of the ball and adjusts the angle of the dribble bar based on the result. If the ball moves to the left center of the dribble bar, for example, the ball can be returned to the center by changing the angle of the dribble bar. This makes it possible to keep the position of the ball at the center of the dribbling bar at any time.

In addition, even when the robot is dribbling, the angle of the dribble bar can be properly changed by using the servo mechanism. Therefore, it is possible to continue dribbling without releasing a ball.

Since the effectiveness of these performances has not been confirmed yet, detailed verification will be examined in near future.

In RoboCup SSL, a wide choice for kicking mechanisms including straight kick, chip kick and curve kick is effective in expanding the range of AI strategies. The mechanism of the curve kick has been already reported by OP-AmP [1] and RoboTeam Twente [2], but we also considered its introduction. We propose a new method for curve kick using a movable dribbler. Using the prototype mechanism shown in **Fig. 3**, we tried to realize a curve kick by applying a diagonal force to the rotating ball. It is possible to achieve by changing the angle of the dribbling-bar with a servomotor before kicking. It was confirmed that the trajectory of the ball after kicking can be controlled by changing the angle of the movable dribbler. But, it does not implement yet, because of some problems are occurred for mounting a dribbling motor and link mechanism compactly and for considering the shape and material as same as conventional dribbler. In near future, due to control the rotating speed of dribbler, the orbital radius of the ball will be able to change and expand the application range of tactics. It will also give advantage in penalty kick.

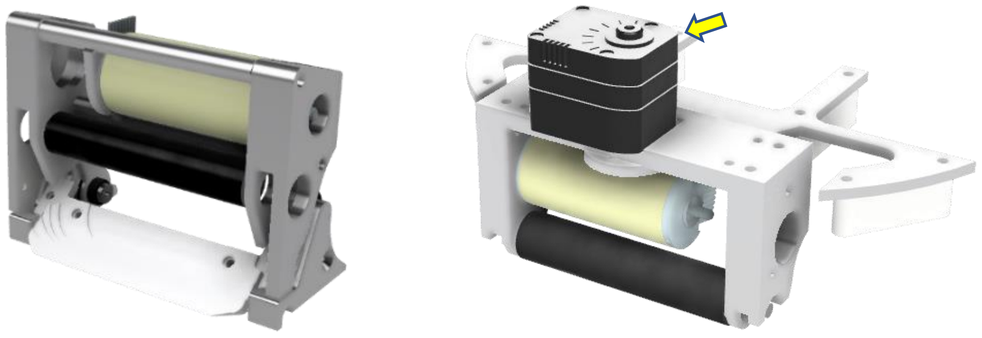


Fig. 2 3DCAD image of dribbler (conventional:Left and New one with servo-mechanism:Right).

