PIFE: Permutation Invariant Feature Extractor for Danmaku Games

***Abstract*—Touhou Project is one of the best-known games in the bullet hell genre, which is a game that a player dodges complex patterns of bullets on the screen. In this game, the agent needs to react to the environment in real-time, which made existing methods having difficulties processing the high-volume data of objects; bullets, enemies, etc. We introduce an environment for the Touhou Project game '東方花映塚 ～ Phantasmagoria of Flower View.' which manipulates the memory of the running game and enables to control the character. However, the game state information consists of unstructured and unordered data which is not amenable for training existing reinforcement learning models, as they are not invariant to order changes in the input. To overcome this issue, we propose a new pooling-based reinforcement learning approach that is able to handle permutation invariant inputs by extracting abstract values and merging them in an order-independent way. Experimental results corroborate the effectiveness of our approach which shows significantly increased scores compared to existing baseline approaches.**

***Index Terms*—reinforcement learning, permutation invariance, pooling, touhou**

I. Introduction

In danmaku game, also known as bullet-hell shooting game, the player attacks the enemy on a 2D screen while controlling the player's ship to avoid being hit by a large number of slow bullets and moving enemies that fill the screen. These objects, observed in lists of information are uncertain in numbers and in order. Suppose there are data of two bullets 'A' and 'B' stored in the game's memory. Since the order of data stored in the game is arbitrary, there exists an equal possibility of observing the environment as (AB) or (BA). Despite that deep reinforcement learning has achieved significant performance in many domains such as playing Go [1] and Atari games [2], this feature of danmaku games makes these standard approaches struggle since they may act differently even when only the order of the data is changed In other words, the commonly used method for reinforcement learning has a weakness in understanding the underlying meaning of the observed data.

Another difficulty posed by danmaku games is that the agent is required to have quick and precise reactions. Compared with traditional 2D shooting games such as 'space invaders' or 'xevious', the bullets in danmaku games are much greater in numbers, cover the major part of the screen, and follow complex patterns, making it non-trivial to dodge them and requires lots of strategy and memorization. In terms of machine learning, the agent is required to process the huge data in real-time while matching patterns with previously known ones to make the right decision necessary for survival.

In terms of machine learning, danmaku game poses a difficulty that the agent is required to have quick and precise reactions. The bullets in danmaku games are much greater in numbers, cover the major part of the screen, and follow complex patterns, making it non-trivial to dodge them and requires lots of strategy and memorization. Furthermore, the agent is required to process the huge data of bullets and enemies in real-time while matching patterns with previously known ones to make the right decision necessary for survival.

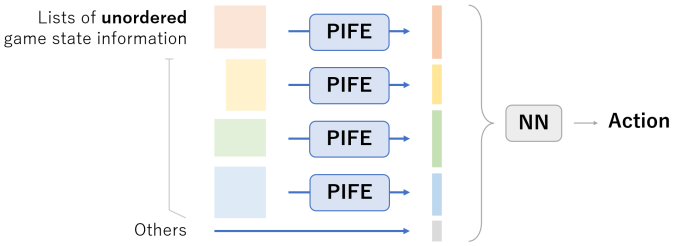

Fig. 1. Our permutation invariant deep reinforcement learning model. Unordered input data is processed by PIFE that extracts the abstract values which is passed to NN.

Providing a new method for deep reinforcement learning that enables it to handle a huge data with permutation invariance is not only useful for danmaku games but also widens its ability to many other fields. For example, autonomous driving uses sensors to detect the obstacles which varies in time and is observed in no such order. Our model is able to handle these inputs very quickly.

To address this issue, we propose a new learning method using a Permutation Invariant Feature Extractor (PIFE) based on PointNet [3], [4]. We interpret the data given from the environment as an unstructured set and extract abstract values that feature the data prior to processing with a normal reinforcement learning method. The overview of our method, shown in Fig. 1, shows that the input data are first processed with PIFE. An added benefit is that this leads to increasing the speed of the network since our method enables to decrease the dimension of the input data.

As a representative example of danmaku games, we focused on a Touhou Project game '東方花映塚～Phantasmagoria of Flower View' [5]. We implemented an OpenAI gym environment [6] for this game. This environment consists of a server that handles multiple game instances and a client that connects to the server to receive the observed data and control the character (more details in Sec. III-A). The server launches an instance of the game for each connection with a client and uses DLL injection [7] to extract the current state of the game that exists in the memory of the game.

