# BAB 7

# KESIMPULAN DAN SARAN

## 7.1 Evaluasi Pencapaian Tujuan Penelitian

Skripsi ini telah berhasil mencapai tujuan yang diberikan pada Bagian 1.3. Berikut diberikan penegasan terhadap beberapa tujuan yang telah dicapai oleh skripsi ini:

- **Mencari cara yang dapat dipakai untuk mengimplementasikan *general game-playing agent* yang menggunakan *neural network* untuk mencari *action* terbaik pada suatu *state* di gim, agar dapat bermain pada beberapa gim di Ludii Player**
  Pada skripsi ini *general game-playing agent* yang dibuat adalah agen yang bisa dilatih untuk bermain pada suatu gim menggunakan *self-play*. Untuk melakukan *self-play*, agen yang menggabungkan algoritma *MCTS* dan *neural network* seperti yang dibahas pada Bagian 2.5.2. Dari hasil eksperimen yang dilakukan pada Bab 6, dibuktikan bahwa agen yang dibuat pada skripsi ini memiliki kemampuan untuk setidaknya secara rasional mencari *action* terbaik pada lebih dari satu gim. Ditambah lagi, agen ini memenuhi sifat general yang definisi lebih lengkapnya terdapat pada Bagian 3.1.2.
  Berdasarkan tujuan agen yang dibuat harus dapat dijalankan di Ludii Player. Untuk melakukan itu *library Deep Learning* DJL dan *general game-playing system* Ludii perlu diintegrasikan sedemikian rupa. Pada Bagian 5.1 dijelaskan dua penyesuaian yang diperlukan untuk menggabungkan *MCTS* dan *neural network*. Penyesuaian pertama adalah konversi objek yang dimiliki Ludii menjadi representasi *tensor* menggunakan *wrapper* yang dirumuskan pada bagian 2.6.5. Kedua, dilakukan perubahan implementasi dari *library* agen dan *system property* dari komputer agar agen dapat dimainkan di Ludii Player. Kedua perubahan ini menyebabkan pada skripsi ini dibuat empat sistem berbeda untuk pengujian dan pelatihan agen.
- **Merancang model *neural network* yang dapat digunakan *game-playing agent* untuk mencari strategi bermain, sesuai dengan gim yang dipilih untuk dimainkan oleh agen tersebut di Ludii.**
  Pada dasarnya, di skripsi ini bentuk dari *neural network* disesuaikan dengan fitur dari gim yang telah dikonversi menjadi *tensor*. Hal ini dilakukan karena seperti yang dituliskan di Bagian 3.1.2 *neural network* memiliki bentuk yang *fix*, sehingga perlu suatu cara untuk menangani *fitur* gim yang jumlahnya berbeda-beda. Untuk mengatasi jumlah fitur yang berbeda-beda tersebut, skripsi ini menggunakan cara yang ada pada Bagian 3.2. Cara ini membuat model *neural network* dirancang mengikuti ukuran dari *wrapper* atau representasi fitur gim dalam bentuk *tensor*. Untuk memperjelas, pada Bagian 6.4.1 dan Bagian 6.4.2 ditunjukkan cara pembangunan model *neural network* yang dipakai oleh eksperimen.
  Sederhananya, ketika mencari strategi bermain *neural network* memiliki dua peran yaitu membantu *MCTS/PUCT* memprediksi *reward* dan probabilitas *action* pada *state* tertentu. Prediksi tersebut dapat dilakukan karena terdapat dua *output head* pada *neural network* yaitu *policy head* dan *value head*. Seperti yang dijelaskan pada Bagian 3.2.3 *policy head* berguna untuk melakukan prediksi terhadap probabilitas *action*, sedangkan *value head* berguna untuk melakukan prediksi terhadap *reward*. Pada Bagian 3.2.3 juga dijelaskan bahwa kedua *output head* memiliki *loss function* dan *activation* yang berbeda.