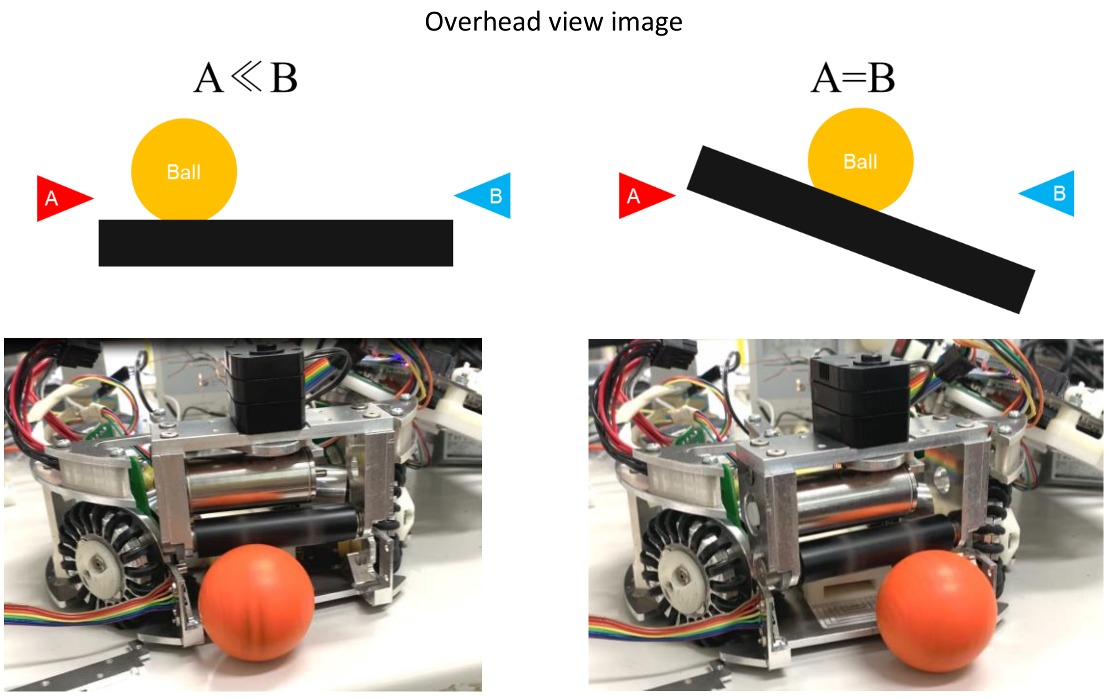


Fig. 3 Dribble bar angle changing mechanism using a servomotor

Software Research Overview In recent years, the field size and the number of robots is expanding in RoboCup SSL. The conventional decision-making algorithm has problems such as not considering prospects and poor expandability. It is impossible to realize cooperative actions such as passes during in play and resulting in a low score rate. Therefore, we tried to apply a technique that combines Monte Carlo tree search (MCTS) and neural network (NN), which has been successful in the field of game AI in recent years. We tested this algorithm in practice games.

Introduction of Neural Network Libraries / Neural Network Console We have introduced the Neural Network Libraries (NNabla)[3] and Neural Network Console[4] provided by Sony Network Communications Inc. to implement Deep Learning. The neural network is designed, learned and evaluated using the Neural Network Console, and inference is performed using C++ API of NNabla. Neural Network Console can be operated with GUI, and trial and error are possible efficiently.

About MCTS When a robot plays soccer, there are various action options such as pass, shoot, and dribble. Significant rule changes in recent years have increased the number of robots and the size of the field. Rule changes may continue in the future. Therefore, decision-making for motion of the robots considering prospects will be more important in strategy. We implemented an algorithm that combines Monte Carlo Tree Search (MCTS) and Neural Networks with reference to AlphaZero [5,6] developed by DeepMind. MCTS is a search method based on simulation results. This method has been used in the field of Go etc. [5,6], but we applied it to soccer by inferring the success probability of action.

Decision-making algorithm using Monte Carlo Tree Search The nodes of the game tree have information of the position, elapsed time, Action, Reward, and Visit-count for the robot in the field. The node transitions by Action. At this time, the transition probability to each node is derived by inferring the success probability of Action using a Neural Network. This inference is also applied to the MCTS Rollout (random simulation), and it is possible to obtain the expected value of the reward and to search for promising actions. Typical game tree is shown in Fig. 4.

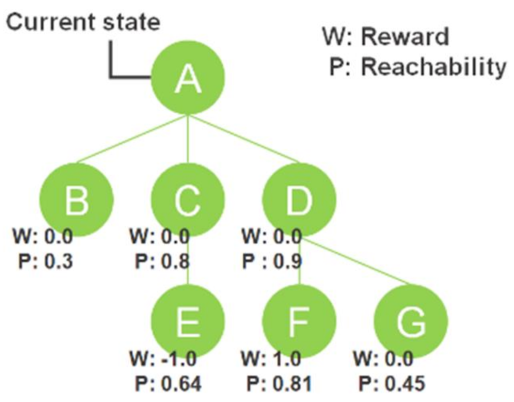


Fig. 4. Typical game tree for decision-making

MCTS is an algorithm that expands and searches a tree based on random sampling. Compared to the αβ method generally used in Japanese chess (Shogi), it has the advantage that there is no need to create and adjust complex evaluation functions. MCTS consists of five elements, i.e., Selection, Expansion, Evaluation, Backup, and Play for the process as shown in Table 2. First, the algorithm creates a root node with the state of the observed field. Next, selection, expansion, evaluation, and propagation from the root node are repeated according to the search situation until a certain time has elapsed. After that, the node with the highest number of selections is selected from the child nodes connected to the root node, and the action to execute is made decision.

Table 2. Process elements on the Monte Carlo Tree Search

Process	Definition
Selection	Select the child node with the highest UCB1 value.
Expansion	When a ball is in playing field even if the search reaches a leaf node, the node is extended in case that the Visit-count is greater than a certain value.
Evaluation	When a ball is in playing field even if the search reaches a leaf node, the result of Rollout (random simulation) from that state is added to the reward of the node in case that the Visit-count is less than a certain value.
Backup	When a ball is not in playing field and the search reaches a leaf node, the state evaluation value is propagated to the parent node.
Play	After the search is completed, the node with the highest Visit-count is selected, and the action to be performed is determined.