In summary, our main contributions are as follows: (a) We created a permutation-invariant deep reinforcement learning method to deal with inputs that are lists of data. Compared with classic deep networks to show our method performs better with

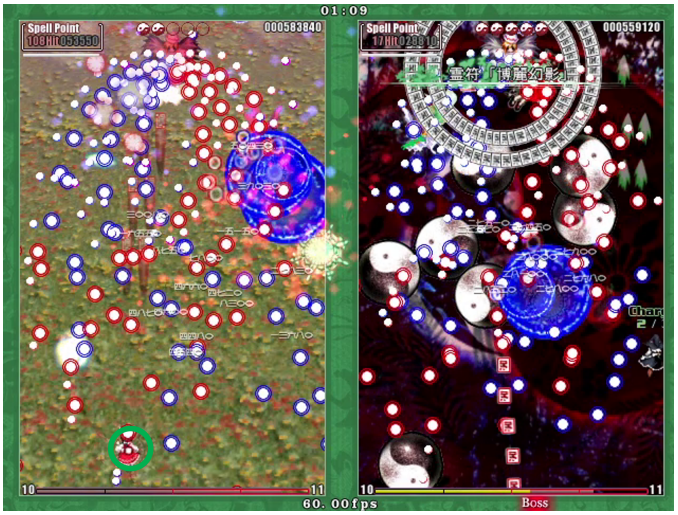

Fig. 2. A capture of a danmaku game. The player controls the red character (marked with green circle) in the left screen and the right screen is for the opponent. All other objects are bullets, enemies etc. which the player should dodge and survive longer than the opponent.

less computational force. (b) We implemented an OpenAI gym environment of a danmaku game, 'Phantasmagoria of Flower View' to advance the research in the field.

II. Backgrounds

A. Danmaku Games and Touhou Series

Danmaku game, also known as bullet hell games, is a popular game genre that is similar to 'Shoot'em ups' games. Some examples of the shoot'em up games are the '*Space Invaders*' and the '*Asteroids*'. In these games, the player controls a character in the game which can shoot bullets. There are enemy characters on the screen and they shoot bullets as well. The goal of the game is to hit the enemy characters with the bullet while dodging their bullets. Fast reaction and very precise control is required for players to score better. Danmaku games also share the same feature with shoot'em up games, however, the number of enemies and the number of bullets they shoot are huge that the enemy's bullets fill the screen and there are not much space to dodge the bullets. This feature distinguishes danmaku games from other shooting games and the difficulty has attracted a lot of players for a long time. Fig. 2 shows a snap shot of danmaku games. The player controls the red character placed around the middle of the left game screen. All the white, red, yellow bullets are from the enemies and the player has to dodge all of them through the tiny gaps between the bullets.

Among other danmaku games, the *Touhou Project* series created by the *Team Shanghai Alice* is very popular and famous. Therefore, as a representative example of danmaku games, we focused on one of the Touhou Project game '東方花映塚～Phantasmagoria of Flower View' [5]. Unlike other Touhou Project games, 東方花映塚 (Kaeiduka) is a competitive game, where the objective is to defeat the opponent by sending more bullets to the opponent's field and eventually shoot them down. The player can move up and down, left and right, fire bullets, and use spell cards to send a barrage to the opponent's field. Thanks to this feature, it will be possible to play against the agent after being well trained to know how well it performs. Additionally, the game features large amounts of randomness which force reinforcement agents to have to react to novel circumstances encountered during gameplay.

B. Permutation-Invariant Networks

Within the field of supervised learning, there exists many fields of studies according to the special features of the input dataset. Permutation-invariant dataset is one of the interesting fields for supervised learning. Permutation invariance means that the order of objects in a list does not matter to determine the feature or the meaning of the dataset. One example of a permutation invariant dataset is point cloud which the data is the coordinate of points in a multi-dimension space. The order of the input data has no 'meanings' to what the data suggests. The order of the coordinates could be completely shuffled, yet the overall shape would not change at all. In order to train a model with these kind of dataset efficiently, the output from the model should be exactly same regardless of the order of the input data.

To extract the feature of point cloud, PointNet [3] and PointNet++ [4] were suggested. PointNet is used for solving clustering and classification problems on point cloud data and is mainly applied to automatic driving and construction projects. PointNet is characterized by its ability to learn regardless of order invariance [3], movement invariance [3], and locality [4]. Given a set of n data, n-Max pooling is performed in the last layer to obtain an output of arbitrary dimensionality regardless of the number of data.

III. Proposed Approach

Our method for training the agent has the following components: the permutation invariant reinforcement learning agent, the game server who launches the game, and the game client (gym environment) who connects the agent to the server.

A. Environment and Setup

Unfortunately, there has not been any previous attempts of reinforcement learning applied to danmaku games. Meaning that there are no existing implementations or wrappers of the game that enables interaction using python. In order to do machine learning with this game, we created a gym environment: 'Touhou Gym'.