- **Melakukan pelatihan pada *neural network* secara *online* dengan mendapatkan data dari *MCTS*. Kemudian memperbaharui parammeter *neural network* dengan cara menandingkan agen dengan cara *self-play*.**
  Agen yang dibuat di skripsi ini dilatih dengan cara *self-play*. Secara keseluruhan algoritma *self-play* adalah algoritma *reinforcement learning* yang bersifat *online learning*. Pada setiap iterasi pelatihan menggunakan algoritma *self-play* banyak manipulasi data dilakukan (terutama untuk menghasilkan data pelatihan) menggunakan algoritma *MCTS/PUCT*. Data yang dihasilkan tersebut digunakan untuk melatih *neural network* sehingga memiliki nilai parameter baru yang dianggap meningkatkan kecerdasan dari agen.
  Apabila dijelaskan secara sederhana sebenarnya algoritma *self-play* adalah algoritma yang melatih agen dengan melakukan pelatihan dengan membuat beberapa pertandingan antar agen yang menggunakan algoritma MCTS/PUCT. Pada skripsi ini definisi dari pelatihan dengan self-play telah dijelaskan pada Bagian 2.5.4. Kemudian, dari sisi perancangan di Bab 4 tahap *self-play* ke dibagi empat bagian yaitu inisiasi, pelatihan, pembuatan data, pelatihan *neural network*, dan evaluasi. Pelatihan ini dilakukan untuk mencari *parameter* yang dianggap optimal untuk digunakan pada agen. Untuk memenuhi hal tersebut, di Bagian 5.2.3 juga ditunjukkan bahwa terdapat mekanisme untuk yang menyimpan dan menyemat *file* parameter. Penyimpanan *file parameter* di *file system* dilakukan pada setiap iterasi. Sehingga, *file* parameter yang merupakan hasil *output* dari pelatihan dapat digunakan digunakan kembali untuk dilatih lebih lanjut atau digunakan oleh agen untuk bermain suatu gim.

## 7.2 Jawaban Masalah

Pada Bagian 1.2 telah dirumuskan beberapa masalah yang ingin diselesaikan oleh skripsi ini. Apabila ditelusuri sebenarnya masalah utama yang diselesaikan adalah membuat *general game-playing agent* yang menggunakan *neural network* untuk memperkirakan *action* terbaik agar dapat digunakan untuk bermain beberapa gim di Ludii Player. Terlepas dari performa yang ditunjukkan pada Bab 6, dengan hanya satu menggunakan buah rancangan dan tanpa bantuan berupa kemampuan manusia, agen berhasil digunakan untuk bermain pada lebih dari satu gim di Ludii Player. Pada Bagian 3.1.2 dijelaskan bahwa agen yang memenuhi sifat general adalah agen yang dapat bermain lebih dari satu gim dengan satu metode tak berubah dan tidak butuh bantuan dalam bentuk kemampuan manusia dalam domain gim. Berdasarkan pernyataan tersebut dapat disimpulkan bahwa agen yang dibuat memenuhi sifat *general* dan dapat disebut sebagai *general game-playing agent*.

Sifat general tentunya memiliki *trade off* yaitu semakin general agen yang ingin dibuat semakin banyak pengaturan pada rancangan yang harus dibuat, terutama dari sisi integrasi terhadap sebuah lingkungan. Bahkan, perlu digarisbawahi bahwa masih diperlukan beberapa modifikasi terhadap agen agar dapat berhasil dijalankan di Ludii Player yang dijelaskan di Bagian 5.1. Modifikasi tersebut juga dilakukan supaya *library Deep Learning* DJL dapat berintegrasi dengan *general game-playing system*. Selain itu, modifikasi yang dibuat juga membantu menghilangkan kesalahan, khususnya pada *wrapper* yang disediakan di Ludii 1.2.6. Walau didapatkan setelah dimodifikasi agen kehilangan beberapa kemampuan, agen tetap dapat disebut sebagai *general game-playing agent* karena agen tetap dirancang menggunakan satu rancangan yang sama.

Pernyataan pada Bagian 2.5.2 menujukan bahwa dengan bantuan dari *neural network*, agen dengan *MCTS* dapat digunakan untuk memperkirakan *action* terbaik. *MCTS* ini bersifat *aheuristic*, sehingga secara otomatis dapat digunakan untuk membangun agen *general*. Berbeda dengan *neural network* yang harus diatur sedemikian rupa agar bisa menjadi *general*. Pada Bagian 2.6.5, ditunjukkan bahwa *wrapper* membuat *input* dan *output tensor* yang selalu berukuran tetap sehingga memungkinkan agen untuk dimainkan pada lebih dari satu gim dengan memanipulasi *input* dan *output layer* yang dimiliki *neural network*. *Wrapper* dibuat berdasarkan fitur gim yang didapatkan melalui objek `Game` di Ludii. Sehingga dapat disimpulkan bahwa faktor general yang dimiliki agen bergantung terhadap seberapa banyak fitur gim yang dapat dimodelkan oleh *wrapper*.