The UCB1 (Upper Confidence Bound) [7] described in eq. (1) was used as the selection criteria for child nodes to be searched. UCB1 is well known for the multi-armed bandit problem, and it is possible to search in consideration of the evaluation value and accuracy for decision branches. where, w_i and n_i show the Reward and the Visit-count for nodes *i*, respectively. The *t* indicates the Visit-count for all of child nodes, *c* is the constant. In eq. (1), the first term indicates the average of the reward, and the second term means the value that gives the reliability of the evaluation. The second term becomes larger as the Visit-count (evaluation value) to the node is smaller than the total Visit-count and is easily selected. That is, the evaluation value increases as the reliability of the first term decreases.

$$UCB_i = \frac{w_i}{n_i} + c\sqrt{\frac{2\log(t)}{n_i}} \qquad (1)$$

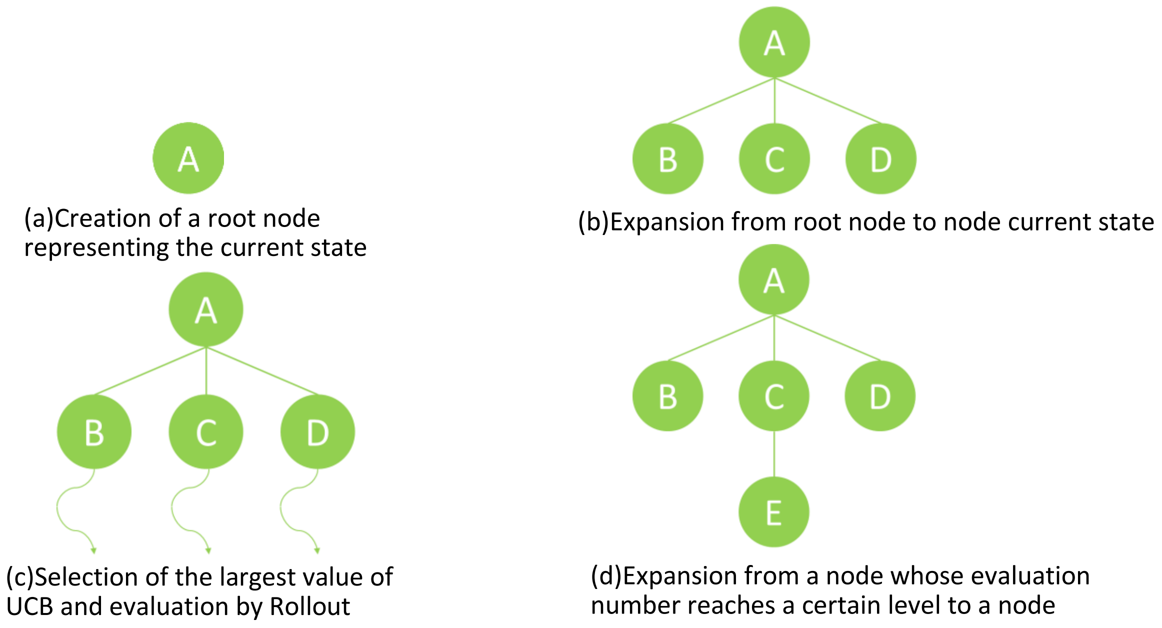
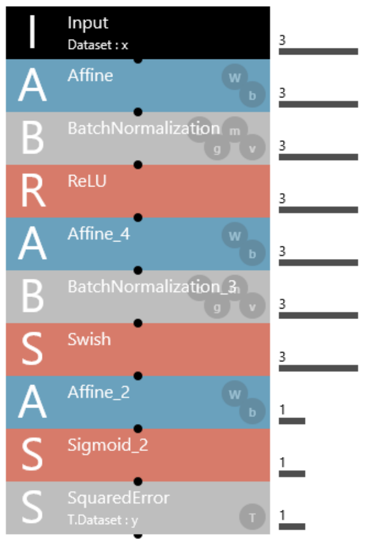

A
B
C
D
E
(a)Creation of a root node representing the current state
(b)Expansion from root node to node current state
(c)Selection of the largest value of UCB and evaluation by Rollout
(d)Expansion from a node whose evaluation number reaches a certain level to a node

