Pos2Pos: Automatic Position-to-Position Translation in Chess-Like Games

Shanchuan Wan[1,a)] Tomoyuki Kaneko[2,b)]

Abstract: Chess-like games share much common knowledge that is useful for both human players and computer programs, but is difficult to be well extracted and represented by using existing learning methods. This paper presents a novel semi-supervised learning method for automatically translating positions in one game to equivalent positions in another game. Preliminary experimental results in chess and shogi demonstrated the effectiveness of the proposed method.

1. Introduction

It is considered that chess-like games including *chess*, *shogi* and *xiangqi* share plenty of common features and tactics. According to this commonality, human players are supposed to be able to improve their skills and develop new playing patterns by learning from a wider range of game records. Also, the joint learning of evaluation functions for multiple games is expected to help achieve higher prediction accuracy in each game.

It is challenging to accurately extract and efficiently utilize common knowledge in different games. Generally, there are two feasible approaches: (1) to separate game features into common and domain-dependent parts; (2) to translate positions and moves from one game to another, and then incorporate the translated data into the original dataset of the target game. In this paper, we mainly focused on the second approach.

Cycle-consistent adversarial networks (CycleGANs) [1] are proposed by J. Zhu et al. based on the framework of pix2pix [2] and the adversarial learning method of original generative adversarial networks (GANs) [3] to solve unpaired image-to-image translation tasks. As proposed in AlphaGo series [4], [5], [6], a move in board games can be represented as a pair of successive positions, in which a position can be represented as a multi-channel image and processed by using convolutional neural networks (CNNs). Thus, we consider that learning methods for image-to-image translation are also able to be applied in translating game positions.

Based on CycleGANs [1] and other related work, we proposed a novel adversarial learning method named *pos2pos*, to learn translation functions $T(x)$ in chess-like games, that can automatically find a corresponding position $P_{\mathrm{T}} = T(P_{\mathrm{S}})$ in the target game G_{T} for a given position P_{S} in the source game G_{S} ($G_{\mathrm{S}} \neq G_{\mathrm{T}}$). Specifically, for finding corresponding positions in different games, the translation results shall satisfy the following two conditions: (1) P_{T} is a legal position of G_{T} or at least meaningful enough to be explained; (2) P_{S} and P_{T} are of approximately equivalent advantages for winning a game. It should be noted that the second condition is not required in image translation tasks, which makes the translation of game positions more unique and interesting.

We conducted experiments in chess and shogi, and preliminarily demonstrated the feasibility and effectiveness of the proposed method.

2. Related Work

Generative adversarial networks (GANs) [3] were first proposed by I. Goodfellow et al. in 2014, and has been applied in recent years in solving a lot of novel tasks including content generation, style transfer, super resolution and object detection.

Typically, a GAN consists of a generator network for producing textual, vocal or visual samples, and a discriminator network for judging if a given sample is a real sample or automatically generated artifact. The generator accepts a noise signal or a base sample as the input, and is trained to produce high-quality samples to fake the discriminator. The discriminator is trained to differentiate input samples by using real samples from the training dataset and fake samples output by the generator. Both networks have their independent parameters which are trained simultaneously and adversarially. In this way, the generator is able to acquire necessary gradients from the discriminator's judge results, to polish the representation of generated samples.

In the paper of least squares GANs (LSGANs) [7], X. Mao et al. proposed applying mean squared error (MSE) to the loss function of the discriminator to provide smooth and non-saturating gradients. Furthermore, improved Wasserstein GANs with gradient penalty (WGANs with GP) [8] are proposed by I. Gulrajani et al. based on M. Arjovsky

[1] GSII, The University of Tokyo
[2] III, The University of Tokyo and JST, PRESTO
[a)] swan@graco.c.u-tokyo.ac.jp
[b)] kaneko@graco.c.u-tokyo.ac.jp

et al.'s previous work [9]. In their work, a gradient penalty loss is adopted to make the discriminator's outputs conform to the requirements of the Lipschitz-1 continuous function. Both the two methods have been experimentally proven to be effective in improving the quality of generated results.

As an important application of GANs in domain transfer tasks, the design of CycleGANs [1] is capable of transferring images from one domain to another. Following the common practice for dual learning in neuro-linguistic programming, in training, CycleGANs transfer an image to a different domain, then transfers it back to the original domain and compares it with the original image. Additionally, CycleGANs introduce two discriminators from GANs to verify and enhance the transferred results in each domain.