The Touhou Gym is based on the OpenAI gym environment [6]. It is composed of two main components: the game server and the game client. The game server will wait for a connection from the client, and for each connection, the server starts a new game instance that collects the data using DLL injection in the game and sends it to the client. After the connection between the client and the server is established, the client provides the game data to the agent.

B. Network Structure

1) Permutation Invariant Feature Extractor (PIFE): As we mentioned in Sec. III-A, there are four types of objects that

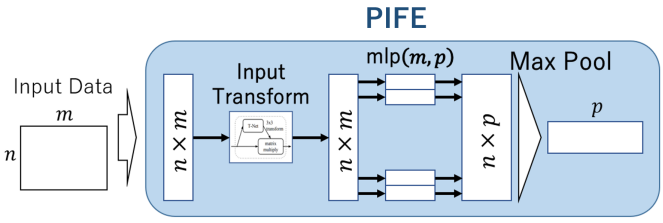

Fig. 3. Overview of the Permutation Invariant Feature Extractor (PIFE) used in our architecture.

are permutation invariant in this game; *Enemies*, *Bullets*, *Items*, and *ExAttacks*. In our model, we use four different Permutation Invariant Feature Extractors (PIFE) and then concatenate the output for further processing with the rest of the model. The structure of PIFE shown in Fig. 3 is inspired from PointNet and consists of three modules; an InputTNet, followed by a multi layer perceptron (MLP), and finally, a max-pooling layer outputs the feature vector of the given input. The InputTNet aims to normalize the input data. The output is then passed to an MLP and then to a max-pooling layer with the number of input objects gives the feature vector of the given input data. Through the use of max-pooling, the output becomes the maximum value of the input values. Since the maximum value is not affected by the order of the data, the PIFE is able to extract the features that are invariant to the order of the input.

2) Reinforcement Learning Methods: The player data and the outputs of the four PIFE are concatenated and processed by traditional reinforcement learning methods. Since all the permutation-invariant data from the environment are processed by PIFE, the network is able to select an action regardless of the order and only by the underlying meaning of the environment with traditional methods. In our experiment, three methods were used to calculate the optimal action given the player data and the outputs of the PIFEs; Dueling Double DQN (D2DQN), Advantage Actor Critic (A2C), and Proximal Policy Optimization (PPO).

IV. Comparison

Our goal is to show the effectiveness of our method on environments with permutation invariant datasets i.e. Touhou Gym. Therefore, we trained two variations of models; one that has permutation invariance and one that does not using the same gym environment. The model without permutation invariance uses a two layer Fully Connected neural Network (FCN) instead of the Permutation Invariant Feature Extractor (PIFE). In this experiment, we used prioritized experience replay with mini-batches of size 64. The behavior policy was ε greedy with ε annealed from 1 to 0.01 and fixed at 0.01 thereafter. Both models trained for 150 games which are roughly 150,000 iterations. The parameters of the D2DQN model were copied to the second network after each game. We utilized PFRL [8] for experiments.

Tab. I shows the number of parameters in the neural network of each model. It is clear that the number of parameters of the model using dueling neural network with permutation invariant is about five times less and model for A2C using permutation-invariant is 20 times less than that of FCN. This difference not only increases the training efficiency but also enables the agent to react quicker to the game due to less computational time.

TABLE I
Number of parameters in the neural network of each model.

	D2DQN	A2C	PPO
FCN	234,482,021	201,587,344	234,482,021
PIFE	43,104,035	10,209,358	43,104,035

TABLE II
Evaluation result of all agents. Average of 10 game plays.

	FCN + D2DQN	PIFE + D2DQN	A2C	PPO
Ave. score	33798.49250	46700.18732	31601.54720	35596.64989
			26153.88531	**48222.17446**
Ave. survival frames	10525.2	13269.6	9312.4	11663.3
			8257.0	**13642.8**

A. Quantitative Results

Fig. 4 shows the comparison of the total score of each game using D2DQN and other methods. The orange line shows the change of the total scores of each game during the training using PIFE, and the blue line shows it of FCN. Comparing the two lines, we can point out that figure of D2DQN and PPO shows more rapid growth in PIFE compared to FCN. Besides, PPO reached its peak after approximately 50 epochs. This means that the model with permutation invariance is able to learn more efficiently. However, looking at A2C, the growth of score does not seem to vary between using PIFE and FCN. Besides, the line of each method seem to only have small improvements. In the case of A2C, it seems that the training itself had a problem somewhere.