Fig. 5. MCTS execution procedure

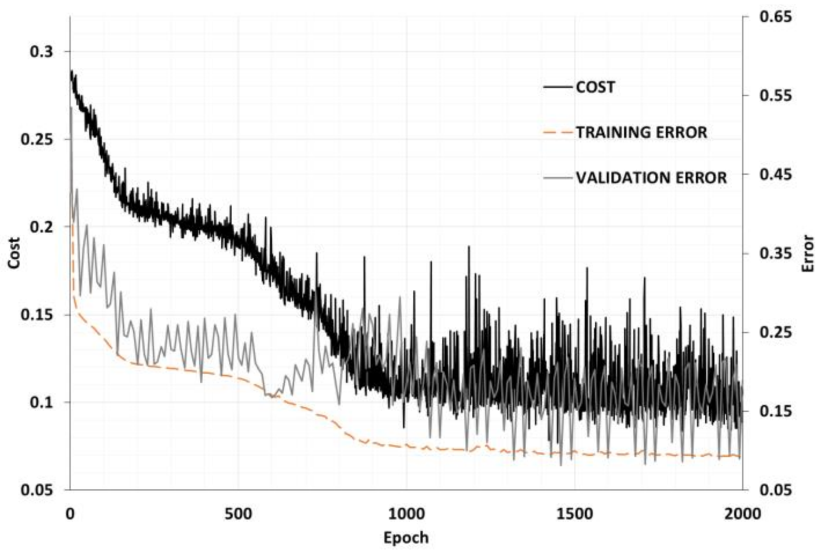
The ϵ-greedy method, in which the evaluation value is randomly selected with a certain probability, was also studied. In this study, however, UCB1 that mathematically guarantees convergence to the optimal solution was used. The execution procedure of MCTS is shown in Fig. 5. The update of the evaluation value and the expansion of the tree are repeated in Fig. 5 (c) and (d). Finally, the action that transitions to the child node with the highest Visit-Count is executed except the root node. For example, in Fig. 5, an action to transition to any of nodes B, C and D is executed.

Neural Network Designs The success probability of actions such as pass and shoot is inferred by a neural network. The processing for the enemy robots is performed one by one, and the inferred values of all the robots existing on the field are combined. Also, when creating teaching data and performing inference, it was assumed that there was no enemy robot farther than the target position at the start position of the action. That is, the inference was based only on the distance to the robot located in front of the target position. For the input data, it was used the ball speed, the distance between the ball and any enemy robot, and the angle between the ball and any enemy robot. The output is that the ally robot can either act or not, without intercepting the ball by the enemy robots. The reachability inference NN described here was composed of all connected layers, and several NNs with different numbers of layers, nodes, and activation functions were compared.

Learning for Neural Networks More than one hundred learning data were obtained from the game simulation on grSim[8], and sent to Neural Network Console and NNabla[3,4]. For the learning, it was performed with a batch size of 32 and 2000 epochs.



(a) Neural Network structure



(b) Typical Learning Curve

Fig. 6. Neural Network structure (a) and learning process (b).

Evaluation of neural network performance Fig. 6 shows the structure and learning curve of the introduced reachability inference NN. The success probability of action is inferred by a NN structure with all layers of connections as shown in Fig. 6 (a). This network infers the probability that an enemy robot will get a moving ball. This inference is applied to all enemy robots on the field to calculate the probability that the ball can move at any position and any speed. Execution example of learning process is shown in Fig. 6(b). The horizontal axis shows the number of epochs (generations), the left vertical axis shows the output of the cost function, and the right vertical axis shows the error rate. TRAINING ERROR and VALIDATION ERROR indicate error rates for learning data and for verification data, respectively. The introduced NN has two hidden layers. The first hidden layer has three nodes, and the activation function is ReLU. The second hidden layer has three nodes, and the activation function is Swish. The time until completion for learning was 36.2sec. Obtained probability are tabulated in Table 3. The accuracy of the inferred value was about 84%, which was enough for use in the actual games.

On the other hand, Table 4 shows the accuracy of NNs with different numbers of node in the hidden layers. As the result, it can be confirmed that the accuracy is lower than that of introduced NN.