Seperti yang dituliskan pada Bagian 1.2, selain mencari cara membuat agen yang *general* pada skripsi ini juga ingin dicari model *neural network* yang baik. Pada Bagian 1.2 juga diperjelas bahwa model *neural network* baik yang dimaksud skripsi ini adalah model yang dapat dipakai oleh *game-playing agent* untuk bermain beberapa permainan pada Ludii. Sehingga apabila diperjelas, maksud daripada skripsi ini baik yang dimaksud bukanlah diukur dari segi performa, melainkan semua data yang diberikan oleh *tensor* kepada *neural network* harus dapat diproses secara utuh supaya agen dapat bermain secara rasional di Ludii. Pada *fully connected neural network* hal ini mudah dicapai karena satu seluruh *output* dari *neuron* pada satu *layer neuron* pasti terhubung dengan seluruh *input* dari *neuron* pada *layer* berikutnya. Namun, pada *convolutional neural network* bentuk *kernel* harus diatur agar sesuai dengan bentuk data yang diberikan ke *layer* tersebut. Aturan pembentukan *convolutional neural network* pada Bagian 3.2 telah berhasil merumuskan cara yang dapat digunakan untuk membuat model yang dapat menampung data secara utuh.

Untuk membangun model *neural network* yang melakukan proses data secara utuh, sebenarnya yang menjadi penting adalah *input layer* memiliki kemampuan untuk menampung semua *element* pada *state tensor*. Sedangkan, setiap *element move tensor* dan *reward* harus bisa ditampung dan dipetakan pada *output layer*. Ketika melakukan prediksi data yang diberikan *move tensor* dan *reward* juga harus dipisahkan sehingga pada skripsi ini kedua data tersebut dipisahkan ke dua *output head*. Untuk mengetahui secara lebih mendetail, Pada Bagian 6.4.1 dan Bagian 6.4.2 telah dijelaskan secara spesifik cara yang digunakan untuk membangun dua model pada eksperimen.

Pada skripsi ini, pelatihan dengan *self-play* menjadi cara dari agen untuk meningkatkan performa. Seperti yang dituliskan pada Bagian 3.4 *self-play* adalah penelitian yang memerlukan *resources hardware* yang besar sehingga disadari bahwa masih dibutuhkan *resources hardware* yang lebih besar agar eksplorasi pada performa agen terutama ketika digunakan untuk bermain pada gim dengan ukuran papan yang "besar" dapat ditingkatkan. Namun, faktor performa ini tidak menjadi fokus dari masalah yang ingin diselesaikan, karena dapat dianalisis bahwa walaupun agen tidak memiliki performa yang baik, agen yang dihasilkan pada eksperimen masih dapat memilih *action* yang dianggapnya terbaik secara rasional. Selain itu, setidaknya skripsi ini berhasil membuktikan bahwa agen yang general dapat dibangun di atas *general game-playing system* Ludii walau menggunakan *resources hardware* yang jauh lebih sedikit dari penelitian serupa dengan performa yang lebih baik.

Di sisi lain, eksperimen pada Tic-Tac-Toe dan Hex $4 \times 4$ pada Bagian 6.5.1 dan Bagian 6.5.2 menjawab masalah melakukan pelatihan pada agen agar agen dapat memperkirakan *action* dengan optimal pada suatu state gim. Kedua gim ini telah berhasil mendapatkan langkah permainan yang dapat menyamai kemampuan agen sempurna yaitu *alpha-beta pruning* dengan hanya 50 iterasi *MCTS* dibantu dengan *neural network*. Hal ini membuktikan bahwa dengan *neural-network* yang tepat perkiraan terhadap *action* optimal pada suatu state gim bisa didapatkan. Walaupun karena terbatasnya *resource hardware* belum dilakukan eksplorasi pelatihan lebih detail terhadap gim dengan ukuran papan yang lebih besar sehingga walaupun sudah mengindikasikan adanya rasionalitas, hasil yang ditunjukkan belum menujukan performa yang tinggi.