In this paper, we incorporated the advantages of CycleGANs, LSGANs and WGANs with GP, and introduced game-specific discriminators for the position translation between chess and shogi.

3. Proposed Method

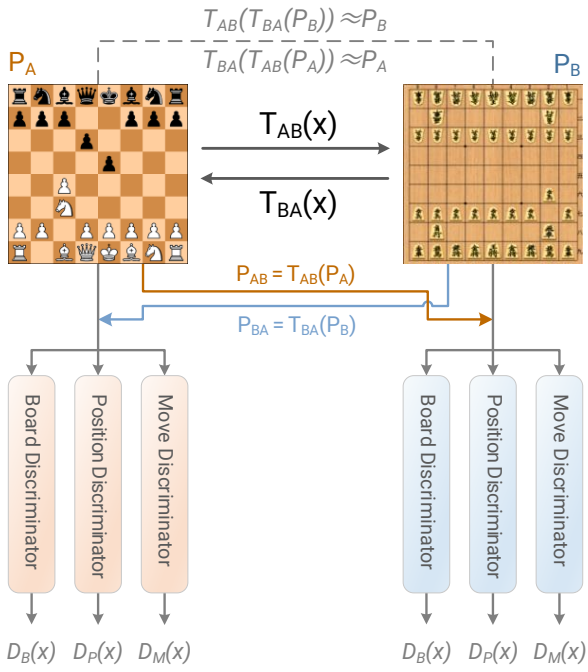


Fig. 1 Learning Framework for Position-to-Position Translation

The learning framework of the proposed *Pos2Pos* is demonstrated in Figure 1. The framework contains 1 translator, 3 discriminators for each game. All the translators and discriminators are based on convolutional neural networks.

3.1 Learning Objectives

As the core outcome of our method, a pair of inverse translators $\mathrm{T_{AB}}$ and $\mathrm{T_{BA}}$ is adopted for translating positions between *chess* and *shogi*.

For each game, we defined 3 discriminators for ensuring the quality of translated results.

- **Board discriminator** (BD) is for judging the legacy of a given game board. It takes a game position as input, and outputs a probability in $[0, 1]$ that indicates whether the input position could be a real state in the corresponding game.
- **Position discriminator** (PD) is for evaluating the scores of two related positions. Technically, the two positions are two positions subsequent to the same previous game state. The position discriminator takes the two positions as input, and determines that compared with the second input position, if the first position is chosen, which player is more likely to win.
- **Move discriminator** (MD) is for verifying the legacy of a given move. It takes a pair of successive positions as input, and outputs a probability in $[0, 1]$ that indicates whether the two positions could form a real move in the corresponding game.

In the original design of CycleGANs [1], there is only one discriminator responsible for ensuring the quality of generated contents, like the board discriminator we defined above. The position and move discriminators are newly designed and added in this paper, to ensure translated positions are relatively consistent and comparable to each other.

3.1.1 Translators

For two related games G_A and G_B, $\mathrm{T_{AB}}(x)$ and $\mathrm{T_{BA}}(x)$ are inverse translation functions to each other, and their main learning objectives are

$$\begin{aligned} P_\mathrm{ABA} &\approx \mathrm{T_{BA}}(\mathrm{T_{AB}}(P_\mathrm{A})),\ P_\mathrm{ABA} \approx P_\mathrm{A}, \\ P_\mathrm{BAB} &\approx \mathrm{T_{AB}}(\mathrm{T_{BA}}(P_\mathrm{B})),\ P_\mathrm{BAB} \approx P_\mathrm{B}, \end{aligned} \tag{1}$$

where $x \approx y$ is defined in this paper as the simplified expression of using mean squared error (MSE) as a loss function to minimize the difference between x and y in training. P_ABA and P_BAB are two generated positions after applying the two translation functions successively.

3.1.2 Board Discriminators

Two board discriminator networks $\mathrm{BD_A}(x)$ and $\mathrm{BD_B}(x)$ are adopted for judging if a given input x is a legal position in G_A or G_B, and trained to adjust the translation results in an adversarial way.