Table 3. Probability predicted by Neural Network on Confusion Matrix (50 samples)

Data\Predicted probability	<0.5	≥0.5	Recall
FALSE	18	7	0.72
TRUE	1	24	0.96
Precision	0.9473	0.7741	
F-Measures	0.8181	0.857	
Accuracy	0.84		
Average of Precision	0.8607		
Average of Recall	0.84		
Average of F-Measures	0.8376		

Table 4. Comparison of Accuracy for NN under varying conditions

Condition	First hidden layer: 8, Activ. func.: ReLU Second hidden layer: 4, Activ. func.: Swish	First hidden layer: 5, Activ. func.: ReLU Second hidden layer: 3, Activ. func.: Swish	First hidden layer: 3, Activ. func.: ReLU Second hidden layer: 3, Activ. func.: ReLU
Accuracy	0.76	0.76	0.64

Evaluation of decision-making algorithm An example for the action of this algorithm described above is shown in Fig. 7. The circle in the lower center represents the ball, and the markers numbered B or Y indicate the robots of each team (Observation on AutoReferee [9]). First, at the lower left in Fig. 7, the B5 robot is moving toward the opponent's goal after passing the ball toward B1. After that, by receiving the ball again from B1 in front of the goal (ducking and weaving the opponent robots, Y1 and Y4), it is found that the B5 robot will attack toward the goal. We introduced this algorithm to real game (RoboCup JapanOpen 2019) and confirmed that an effective pass play could be realized. Furthermore, we plan to use the selection probability and evaluation value for Action as the inference value of Neural Network. As the result, the efficiency of the tree search is improved, and the range of the decision branch for the action will be wider.

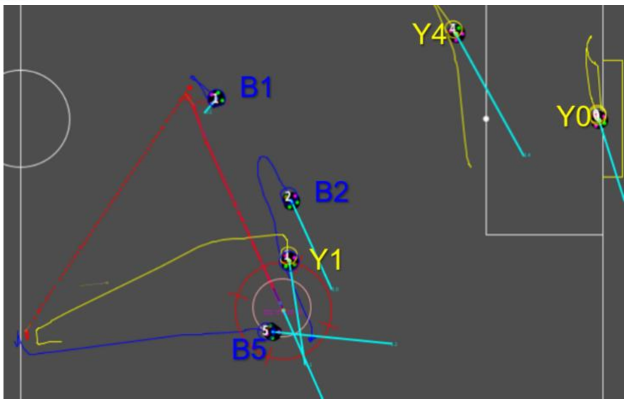


Fig. 7 An example for the action of this algorithm on grSim

Planned Neural Network structure is shown in Fig. 8. In this network, we consider a matrix of 1m intervals. Only once per frame, it receives three channels as input, i.e., the positions for a ball and both team's robots, and outputs the approximate expected score and the distribution of the selection probabilities for action candidates. One of the problems of this method is that the computational complexity is greater than conventional methods. As the solution, now we are studying about the introduction of GPU for NN processing, and the separation and parallelization of this method.