## 7.3 Saran Untuk Peneletian Berikutnya

Berdasarkan data yang didapatkan dari eksperimen, didapatkan bahwa agen yang dibuat masih dapat dikembangkan dari sisi performa. Berikut beberapa usulan yang dapat dijadikan panduan untuk penelitian berikutnya:

- **Melakukan terlebih dahulu pelatihan *neural network* tanpa *MCTS***
  Pelatihan menggunakan *self-play* yang menggabungkan *MCTS* dan *neural network* memiliki *hyperparameter* yang banyak. Banyaknya *hyperparameter* tersebut membuat *hyperparameter tuning* lebih sulit dilakukan. Selain itu *MCTS* memiliki pengaruh besar terhadap *state* gim pada saat penelitian sehingga terdapat kemungkinan banyak *state* yang tidak terjamah saat pelatihan. Oleh karena itu, skripsi ini mengusulkan penelitian berikutnya untuk terlebih dahulu mencoba menggunakan *neural network* yang dilatih menenggunakan pemilihan *action*

secara *random* tanpa bantuan *MCTS* atau melatih *neural network* membandingkannya dengan *data set*. Dengan cara ini *hyperparameter* lebih sedikit dan *neural network* dapat dilatih tanpa pengaruh faktor eksplorasi dan eksploitasi dari *MCTS* dan evaluasi terhadap hasil juga dapat secara lebih mudah dilakukan.

- **Memisahkan prediksi *policy* dan *value* ke dalam dua *neural network* berbeda**
  Eksperimen mengindikasikan terdapat kondisi ketika prediksi terhadap *policy* sudah lebih dahulu mendekati *convergence* ketimbang prediksi terhadap *value*. Kondisi ini terjadi karena *policy head* dan *value head* digabungkan ke dalam satu *neural network*. Dengan memisahkan kedua *head* tersebut menjadi *dua neural network* prediksi terhadap *policy* dan *value* dapat dipisahkan secara eksplisit. Dengan perubahan ini *evaluasi* yang dilakukan pada *neural network* menjadi lebih sederhana sehingga dapat dilakukan lebih banyak eksplorasi terhadap *neural network* optimal.
- **Mencari *neural network* yang lebih optimal**
  Pada skripsi ini karena lebih berfokus ke pengembangan agen, eksplorasi terhadap arsitektur dari *neural network* untuk mendapatkan performa *state of the art* pada gim dengan ukuran papan besar. Disadari bahwa salah satu penyebab terbesar kondisi tersebut terjadi adalah rancangan *neural network* belum dirancang secara optimal. Terdapat beberapa hal yang belum dicoba di skripsi ini yang dapat menambahkan performa dari neural network diantara lain penggunaan *skip conection* dan penggunaan *regularization*. Dengan melakukan hal tersebut mungkin dapat didapatkan keseimbangan yang optimal antara *bias* dan *variance* pada pelatihan.
- **Menggunakan arsitektur yang umum digunakan pada penelitian self-play**
  Skripsi ini menggunakan *convolution layer* yang berdimensi tiga pada *convolutional neural network* dan *fully connected neural network* yang bentuknya diatur oleh fitur melalui *wrapper* dari gim. Arsitektur dan cara penghasilan *neural network* seperti ini tidak umum digunakan pada penelitian *self-play* serupa. Mungkin di penelitian selanjutnya dapat dicoba arsitektur *resnet* mengikuti penelitian *state of arts* yaitu AlphaZero. Apabila *neural network* tetap ingin dibentuk sesuai ukuran fitur gim manipulasi terhadap jumlah *residual layer* dapat dilakukan.
- **Membuat sistem *self-play* yang mendukung multiprocessing**
  Ditemukan bahwa pada saat pelatihan *MCTS* memakan waktu yang cukup lama dijalankan. Sedangkan, *Hardware* yang berkontribusi terhadap performa kecepatan komputasi dari *MCTS* adalah CPU. Untuk itu, supaya pelatihan dapat dilakukan dengan lebih cepat dapat jumlah *core* dari CPU dapat ditambahkan. Namun, untuk melakukan hal tersebut secara optimal perlu ditambahkan kemampuan *multiprocessing* pada agen.
- **Menambahkan modifikasi berupa simetri pada *tensor* dan menggunakan $c_{puct}$ dengan nilai dinamis**
  Apabila dibandingkan AlphaZero masih terdapat beberapa penyerhanaan yang dilakukan ketika pelatihan. Penyederhanaan tersebut diantara lain tidak dilakukannya pemrosesan simetri berupa refleksi dan rotasi pada papan gim menggunakan wrapper dan nilai $c_{puct}$ yang konstan. Dengan adanya pemrosesan simetri data diharapkan *neural network* bisa secara lebih cepat mendeteksi sifat *invariant* pada gim. Selain itu nilai $c_{puct}$ bisa diganti menjadi dinamis tergantung dari jumlah kunjungan pada satu *state*.
- **Melakukan penelitian *self-play* mengunakan *library-library* yang lebih stabil**
  Terdapat tiga *library* (Ludii, DJL, JSAT) yang digunakan pada skripsi ini dan dua di antaranya (Ludii, DJL) masih dalam pengembangan. Penelitian menggunakan kedua *library* tersebut masih belum banyak dilakukan sehingga hanya ada sedikit referensi untuk penelitian yang serupa. Bahkan terkadang *open source code* masih perlu dibuka untuk melihat detail dari dokumentasi. Stabilitas *library* juga masih belum dapat dipastikan karena terdeteksi adanya sedikit *memory leak* di Library DJL yang digunakan dan kesalahan pada *wrapper*. Oleh karena itu apabila mengincar segi performa agen, perlu dicari *library* yang sudah lebih stabil.