After training each agent for 150 epochs, we evaluated the agents by making them play the game 10 times. The average scores and the average number of frames the agent survived are listed in Tab. II. Judging from the results, PIFE + PPO showed the best performance of all and methods using PIFE scored better than that of FCN except when combined with A2C.

B. Qualitative Results

Fig. 5 to Fig. 8 are snapshots of the agent trained using our PIFE playing the game. The pictures are taken rapidly and shown from left to right. The character controlled by the agent is the red character marked with a yellow circle. Fig. 5 is a sequence when the agent is successfully attacking the opponents. The agent controls the character to move towards the enemies and attacks them by shooting the bullets upwards. Fig. 6 shows when the agent successfully avoids the bullets in a very tough situation. The character is surrounded by bullets, bars from below (which are also objects that should be avoided), yet avoids all without being damages and escapes from the difficult area. These pictures show that the agent is

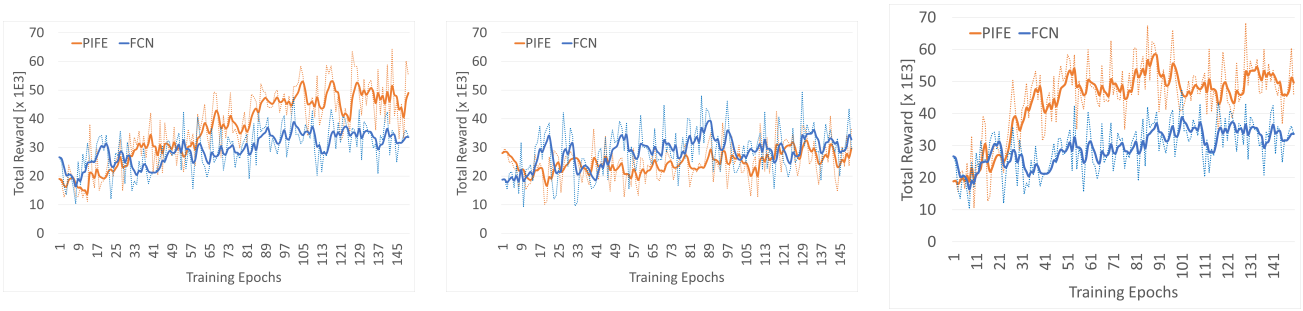

Fig. 4. The total score of each game while training between FCN and PIFE using D2DQN (left), A2C (middle), and PPO (right).

Fig. 5. The agent successfully attacks enemies.

Fig. 6. The agent successfully dodges enemies.

Fig. 7. The agent successfully avoids enemies and bullets.

Fig. 8. The agent fails to avoid irregular bullets.

successfully learning how to play the game, and to obtain more scores.

Despite that the quantitative results of A2C (Fig. 4) did not show good results and the total scores of each game did not increase, from a qualitative perspective, the agent seemed to be learning how to dodge the bullets pretty well. Around the third iteration the agent already learned how to dodge the normal bullets. However, the agent was not able to dodge some uncommon enemies and bullets like the red big dots seen in Fig. 8. The ability to avoid these kind of uncommon bullets should be obtained with further exploration and more training, however the agent started sticking on the top of the field after the forth iteration. We were not able to suppress this behavior and unfortunately, the agent continued to stick to the top even with any combinations of the hyper parameters.

V. Limitations and Discussions

The results indicate that a permutation-invariant deep reinforcement learning method is effective for environments that have permutation-invariant data structures. Compared to DQN methods that use classic deep networks, our method can learn faster with less computational force. However, the results show that further improvements for using A2C in the Touhou gym environment can be made. Moreover the stability of the agent and the gym environment leaves a room for further exploration. Especially, the issue that the agent has a possibility to stick to the top of the screen is crucial.

Additionally, our contributions to implementing the gym environment must open up a high potential for research in danmaku games. This gym environment provides a way to obtain the inner-stored values of the game from python easily. This allows easier researches and testing of reinforcement learning algorithms. Moreover, compatibility with the OpenAI gym environment enables testing the environment with codes implemented for other environments too.

References

[1] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[3] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.

[4] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *arXiv preprint arXiv:1706.02413*, 2017.

[5] ZUN, "弾幕開花宣言　東方花映塚　～ Phantasmagoria of Flower View." https://www16.big.or.jp/~zun/html/th09top.html, 2005.

[6] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," 2016.

[7] J. Shewmaker, "Analyzing dll injection," *GSM Presentation*, 2006.

[8] Y. Fujita, P. Nagarajan, T. Kataoka, and T. Ishikawa, "Chainerrl: A deep reinforcement learning library," *Journal of Machine Learning Research*, vol. 22, no. 77, pp. 1–14, 2021. [Online]. Available: http://jmlr.org/papers/v22/20-376.html