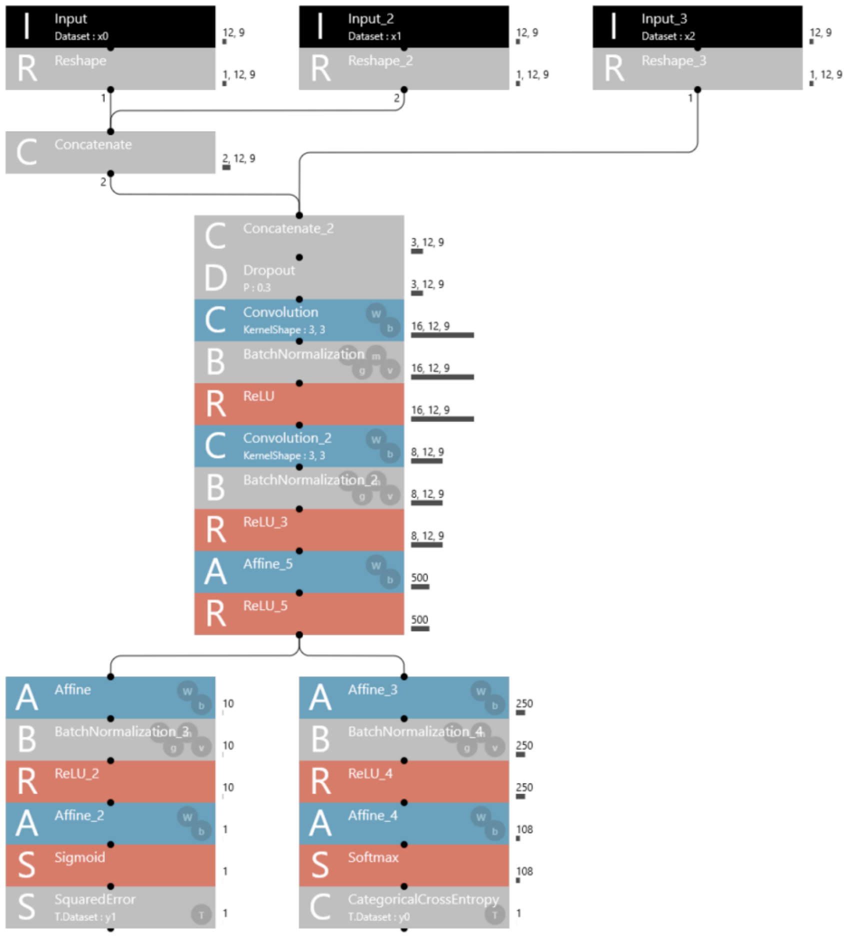


Fig. 8 Planned Neural Network structure for inferring expected score and the distribution of the selection probabilities for action candidates.

4. Track Record

Up to now, we won 3rd place in 2012, 4th place in 2011 and 2010 in RoboCup SSL world competition, and also won 3-times and 4-times champion in RoboCup Japan Open SSL and SSL-H, respectively. In RoboCup2021, we got 5th place in upper tournament of virtual competition and 6th place in hardware challenge of SSL. Moreover, as other robotics competition achievement, we won First place in 1994, 1998 and Grand Champion in 2004 in all KOSEN ROBOCON held in Japan.

5. Discussion and Conclusion

At the RoboCup Japan Open in Nagaoka 2019 and RoboCup virtual competition 2021, it was confirmed that our system could mostly realize the pass plays. The team won the third place and fifth place, respectively. For the hardware, we devised two type dribbler which uses two small rollers to improve stability and dribbler which uses a parallel link mechanism. Now we are developing these prototype dribblers. We are going to compare the improved dribbler with conventional one. Based on that experiment, we will compare it with the current dribbler and make improvements to make it a better dribbler. For the software, we implemented an algorithm using neural networks and Monte Carlo tree search. These technologies can be applied in other areas as well. We are also looking at introducing neural networks at other layers, such as control and action.

Acknowledgements

The authors wish to express their deep appreciation to SNB corp. for providing the covers and some parts of the robots. This work was supported in part by a Grants-in-Aid for Scientific Research (C) (JSPS KAKENHI Grant Number 20K03263), a Grant-in-aid from Chuden Foundation for Education and The Nitto Foundation.

References

[1] Takamichi Yoshimoto, Takato Horii, Shoma Mizutani, Yasuyuki Iwauchi,Yutaka Yamada, Kousei Baba, and Shota Zenji: OP-AmP 2017 Team Discription Paper, https://www.robocup2017.org/le/symposium/soccer sml size/Robocupssl2017-final9.pdf (2017).
[2] Cas Doornkamp, Zahra van Egdom, Gal Humblot-Renaux, Leon Klute, Anouk Leunissen, Nahuel Manterola, Sebastian Schipper, Luka Sculac, Emiel Steerneman, Stefan Tersteeg, Christophe Vanderwalt, Wouter van Veelen, Haichuan Wang, Jeroen Weener, Jelle Zult: RoboTeam Twente 2018 Team Description Paper, https://ssl.robocup.org/wp-content/uploads/2019/01/2018 TDP RoboTeam Twente.pdf (2018).
[3] Neural Network Libraries, Sony Network Communications Inc., https://nnabla.org.
[4] Neural Network Console, Sony Network Communications Inc., https://dl.sony.com.
[5] D. Silver, J. Schrittwiser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. v. d. Driessche, T. Graepel and D. Hassabis: Mastering the game of Go without Human Knowledge, nature, 2017.
[6] D. Silver, A. Huang, J. C. Maddison, A. Guez, L. Sifre, G. v. d. 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, 2016.
[7] P. Auer, N. Cesa-Bianchi and P. Fischer: Finite-time Analysis of the Multiarmed Bandit Problem, Machine learning, 47(2-3):235–256, 2002.
[8] M. Valiallah (Mani), A. Koochakzadeh and S. S. Ghidary: grSim – RoboCup Small Size Robot Soccer Simulator In Robot Soccer World Cup, pp. 450-460. Springer Berlin Heidelberg, 2011.
[9] TIGERs Mannheim Team: AutoReferee, [Online]. Available: https://github.com/TIGERs-Mannheim/AutoReferee.
[10] G. M.-B. Chaslot, M. H. Winands and H. J. v. d. Herik, Parallel Monte-Carlo Tree Search, in Proc. Comput. and Games, LNCS 5131, Beijing, China, 2008, pp. 60–71, 2008.