- **Mencari cara evaluasi yang lebih baik pada tahap pelatihan**
  Pada tahap evaluasi diambil atau tidaknya suatu *neural network* ditentukan dengan melakukan pertandingan agen menggunakan *MCTS* lalu diukur melalui *threshold* kemenangan tertentu. Di tahap ini pada gim yang kemampuan permainan *player* 1 dan *player* 2 tidak stabil, sering terjadi *neural network* batal diambil. Untuk mencegah hal ini mungkin diperlukan cara evaluasi yang lebih detil. Salah satu idenya adalah menghitung *action* yang digunakan pada satu permainan gim.

# DAFTAR REFERENSI


[1] Apt, K. R. dan Grädel, E. (ed.) (2011) *Lectures in Game Theory for Computer Scientists*. Cambridge University Press, Aachen, Germany.

[2] Shoham, Y. dan Leyton-Brown, K. (2008) *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press, USA.

[3] Browne, C. (2017) Back to the past: Ancient games as a new AI frontier. *AAAI Conference on Artificial Intelligence*, San Francisco, California, 4–9 Februari, AAAI Technical Report, **WS-17**, pp. 1025–1026. AAAI Press.

[4] Wooldridge, M. J. (2002) *Introduction to Multiagent Systems*. John Wiley & Sons, Inc., USA.

[5] Russell, S. dan Norvig, P. (2009) *Artificial Intelligence: A Modern Approach*, 3rd edition. Prentice Hall Press, USA.

[6] Turing, A. M. (1950) Computing Machinery and Intelligence. *Mind*, **LIX**, 433–460.

[7] Bychkov, D., Linder, N., Turkki, R., Nordling, S., Kovanen, P. E., Verrill, C., Walliander, M., Lundin, M., Haglund, C., dan Lundin, J. (2018) Deep learning based tissue analysis predicts outcome in colorectal cancer. *Scientific reports*, **8**, 1–11.

[8] Maqueda, A. I., Loquercio, A., Gallego, G., García, N., dan Scaramuzza, D. (2018) Event-based vision meets deep learning on steering prediction for self-driving cars. *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018*, Salt Lake City, USA, 18–22 Juni, pp. 5419–5427. IEEE Computer Society.

[9] Gudelek, M. U., Boluk, S. A., dan Ozbayoglu, A. M. (2017) A deep learning based stock trading model with 2-d CNN trend detection. *2017 IEEE Symposium Series on Computational Intelligence, SSCI 2017*, Honolulu, USA, 27 November–1 Desember, pp. 1–8. IEEE.

[10] Campbell, M., Hoane, A. J., dan Hsu, F.-h. (2002) Deep blue. *Artif. Intell.*, **134**, 57–83.

[11] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., dan Hassabis, D. (2016) Mastering the game of go with deep neural networks and tree search. *Nature*, **529**, 484–503.

[12] Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T. P., Simonyan, K., dan Hassabis, D. (2017) Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *CoRR*, **abs/1712.01815**.

[13] Piette, É., Soemers, D. J. N. J., Stephenson, M., Sironi, C. F., Winands, M. H. M., dan Browne, C. (2020) Ludii - the ludemic general game system. *ECAI 2020 - 24th European Conference on Artificial Intelligence*, Santiago de Compostela, Spain, 29 Agustus–8 September, pp. 411–418. IOS Press.

[14] Stephenson, M., Piette, É., Soemers, D. J. N. J., dan Browne, C. (2019) An overview of the ludii general game system. *IEEE Conference on Games, CoG 2019*, London, United Kingdom, 20–23 Juni, pp. 1–2. IEEE.

[15] Stephenson, M., Soemers, D. J. N. J., Piette, É., dan Browne, C. (2021) General game heuristic prediction based on ludeme descriptions. *CoRR*, **abs/2105.12846**.