The parameters in $\mathrm{BD_A}$ and $\mathrm{BD_B}$ are trained by using

$$\begin{aligned} \mathrm{BD_A}(P_\mathrm{A}) &\approx 1,\ \ \mathrm{BD_A}(P_\mathrm{BA}) \approx 0, \\ \mathrm{BD_B}(P_\mathrm{B}) &\approx 1,\ \ \mathrm{BD_B}(P_\mathrm{AB}) \approx 0, \end{aligned} \tag{2}$$

while the parameters in the two translators $\mathrm{T_{AB}}(x)$ and $\mathrm{T_{BA}}(x)$ are trained to enhance the board discriminators' outputs for the translated positions:

$$\mathrm{BD_A}(P_\mathrm{BA}) \approx 1,\ \ \mathrm{BD_B}(P_\mathrm{AB}) \approx 1, \tag{3}$$

where $P_\mathrm{AB} = \mathrm{T_{AB}}(P_\mathrm{A})$ and $P_\mathrm{BA} = \mathrm{T_{BA}}(P_\mathrm{B})$ are two positions translated from G_A to G_B and from G_B to G_A, respectively.

3.1.3 Position Discriminators

For each game, a position discriminator $\mathrm{PD}(x)$ is newly

added for evaluating the relative advantages between two related positions. Position discriminators are trained to answer which player is more likely to win, in case that the first position is chosen.

Specifically, parameters in $\mathrm{PD_A}$ and $\mathrm{PD_B}$ are trained as

$$\begin{aligned} \mathrm{PD_A}(\langle Y_\mathrm{A}, R_\mathrm{A}\rangle) \approx p_\mathrm{A}, \quad \mathrm{PD_A}(\langle R_\mathrm{A}, Y_\mathrm{A}\rangle) \approx -p_\mathrm{A}, \\ \mathrm{PD_B}(\langle Y_\mathrm{B}, R_\mathrm{B}\rangle) \approx p_\mathrm{B}, \quad \mathrm{PD_B}(\langle R_\mathrm{B}, Y_\mathrm{B}\rangle) \approx -p_\mathrm{B}, \end{aligned} \tag{4}$$

where X, Y, R denote 3 kinds of different positions in a game. X is a non-terminal game position. Y is a position subsequent to X, and moved by an expert player or program. R is a random position subsequent to X other than Y. $p = \pm 1$ indicates the next player to move.

As for positions $X_\mathrm{AB}, Y_\mathrm{AB}, R_\mathrm{AB}$ translated from G_A to G_B, and positions $X_\mathrm{BA}, Y_\mathrm{BA}, R_\mathrm{BA}$ translated from G_B to G_A, the position discriminators are trained to give results approximate to 0 for them:

$$\begin{aligned} \mathrm{PD_A}(\langle Y_\mathrm{BA}, R_\mathrm{BA}\rangle) \approx 0, \quad \mathrm{PD_A}(\langle R_\mathrm{BA}, Y_\mathrm{BA}\rangle) \approx 0, \\ \mathrm{PD_B}(\langle Y_\mathrm{AB}, R_\mathrm{AB}\rangle) \approx 0, \quad \mathrm{PD_B}(\langle R_\mathrm{AB}, Y_\mathrm{AB}\rangle) \approx 0, \end{aligned} \tag{5}$$

while the parameters in two translators $\mathrm{T_{AB}}(x)$ and $\mathrm{T_{BA}}(x)$ are trained to make the position discriminators output reasonable results, as

$$\begin{aligned} \mathrm{PD_A}(\langle Y_\mathrm{BA}, R_\mathrm{BA}\rangle) \approx p_\mathrm{B}, \quad \mathrm{PD_A}(\langle R_\mathrm{BA}, Y_\mathrm{BA}\rangle) \approx -p_\mathrm{B}, \\ \mathrm{PD_B}(\langle Y_\mathrm{AB}, R_\mathrm{AB}\rangle) \approx p_\mathrm{A}, \quad \mathrm{PD_B}(\langle R_\mathrm{AB}, Y_\mathrm{AB}\rangle) \approx -p_\mathrm{A}, \end{aligned} \tag{6}$$

3.1.4 Move Discriminators

Along with position discriminators, a move discriminator is added to each game for judging the legacy of given moves.