[16] Genesereth, M., Love, N., dan Pell, B. (2005) General game playing: Overview of the aaai competition. *AI Magazine*, **26**, 62.

[17] Browne, C. (2016) A class grammar for general games. Bagian dari Plaat, A., Kosters, W., dan van den Herik, J. (ed.), *Computers and Games*, Cham, Desember, pp. 167–182. Springer International Publishing.

[18] Géron, A. (2017) *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*, 1st edition. O'Reilly Media, Inc., Sebastopol.

[19] Morales, M. (2020) *Grokking Deep Reinforcement Learning*. Manning Publications, New York.

[20] Swiechowski, M., Godlewski, K., Sawicki, B., dan Mandziuk, J. (2021) Monte carlo tree search: A review of recent modifications and applications. *CoRR*, **abs/2103.04931**.

[21] Sutton, R. S. dan Barto, A. G. (2018) *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge.

[22] Gunawan, A., Ruan, J., Thielscher, M., dan Narayanan, A. (2020) Exploring a learning architecture for general game playing. Bagian dari Gallagher, M., Moustafa, N., dan Lakshika, E. (ed.), *AI 2020: Advances in Artificial Intelligence*, Canberra, ACT, Australia, 29–30 November, Lecture Notes in Computer Science, **12576**, pp. 294–306. Springer.

[23] Soemers, D. J. N. J., Mella, V., Browne, C., dan Teytaud, O. (2021) Deep learning for general game playing with ludii and polygames. *CoRR*, **abs/2101.09562**.

[24] Simonyan, K. dan Zisserman, A. (2015) Very deep convolutional networks for large-scale image recognition. Bagian dari Bengio, Y. dan LeCun, Y. (ed.), *3rd International Conference on Learning Representations, ICLR 2015, , Conference Track Proceedings*, San Diego, CA, USA, 7-9 Mei, pp. 1–14. The International Conference on Learning Representations.

[25] Wolf, M. (2021) *Encyclopedia of Video Games: The Culture, Technology, and Art of Gaming*. ABC-CLIO, LLC, Santa Barbara, California.

[26] Thoms, W. J. (1858) *Notes and queries*. Oxford University Press, United Kingdom.

[27] Barron, E. N. (2013) *Game Theory: An Introduction*, 2nd edition. John Wiley & Sons, Ltd, Hoboken, New Jersey.

[28] Roth, A., Balcan, M. F., Kalai, A., dan Mansour, Y. (2010) On the equilibria of alternating move games. *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms*, USA, Oktober SODA '10 805–816. Society for Industrial and Applied Mathematics.

[29] Whitehouse, D. (2014) Monte Carlo Tree Search for games with Hidden Information and Uncertainty. Disertasi. University of York, Heslington, United Kingdom.

[30] Ravichandiran, S. (2018) *Hands-On Reinforcement Learning with Python: Master Reinforcement and Deep Reinforcement Learning Using OpenAI Gym and TensorFlow*. Packt Publishing, Birmingham.

[31] Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., Driessche, G., Graepel, T., dan Hassabis, D. (2017) Mastering the game of go without human knowledge. *Nature*, **550**, 354–359.

[32] Browne, C., Powley, E., Whitehouse, D., Lucas, S., Cowling, P., Rohlfshagen, P., Tavener, S., Perez Liebana, D., Samothrakis, S., dan Colton, S. (2012) A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, **4:1**, 1–43.

[33] Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K., dan Hassabis, D. (2018) A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, **362**, 1140–1144.

[34] Moore, D. S. dan Notz, W. I. (2021) *The Basic Practice of Statistics*, 9 edition. Macmillan Learning, New York.

[35] Witte, R. S. dan Witte, J. S. (2017) *Statistics*, 11th edition. Wiley, Hoboken, New Jersey.

[36] Goodfellow, I., Bengio, Y., dan Courville, A. (2016) *Deep Learning*. MIT Press, Cambridge, MA, USA. http://www.deeplearningbook.org.

[37] Xu, B., Wang, N., Chen, T., dan Li, M. (2015) Empirical evaluation of rectified activations in convolutional network. *CoRR*, **abs/1505.00853**.

[38] Szandala, T. (2020) Review and comparison of commonly used activation functions for deep neural networks. *CoRR*, **abs/2010.09458**.

[39] Shannon, C. E. (1948) A mathematical theory of communication. *The Bell system technical journal*, **27**, 379–423.

[40] Dumoulin, V. dan Visin, F. (2016) A guide to convolution arithmetic for deep learning. *ArXiv*, **abs/1603.07285**.