The parameters in $\mathrm{MD_A}(x)$ and $\mathrm{MD_B}(x)$ are trained to recognize real moves in the training data set as 1 by using

$$\begin{aligned} \mathrm{MD_A}(\langle X_\mathrm{A}, Y_\mathrm{A}\rangle) \approx 1, \quad \mathrm{MD_A}(\langle X_\mathrm{A}, N_\mathrm{A}\rangle) \approx 1, \\ \mathrm{MD_B}(\langle X_\mathrm{B}, Y_\mathrm{B}\rangle) \approx 1, \quad \mathrm{MD_B}(\langle X_\mathrm{B}, N_\mathrm{B}\rangle) \approx 1, \end{aligned} \tag{7}$$

and are trained to recognize translated moves as 0 by using

$$\begin{aligned} \mathrm{MD_A}(\langle X_\mathrm{BA}, Y_\mathrm{BA}\rangle) \approx 0, \quad \mathrm{MD_A}(\langle X_\mathrm{BA}, N_\mathrm{BA}\rangle) \approx 0, \\ \mathrm{MD_B}(\langle X_\mathrm{AB}, Y_\mathrm{AB}\rangle) \approx 0, \quad \mathrm{MD_B}(\langle X_\mathrm{AB}, N_\mathrm{AB}\rangle) \approx 0, \end{aligned} \tag{8}$$

while the parameters in two translators $\mathrm{T_{AB}}(x)$ and $\mathrm{T_{BA}}(x)$ are trained to fake the move discriminators' results as

$$\begin{aligned} \mathrm{MD_A}(\langle X_\mathrm{BA}, Y_\mathrm{BA}\rangle) \approx 1, \quad \mathrm{MD_A}(\langle X_\mathrm{BA}, N_\mathrm{BA}\rangle) \approx 1, \\ \mathrm{MD_B}(\langle X_\mathrm{AB}, Y_\mathrm{AB}\rangle) \approx 1, \quad \mathrm{MD_B}(\langle X_\mathrm{AB}, N_\mathrm{AB}\rangle) \approx 1. \end{aligned} \tag{9}$$

3.2 Network Architectures

The network architectures of translators and discriminators are listed in Table 1 and Table 2, containing 38 and 18 convolutional layers, and forming 17 and 9 standard residual blocks [10], respectively. All hidden layers are activated by a leaky rectified linear unit (Leaky ReLU) [11] function with $\alpha = 0.2$. No activation function is applied to any output layer in any network.

The input sizes of *chess* and *shogi* data are $[12, 8, 8]$ and $[28, 9, 9]$, respectively. The first dimensions are encoded according to piece types in the two games. For convenience, we temporarily excluded the information about prisoner pieces in *shogi*.

Table 1 Network Architectures of Translators

Phases	*Layers*	*Filters*	*Kernels*	*Strides*
Input	–	12, 28	–	–
Encoding (Conv)	6	128	3	1
	1	192	3	2
	5	192	3	1
	1	256	3	2
	5	256	3	1
	1	512	3	2, 3
	1	512	3	1
Decoding (Conv)	1	256	4, 3	0.25, 0.33
	1	256	4, 3	1
	4	256	3	1
	6	192	3	1
	1	128	2, 3	0.5, 0.33
	1	128	2, 3	1
	4	128	3	1
Output	1	28, 12	1	1

Table 2 Network Architectures of Discriminators

Phases	*Layers*	*Filters*	*Kernels*	*Strides*
Input	–	12, 28	–	–
Conv	6	128	3	1
	1	128	3	2
	5	128	3	1
	1	128	3	2
	5	128	3	1
AvgPool	1	128	2, 3	1
Output	1	256	–	–
	1	1	–	–

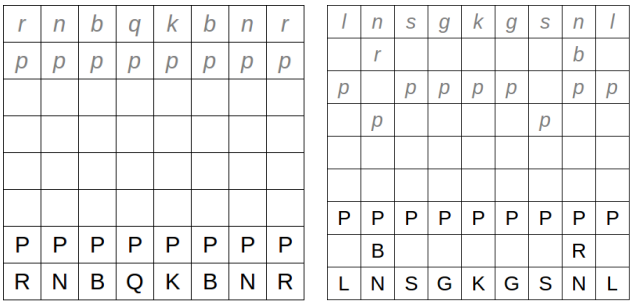
The design of translator networks in this paper is based on auto encoders, in which input images in the source game are first encoded into a 512-dimensional feature vector FV, and then decoded as corresponding images in the target game through deconvolution. In training, we added an extra MSE-based penalty function that punishes drastic changes in the encoded feature vectors FV when a position is translated into another game. This penalty function was proved to be capable of accelerating the training of translators in experiments.

3.3 Training Configurations

We collected 783,129 games from computer chess database CCRL 40/40 [*1] and 868,161 games from computer shogi server Floodgate [*2]. 85% of the data were used for training, and the remaining 15% were for testing. Only moves made by the winner of a game were sampled during training.

In experiments, the batch size was set as 16 instances, initial learning rate started from 0.00005 and decayed to 0.992 of the original rate in every $10,000$ steps. All loss functions were minimized by using the Adam optimization algorithm (beta1 = 0.9). Discriminator and translator networks were

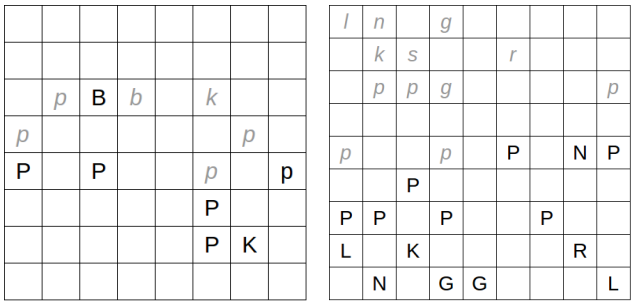
[*1] `http://www.computerchess.org.uk/ccrl/4040`
[*2] `http://wdoor.c.u-tokyo.ac.jp/shogi/index-e.html`



(a) Source position in chess (b) Translated position in shogi

(c) Source position in shogi (d) Translated position in chess

Fig. 2 Translation of Opening Game Positions with *pos2pos*



(a) Source position in chess (b) Translated position in shogi

(c) Source position in shogi (d) Translated position in chess

Fig. 3 Translation of End Game Positions with *pos2pos*

simultaneously trained, however, the discriminators' parameters were set as unchangeable while training the translator networks.

4. Results

Selected translated results are demonstrated in Figure 2 and Figure 3. We put a piece on the board if and only if its corresponding value on the translators' output images can be rounded to 1. The MSE between the real opening positions and the translated results shown in Figure 2 were 0.0029 for *chess* and 0.0166 for *shogi*.

It can be found from the figures that the proposed translators performed well with opening game positions, and was able to understand complicated middle and end game positions and represent them in another game.

The experimental results in this paper demonstrated that, the translation functions trained by using the proposed *Pos2Pos* method are effective in translating and reconstructing given game positions. As a part of our future work, we are scheduled to further improve and quantitatively analyze the translation of a single positions and a sequence of moves in chess-like games.

Acknowledgement

A part of this work was supported by JSPS KAKENHI Grant Number 16H02927 and by JST, PRESTO.

References

[1] Zhu, J.-Y., Park, T., Isola, P. and Efros, A. A.: Unpaired image-to-image translation using cycle-consistent adversarial networks, *IEEE ICCV* (2017).

[2] Isola, P., Zhu, J.-Y., Zhou, T. and Efros, A. A.: Image-to-image translation with conditional adversarial networks, *IEEE CVPR* (2017).

[3] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y.: Generative adversarial nets, *Advances in neural information processing systems*, pp. 2672–2680 (2014).

[4] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M. et al.: Mastering the game of Go with deep neural networks and tree search, *nature*, Vol. 529, No. 7587, pp. 484–489 (2016).

[5] Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A. et al.: Mastering the game of go without human knowledge, *Nature*, Vol. 550, No. 7676, p. 354 (2017).

[6] Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T. et al.: Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm, *arXiv preprint arXiv:1712.01815* (2017).

[7] Mao, X., Li, Q., Xie, H., Lau, R. Y., Wang, Z. and Smolley, S. P.: Least squares generative adversarial networks, *2017 IEEE International Conference on Computer Vision (ICCV)*, IEEE, pp. 2813–2821 (2017).

[8] Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V. and Courville, A. C.: Improved Training of Wasserstein GANs, *Advances in Neural Information Processing Systems 30* (Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S. and Garnett, R., eds.), Curran Associates, Inc., pp. 5767–5777 (online), available from ⟨http://papers.nips.cc/paper/7159-improved-training-of-wasserstein-gans.pdf⟩ (2017).

[9] Arjovsky, M., Chintala, S. and Bottou, L.: Wasserstein generative adversarial networks, *International Conference on Machine Learning*, pp. 214–223 (2017).

[10] He, K., Zhang, X., Ren, S. and Sun, J.: Deep residual learning for image recognition, *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778 (2016).

[11] He, K., Zhang, X., Ren, S. and Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034 (2015).